数据万象

SDK 文档

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

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

文档目录

SDK 文档
- SDK 概览
- 图片处理 Android SDK
  - 快速入门
  - 基础图片处理
    - 图片处理
    - 使用 Glide 加载图片
  - 加载 TPG 图片
  - 加载 AVIF 图片
- 图片处理 iOS SDK
  - 快速入门
  - 加载 TPG 图片
  - 图片基础处理
    - 图片处理
- COS Android SDK
  - 内容审核
    - 图片审核
    - 视频审核
    - 音频审核
    - 文本审核
    - 文档审核
- C SDK
  - 快速入门
  - 图片处理
    - 图片持久化处理
    - 基础图片处理
    - 图片高级压缩
  - 媒体处理
    - 媒体 bucket 接口
    - 媒体信息接口
    - 媒体截图接口
  - 内容审核
    - 视频审核
  - 内容识别
    - 二维码识别

版权所有：腾讯云计算（北京）有限责任公司

C++ SDK
快速入门
图片处理
图片持久化处理
基础图片处理
盲水印
媒体处理
媒体 bucket 接口
媒体信息接口
媒体截图接口
.NET(C#) SDK
快速入门
图片处理
基础图片处理
盲水印
媒体处理
媒体 bucket 接口
媒体信息接口
媒体截图接口
内容审核
图片审核
视频审核
音频审核
文档审核
文本审核
内容识别
图片二维码
Go SDK
快速入门
图片处理
图片持久化处理
基础图片处理
图片高级压缩
Guetzli 压缩
盲水印
数据万象服务
原图保护
媒体处理

媒体 bucket 接口
媒体信息接口
媒体截图接口
文档预览
查询文档预览开通状态
同步请求接口
异步处理队列接口
异步处理任务接口
文档转 HTML
内容审核
图片审核
视频审核
音频审核
文本审核
AI 内容识别
图片二维码
文件处理
文件解压
文件压缩
文件哈希值计算
COS iOS SDK
内容审核
图片审核
视频审核
音频审核
文本审核
文档审核
网页审核
Java SDK
快速入门
常见问题
图片处理
基础图片处理
图片持久化处理
图片高级压缩
盲水印
AI 内容识别
图片标签

版权所有：腾讯云计算（北京）有限责任公司

- 汽车识别
- 媒体处理
  - 模板操作
  - 工作流操作
  - 视频截帧
- 文档预览
  - 生成同步预览链接
- 内容审核
  - 图片审核
  - 视频审核
  - 音频审核
  - 文档审核
  - 文本审核
- 回调信息处理
- 异常处理
- 文件处理
  - 文件哈希值计算

JavaScript SDK

- 快速入门
- 图片处理
  - 基础图片处理
  - 盲水印
- 媒体处理
  - 媒体 bucket 接口
  - 媒体信息接口
  - 媒体截图接口
- 内容审核
  - 图片审核
  - 视频审核
  - 音频审核
  - 文本审核
  - 网页审核
- 文件处理
  - 文件解压
  - 文件压缩
  - 文件哈希值计算

Node.js SDK

- 快速入门

版权所有：腾讯云计算（北京）有限责任公司

- 图片处理
  - 基础图片处理
  - 盲水印
- 媒体处理
  - 媒体 bucket 接口
  - 媒体信息接口
  - 媒体截图接口

PHP SDK

- 快速入门
- 图片处理
  - 基础图片处理
  - 图片样式
  - 图片持久化处理
  - 盲水印
  - 图片高级压缩
  - Guetzli 压缩
- 媒体处理
  - 媒体 bucket 接口
  - 媒体信息接口
  - 媒体截图接口
  - 私有 M3U8 接口
  - 队列接口
- 文档处理
  - 查询文档预览开通状态
  - 文档转码
    - 同步请求接口
    - 异步处理任务接口
    - 异步处理队列接口
  - 文档转 HTML
- 内容识别
  - 图片标签
  - 图片二维码
- 任务接口
  - 动图任务接口
  - 截图任务接口
  - 转码任务接口
  - 拼接任务接口
  - 智能封面任务接口

版权所有：腾讯云计算（北京）有限责任公司

多任务接口
视频增强任务接口
精彩集锦任务接口
人声分离任务接口
SDR to HDR 任务接口
数字水印任务接口
超分辨率任务接口
视频标签任务接口
图片处理任务接口
转封装任务接口
模板接口
搜索模板接口
文件处理
多文件打包压缩
文件哈希值计算
文件解压
Python SDK
快速入门
图片处理
图片持久化处理
图片高级压缩
盲水印
媒体处理
媒体信息接口
媒体截图接口
内容审核
图片审核
视频审核
音频审核
文本审核
文档审核
网页审核
内容识别
语音识别
任务接口
语音 bucket 接口
队列接口
图片二维码

文档处理
文档转码
同步请求接口
异步处理任务接口
异步处理队列接口
文档转 HTML
查询文档预览开通状态
模板接口
语音识别
文件处理
文件解压
查询文件解压结果
提交文件解压任务
小程序 SDK
快速入门
图片处理
基础图片处理
盲水印
媒体处理
媒体 bucket 接口
媒体信息接口
媒体截图接口
内容审核
直播审核
SDK 产品个人信息保护规则
数据万象 SDK 产品个人信息保护规则

版权所有：腾讯云计算（北京）有限责任公司

SDK 文档

SDK 概览

最近更新时间：2024-03-13 15:17:46

为了开发者更方便、更快速地使用数据万象的基础图片处理和媒体处理功能，以及 CDN 的云闪图片分发功能，我们提供了以下 SDK，开发者可根据具体需求进行选择，详情请参见对应的快速入门文档。对象存储的 SDK 也集成了数据万象的数据处理功能，若您需要使用其他语言的 SDK，例如 C++ 、JavaScript 等，请参见 COS SDK 概览。

SDK	接入文档
Android SDK	快速入门
C SDK	快速入门
C++ SDK	快速入门
.NET(C#) SDK	快速入门
Go SDK	快速入门
iOS SDK	快速入门
Java SDK	快速入门
JavaScript SDK	快速入门
Node.js SDK	快速入门
PHP SDK	快速入门
Python SDK	快速入门
小程序 SDK	快速入门

说明：

关于 TPG 的更多介绍，请参见 TPG 功能介绍。

版权所有：腾讯云计算（北京）有限责任公司

图片处理 Android SDK 快速入门

最近更新时间：2024-03-14 09:20:49

简介

CI Android SDK 主要提供了 AVIF 和 TPG 的图片解码器以及数据万象的图片基础处理功能。

模块	功能
AVIF	AVIF 图片解码器以及 Glide、Fresco 等生态封装。
TPG	TPG 图片解码器以及 Glide、Fresco 等生态封装。
cloud-infinite	该模块包含数据万象的基础图片处理操作，并支持各个操作能够相互组合，构建 URL 进行网络请求。

说明：

图片压缩瘦身介绍请参见 图片压缩瘦身。

CDN 云闪图片分发介绍请参见 云闪图片分发。

关于 AVIF 的更多介绍，请参见 AVIF 功能介绍。

关于 TPG 的更多介绍，请参见 TPG 功能介绍。

相关资源

SDK 源码和 Demo 请参见 CI Android SDK。

SDK 更新日志请参见 ChangeLog。

准备工作

1. 您需要一个 Android 应用，这个应用可以是您现有的工程，也可以是您新建的一个空的工程。
2. 请确保应用基于 Android 4.0.3 及以上版本的 SDK 构建。

集成与使用

根据您需要的模块查看具体的使用文档，进行集成和使用。

AVIF 解码器：

AVIF 解码器基础使用，请参见 AVIF 解码器。

TPG 解码器：

TPG 解码器基础使用，请参见 TPG 解码器。

数据万象的图片基础处理：

图片基础处理，请参见 图片处理。

基础图片处理
图片处理

最近更新时间：2024-03-13 15:17:11

简介

数据万象提供两种方式进行图片处理：图片自适应处理 ResponsiveTransformation 和自定义转换处理 CITransformation。

说明：

支持格式：支持处理 JPG、BMP、GIF、PNG、WEBP 格式，并且支持 HEIF 格式的解码和处理操作。

体积限制：处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

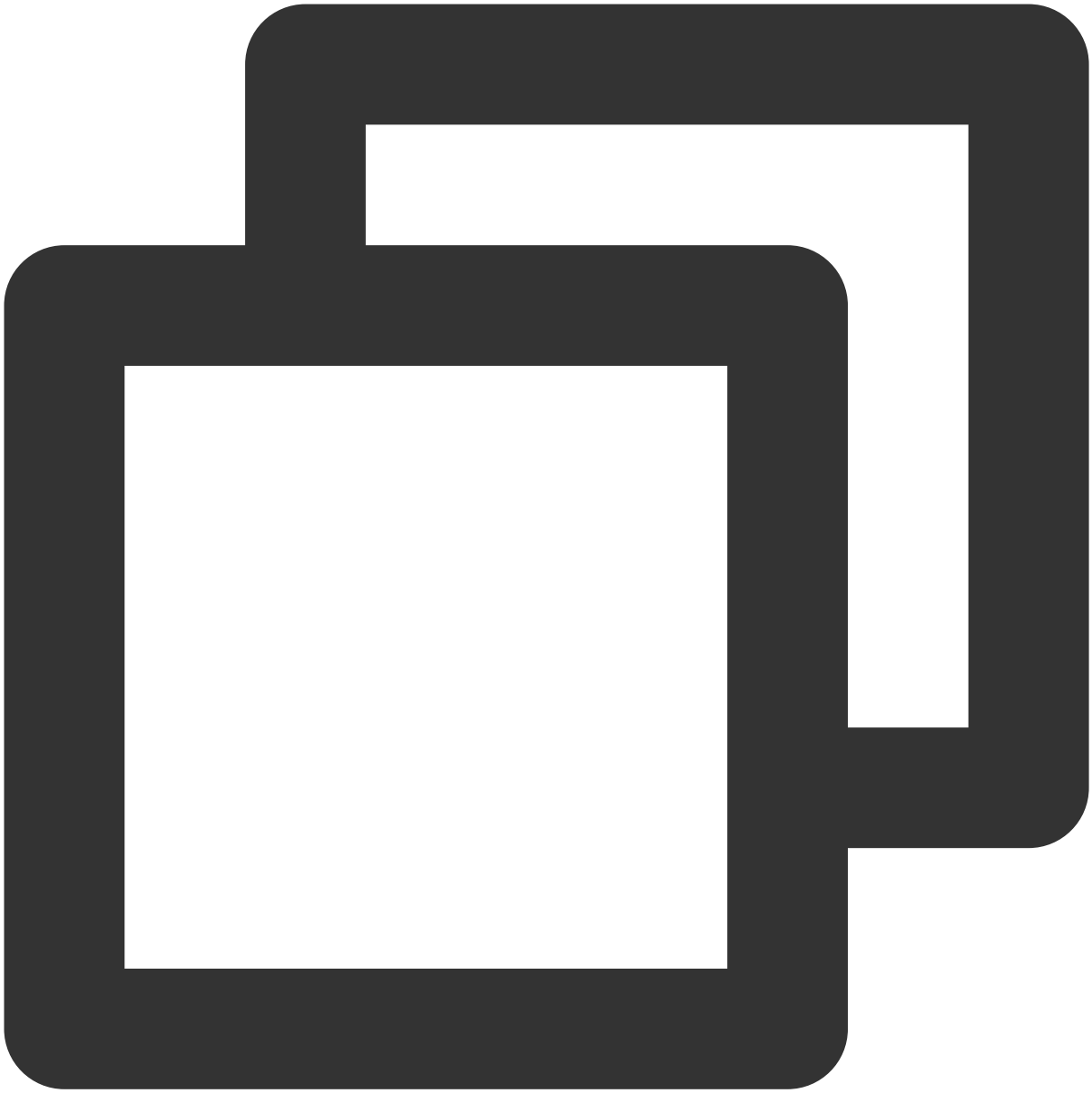
动图帧数限制：gif 帧数限300帧。

图片自适应处理 ResponsiveTransformation

图片自适应处理包括了格式转换（根据系统版本自动变换格式）和缩放（根据图片控件尺寸以及 ScaleType 进行缩放），相关说明如下。

1. 自适应加载 ResponsiveTransformation。
2. 根据当前 imageView 控件尺寸和 ScaleType 自动调整图片大小。
3. 根据系统版本自动调整最优图片格式，Android 4.3以下加载原格式，Android 4.3（包含4.3）到 Android 10.0（不包含10.0）加载 WEBP 格式，Android 10.0及以上加载 HEIC 格式。

版权所有：腾讯云计算（北京）有限责任公司



```
ResponsiveTransformation transformation = new ResponsiveTransformation(imageview);
```

自定义转换处理 CITransformation

cloud-infinite SDK 中的 CITransformation 类封装了数据万象的基础图片处理功能其中包含如下功能：

缩放

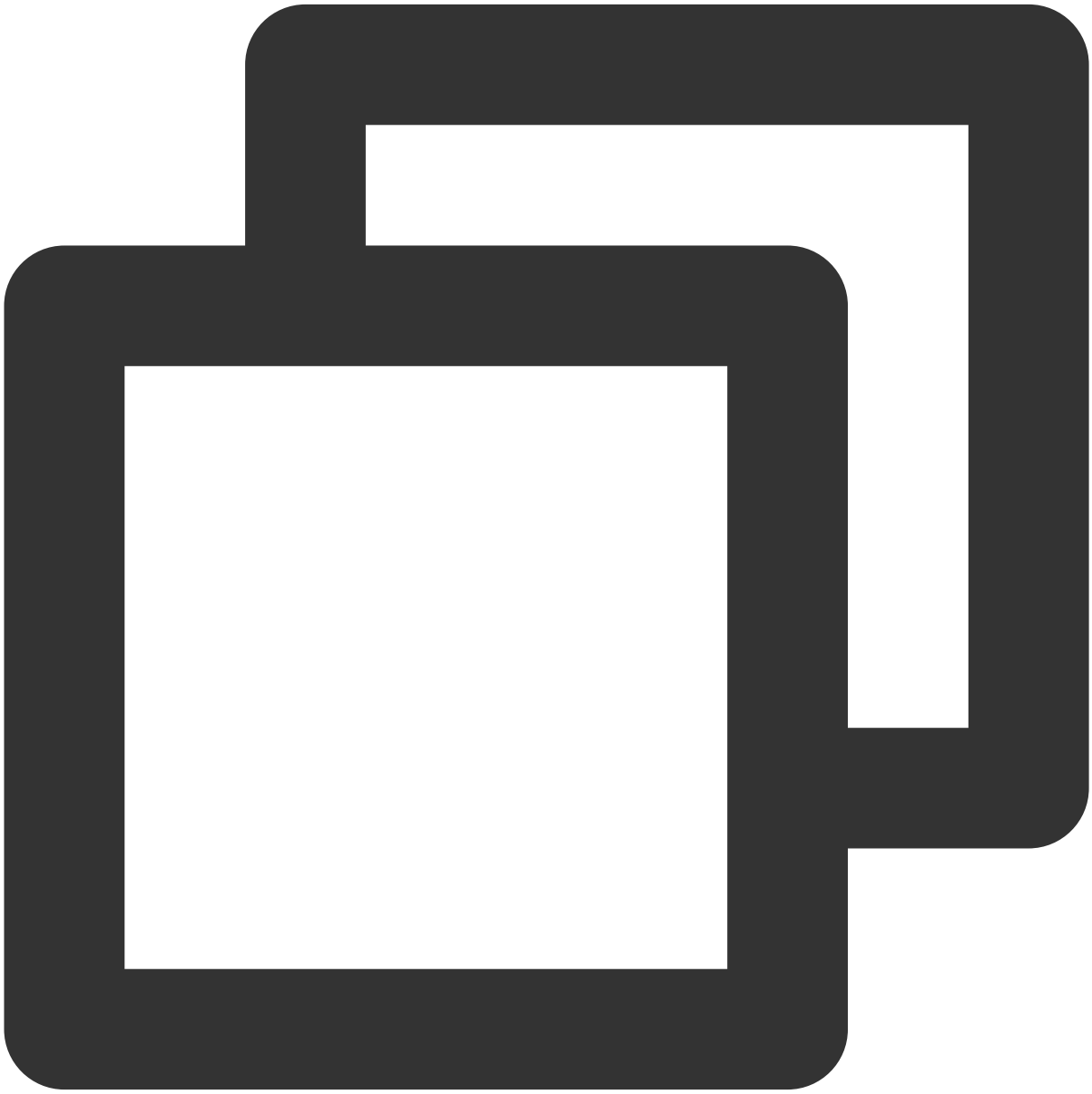
裁剪

旋转

格式转换

质量变换
高斯模糊
锐化
水印
去除图片元信息
组合操作

在使用数据万象基础图片处理功能时首先实例化 CITransformation 类，下面所有操作一致，不再重复说明。



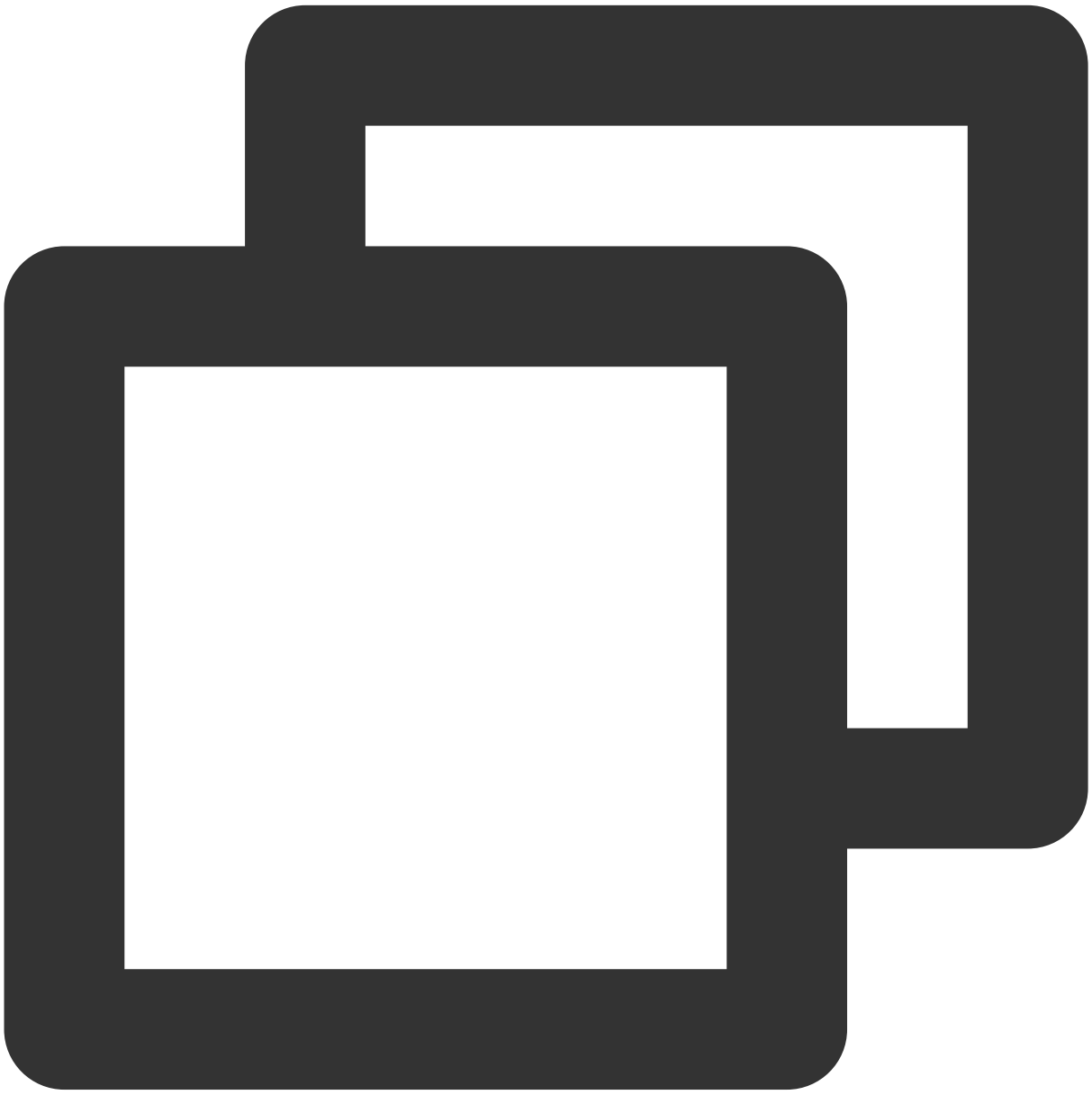
```
CITransformation transform = new CITransformation();
```

缩放

说明：

API 文档请参见 缩放功能接口。

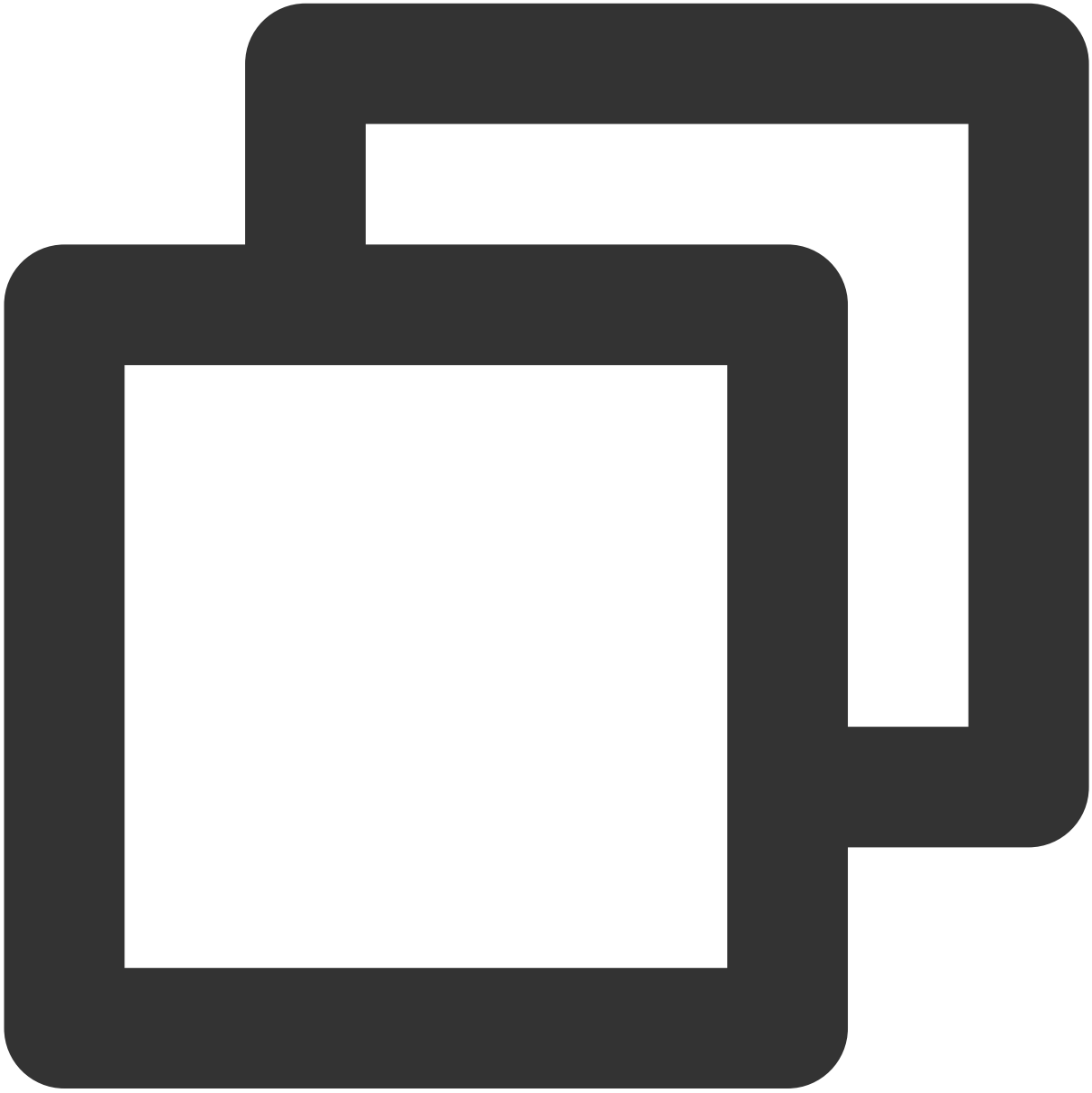
按百分比缩放



```
//指定图片的宽为原图的 50%，高度不变
transform.thumbnailByWidthScale(50);
//指定图片的高为原图的 50%，宽度不变
```

```
transform.thumbnailByHeightScale(50);
//指定图片的宽高为原图的 50%
transform.thumbnailByScale(50);
```

指定宽高缩放

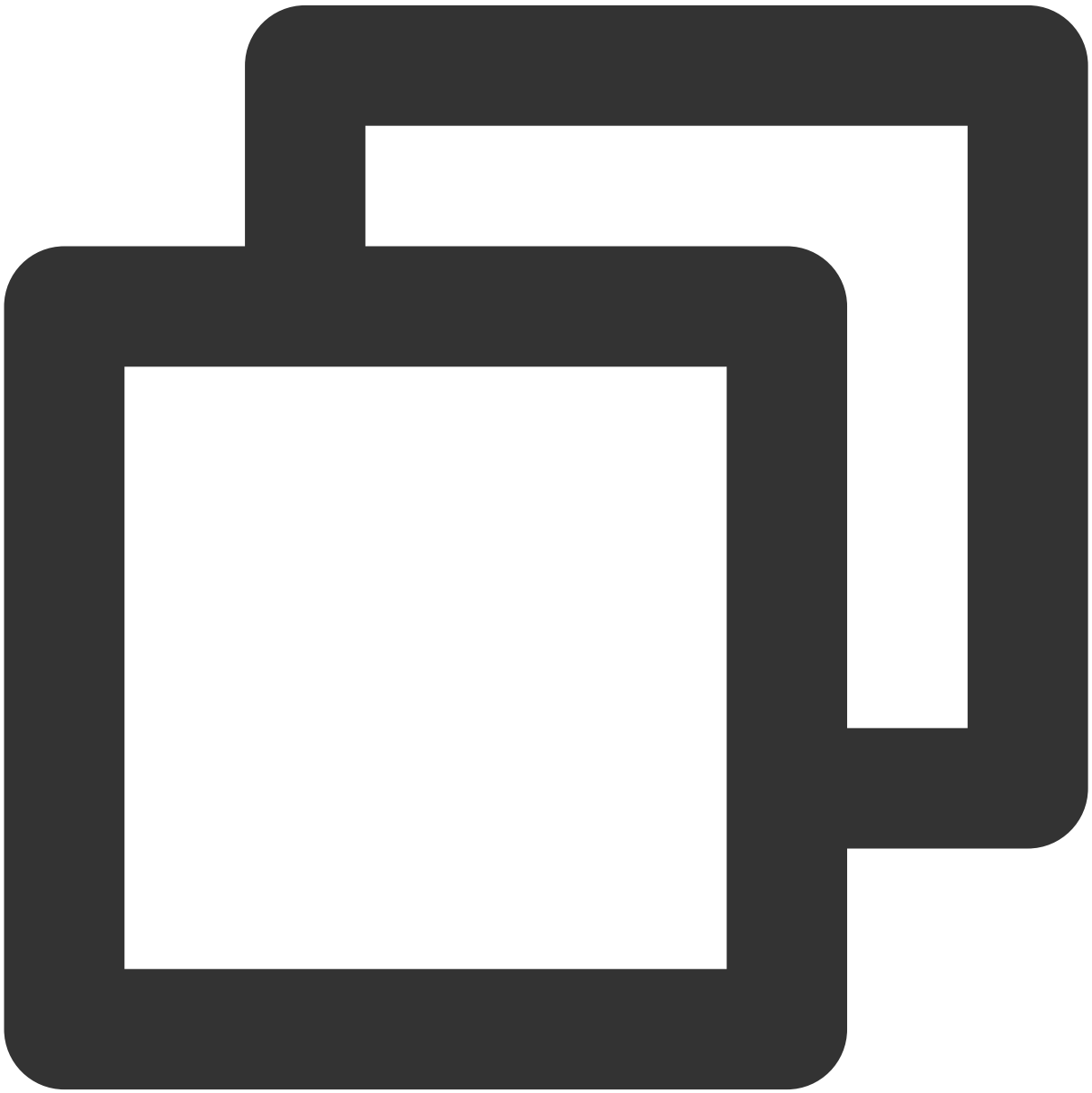


```
//指定目标图片宽度为50，高度等比压缩
transform.thumbnailByWidth(50);
//指定目标图片高度为50，宽度等比压缩
transform.thumbnailByHeight(50);
//限定缩略图的宽度和高度的最小值分别为50和50，进行等比缩放
```

```
transform.thumbnailByMinWH(50, 50);
//限定缩略图的宽度和高度的最大值分别为50和50，进行等比缩放
transform.thumbnailByMaxWH(50, 50);
//忽略原图宽高比例，指定图片宽度为50，高度为50 ，强行缩放图片，可能导致目标图片变形
transform.thumbnailByWH(50, 50);
```

等比缩放

限制像素缩放，缩放后的图像，总像素数量不超过指定数量。



```
//等比缩放图片，缩放后的图像，总像素数量不超过1000
```

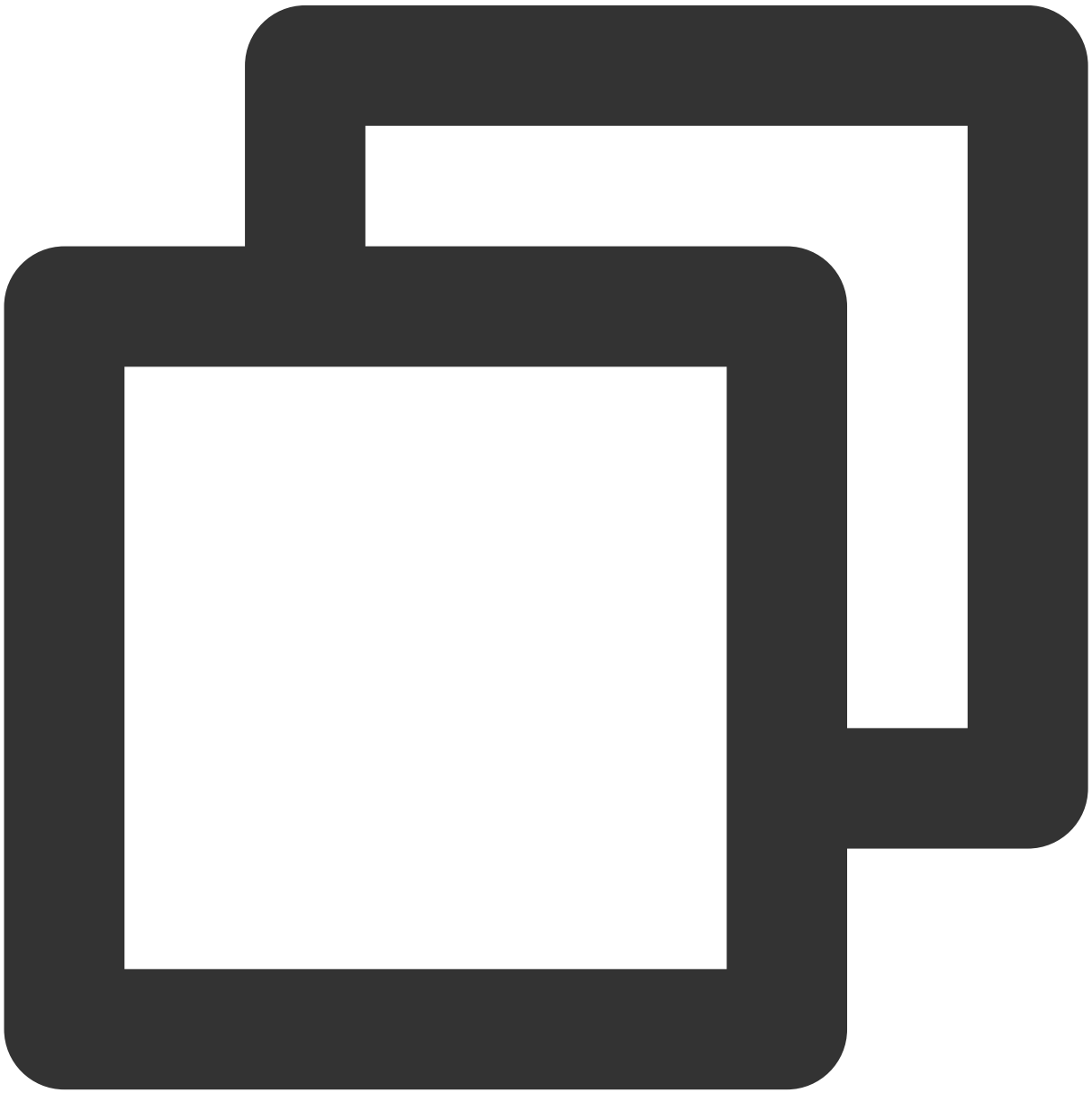
```
transform.thumbnailByPixel(1000);
```

裁剪

说明：

API 文档请参见 裁剪功能接口。

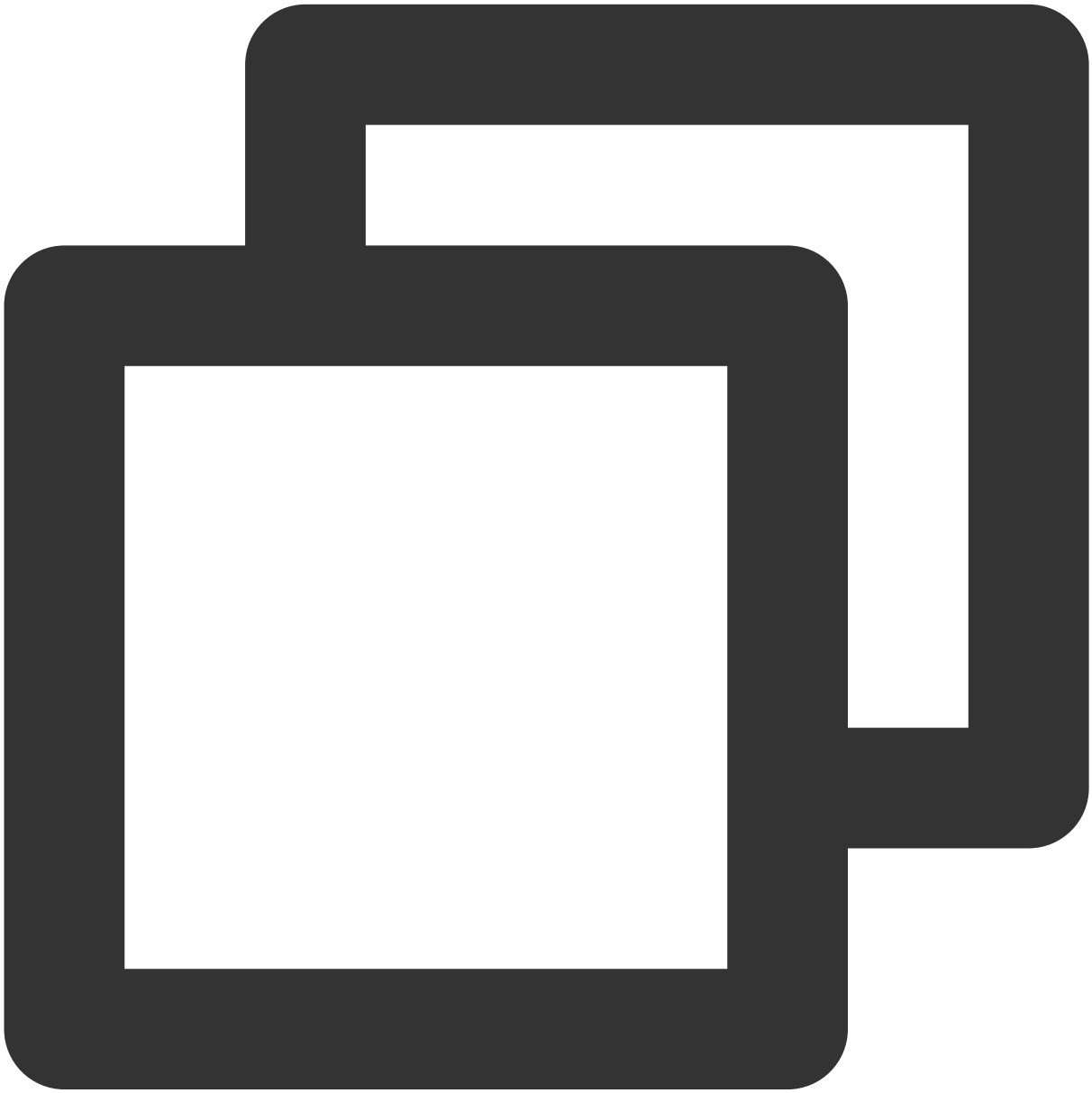
普通裁剪



```
// 指定目标图片宽度、高度、相对于图片左上顶点水平向右偏移、相对于图片左上顶点水平向下偏移进行裁剪
transform.cut(100, 300, 30, 30);
```

内切圆裁剪

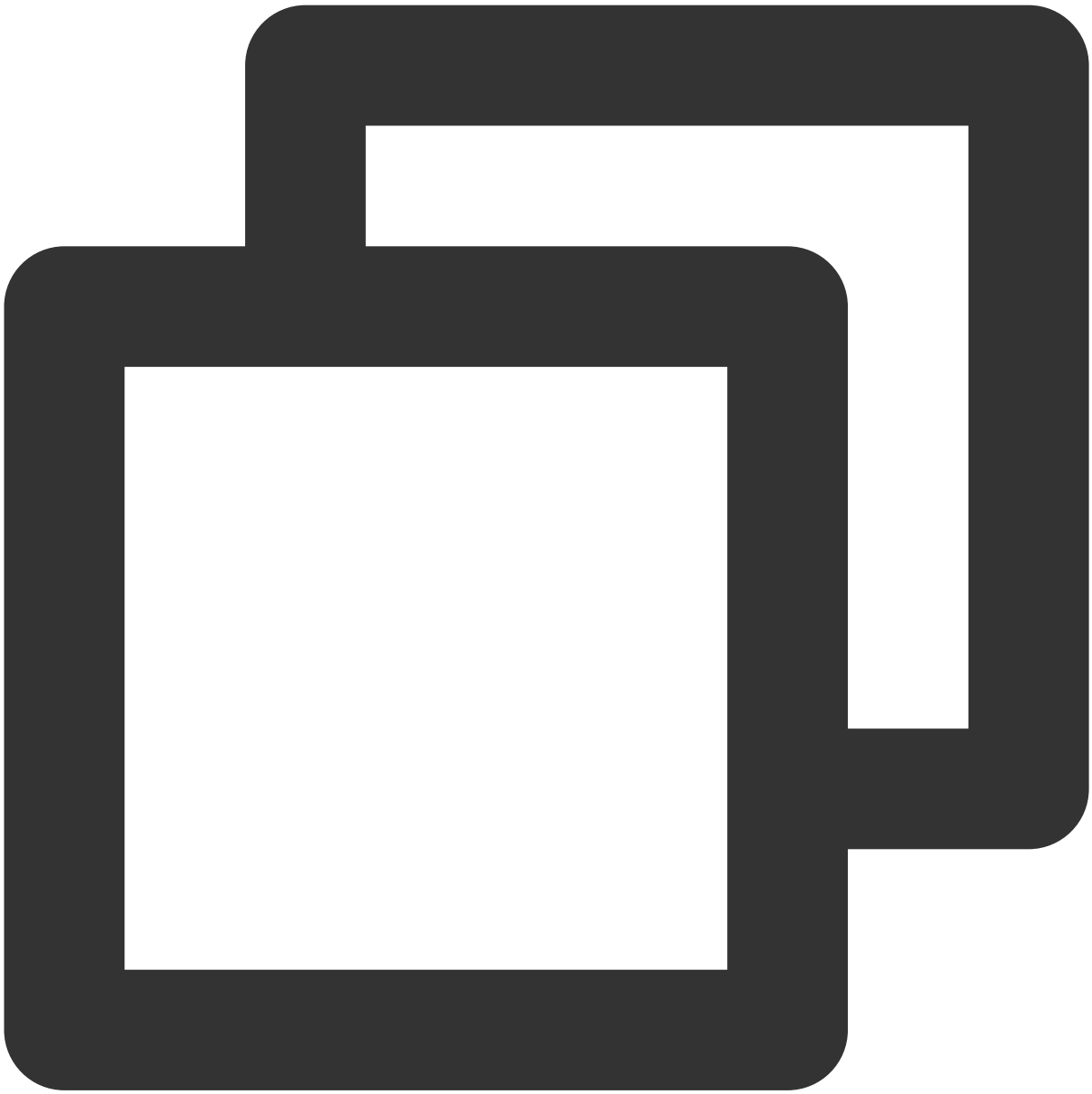
radius 是内切圆的半径，取值范围为大于0且小于原图最小边一半的整数。内切圆的圆心为图片的中心。图片格式为 GIF 时，不支持该操作。



```
// 指定半径100为例
transform.iradius(100);
```

圆角裁剪

radius 为图片圆角边缘的半径，取值范围为大于0且小于原图最小边一半的整数。圆角与原图边缘相切。图片格式为 GIF 时，不支持该操作。

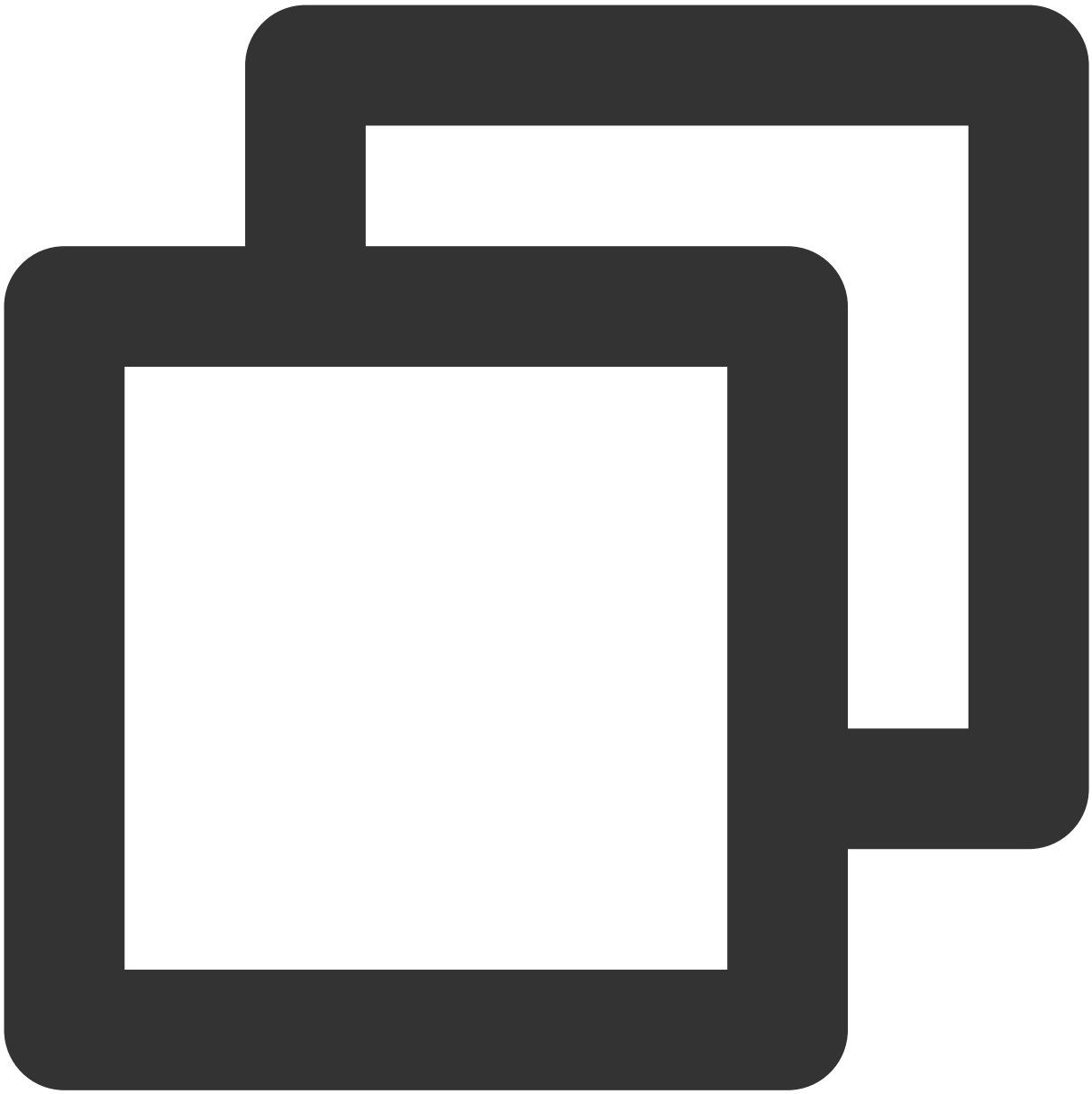


```
// 指定圆角半径100为例
transform.rradius(100);
```

缩放裁剪

说明：

方位图说明，请参见 gravity 介绍。



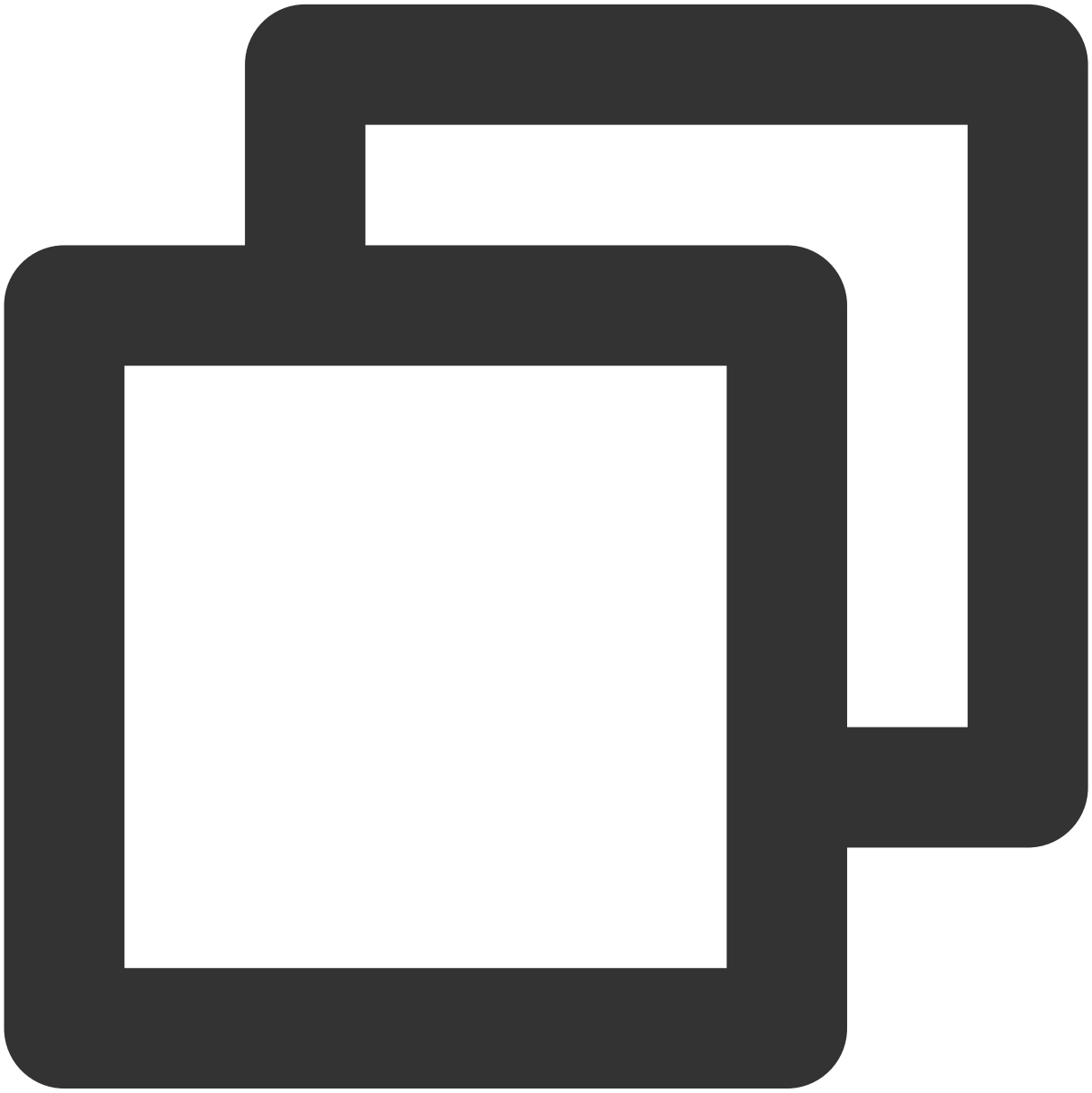
```
//按照指定目标宽度和高度进行缩放裁剪
transform.cropByWH(100, 100);
//按照指定目标宽度和高度进行缩放裁剪，指定操作的起点位置
transform.cropByWH(100, 100, CIGravity.CENTER);
//按照指定目标宽度进行缩放裁剪
transform.cropByWidth(100);
//按照指定目标宽度进行缩放裁剪，指定操作的起点位置
transform.cropByWidth(100, CIGravity.CENTER);
//按照指定目标高度进行缩放裁剪
```

版权所有：腾讯云计算（北京）有限责任公司

```
transform.cropByHeight(100);
//按照指定目标高度进行缩放裁剪，指定操作的起点位置
transform.cropByHeight(100, CIGravity.CENTER);
```

人脸智能裁剪

基于图片中的人脸位置进行缩放裁剪。目标图片的宽度为 Width、高度为 Height。



```
// 裁剪人脸并宽高指定100缩放为例；
transform.scrop(100, 100);
```

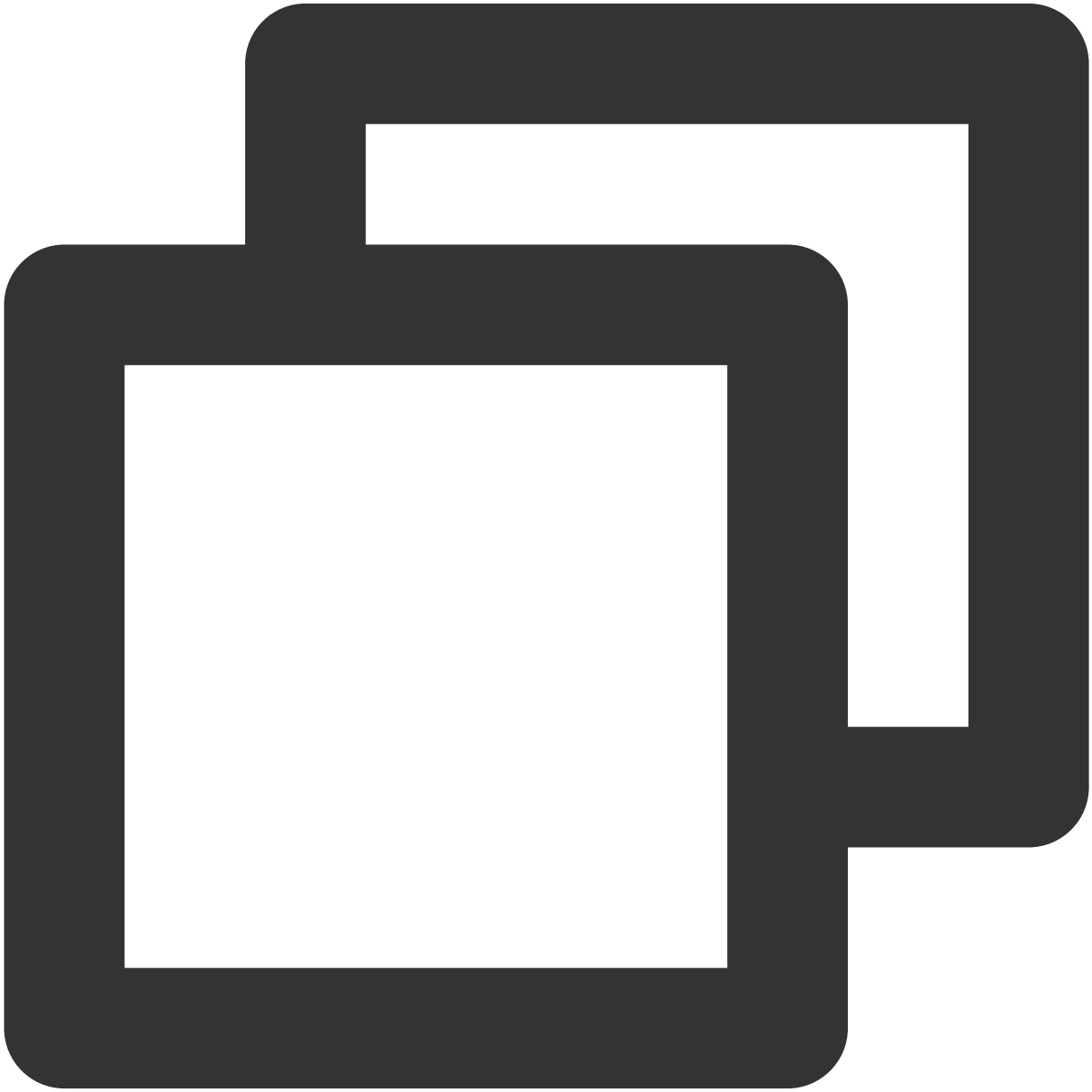
旋转

说明：

API 文档请参见 旋转功能接口。

普通旋转

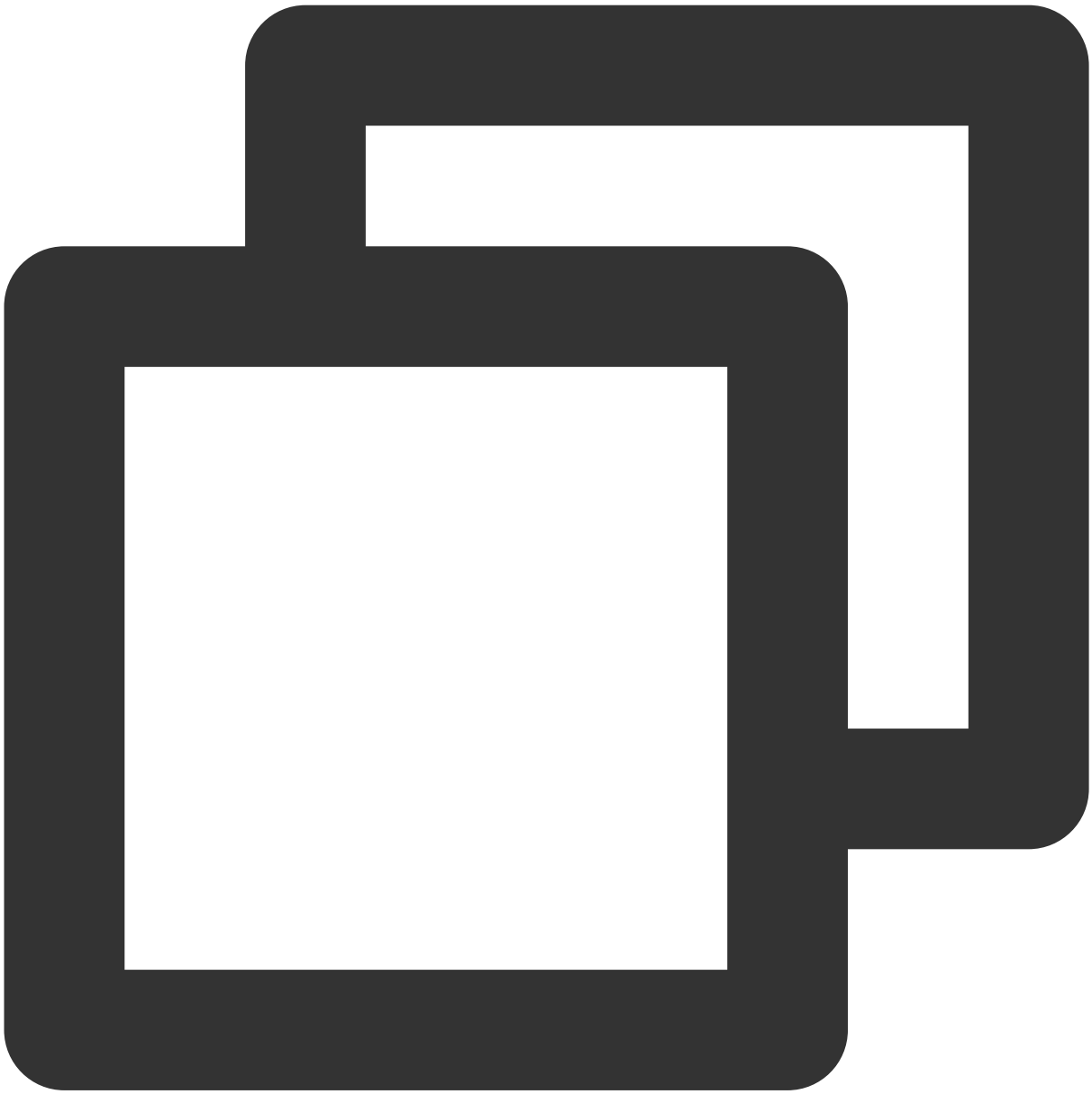
图片顺时针旋转角度，取值范围0 - 360 ，默认不旋转。



```
// 以旋转45度为例；
transform.rotate(45);
```

自适应旋转

根据原图 EXIF 信息将图片自适应旋转回正。



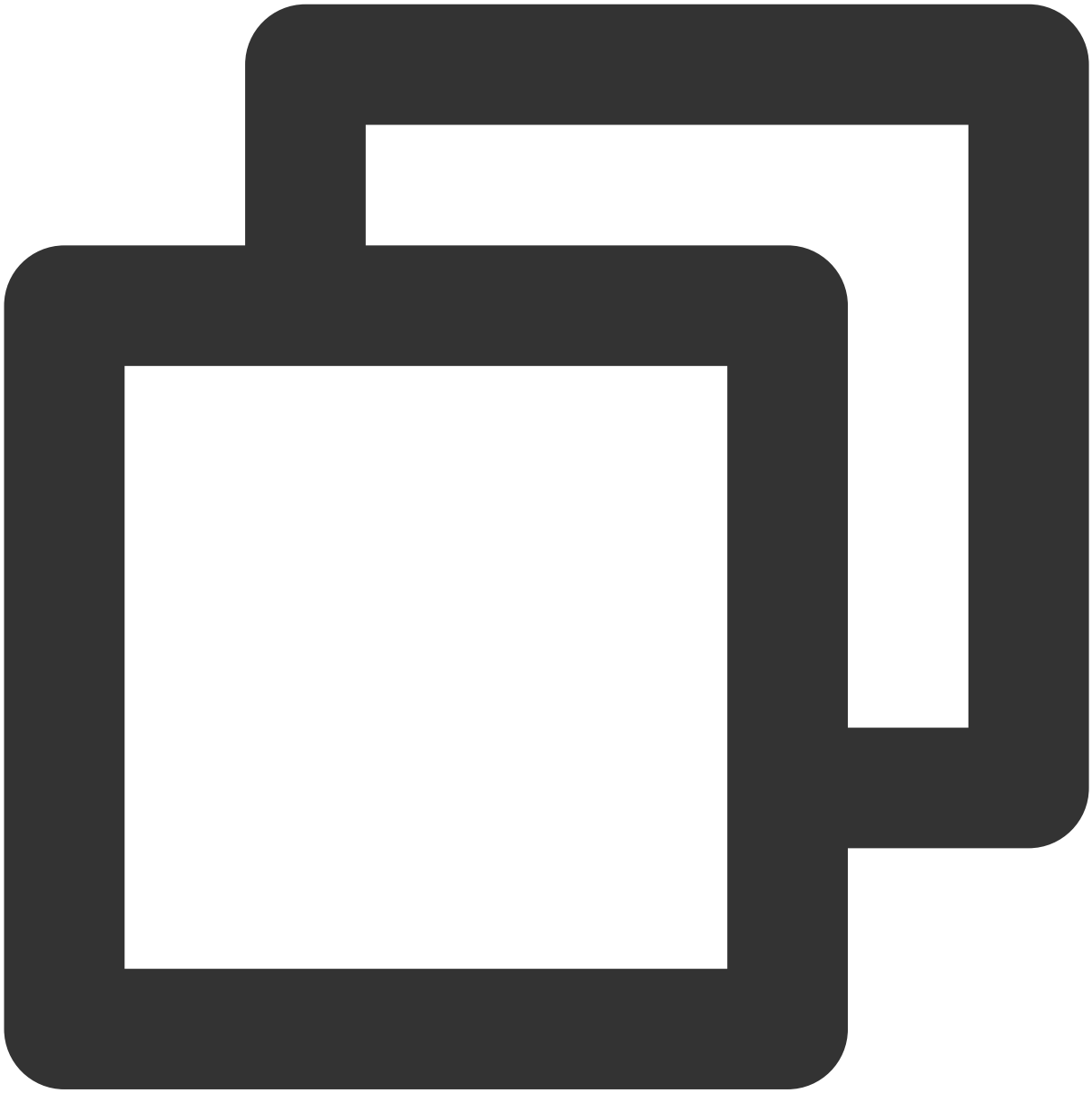
```
transform.rotate();
```

格式转换

说明：

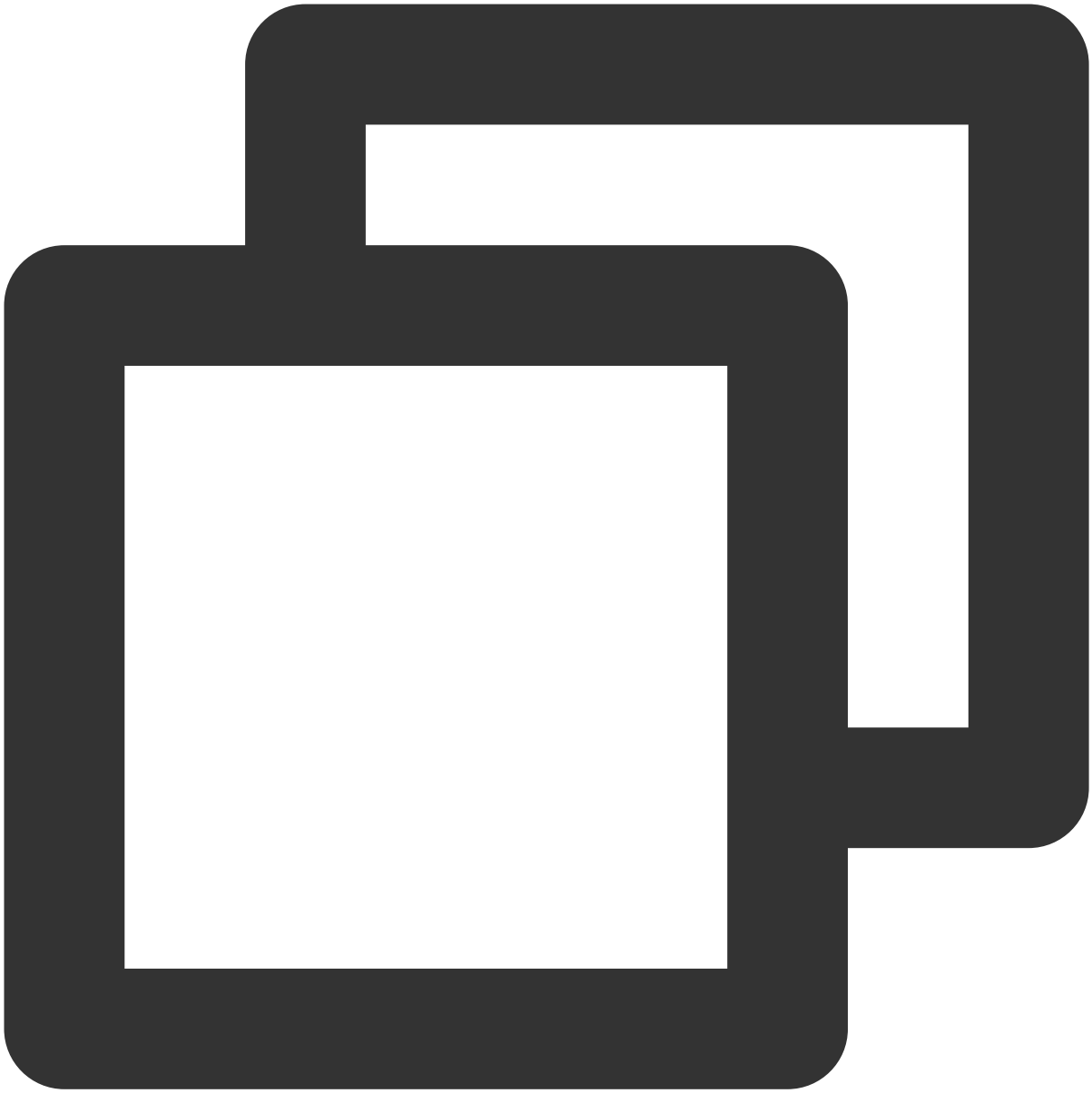
API 文档请参见 格式转换接口。

目标缩略图的图片格式可为：TPG，JPG，BMP，GIF，PNG，HEIC，YJPEG 等，其中 YJPEG 为数据万象针对 JPEG 格式进行的优化，本质为 JPG 格式；缺省为原图格式。



```
// 以转换为 JPEG 为例
 transform.format(CIImageFormat.JPEG, CIImageLoadOptions.LoadTypeUrlFooter);
```

CIImageLoadOptions



```
// 加载类型一 带 accept 头部 accept:image/xxx
LoadTypeAcceptHeader,

// 加载类型二 在URL后面拼接 imageMogr2/format/xxx
LoadTypeUrlFooter,
```

说明：

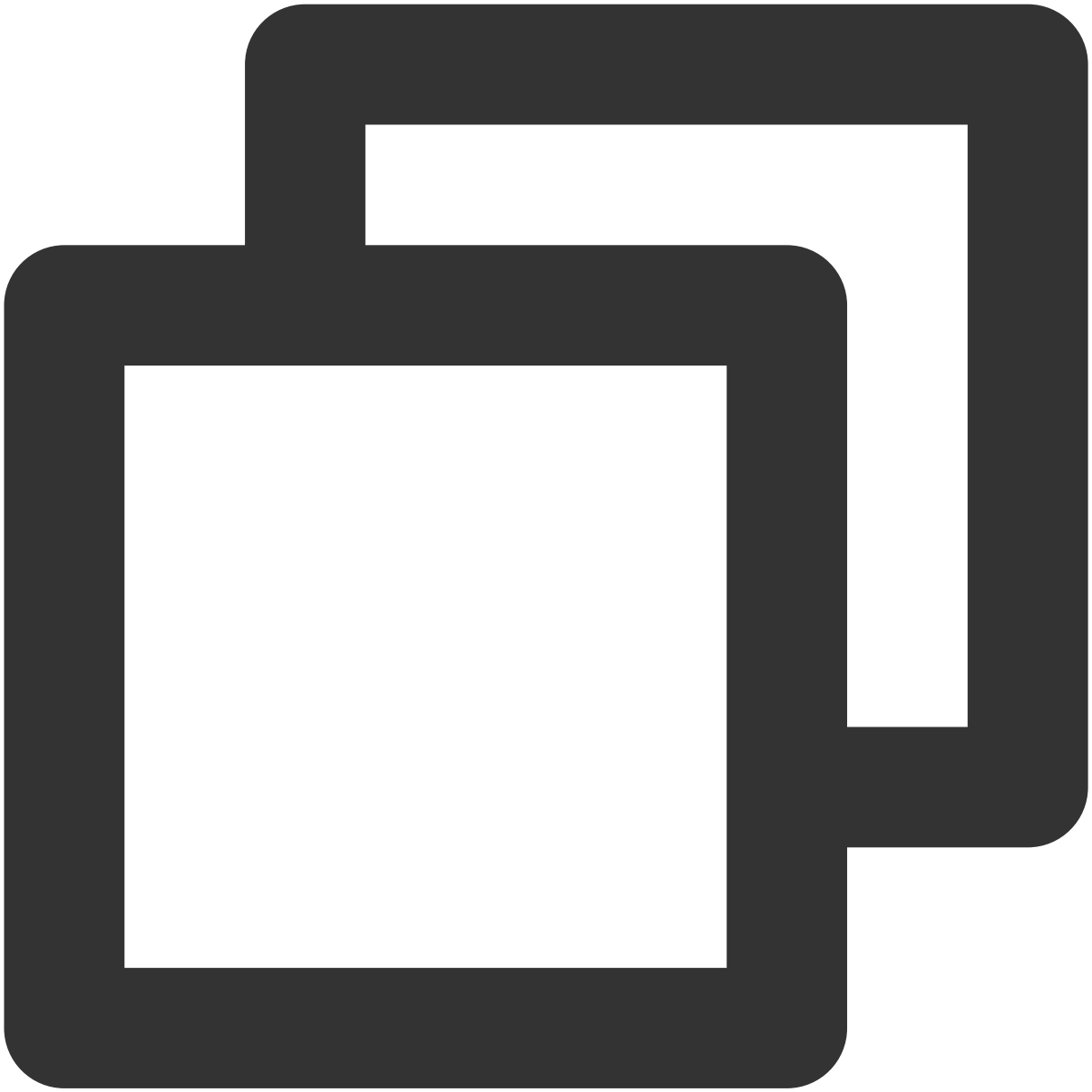
使用 WEBP 格式，需要在 Android 4.3及以上。

使用 HEIC 格式，需要在 Android 10及以上。

不支持将 GIF 格式图片转换为 HEIF 格式。

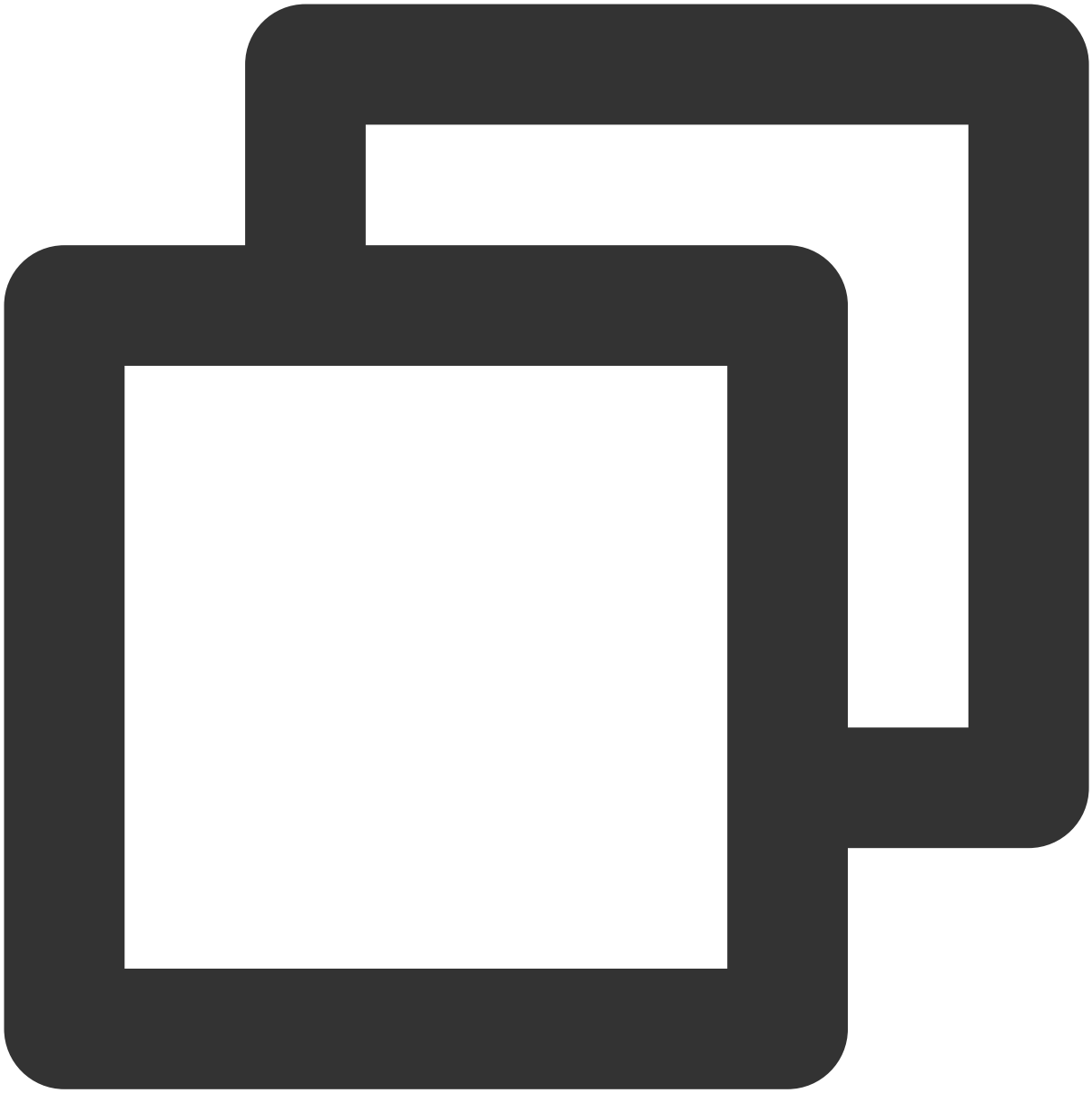
版权所有：腾讯云计算（北京）有限责任公司

当指定为 LoadTypeAcceptHeader 方式传参时，并且组合了其他的转换则 header 失效，并且在 sdk 内部自动转换为 footer 的方式。
在使用图片格式转换时，如果需要转为 TPG 格式，则需要依赖 tpg SDK，如下所示。



```
implementation 'com.tencent.qcloud:tpg:1.3.1'
```

使用 TPG 时，安装时会自动包含 SO 库，建议在 Module 的 build.gradle 文件中使用 NDK 的“abiFilter”配置，设置支持的 SO 库架构。



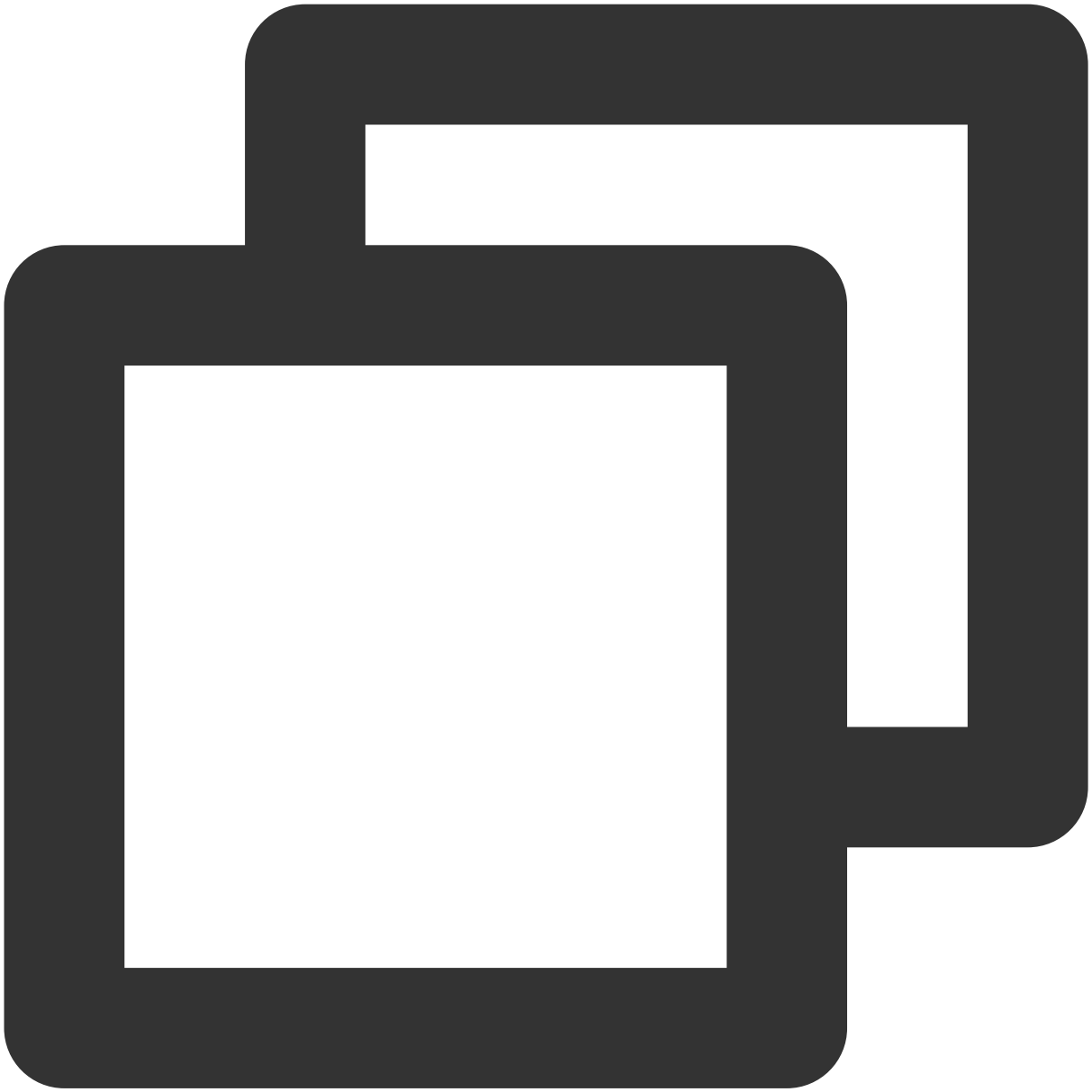
```
defaultConfig {
    ndk {
       // 设置支持的 SO 库架构
        abiFilters 'armeabi' //, 'x86', 'armeabi-v7a', 'x86_64', 'arm64-v8a'
    }
}
```

GIF 格式优化

只针对原图为 GIF 格式，对 GIF 图片格式进行的优化，降帧降颜色。

FrameNumber=1，则按照默认帧数30处理，如果图片帧数大于该帧数则截取。

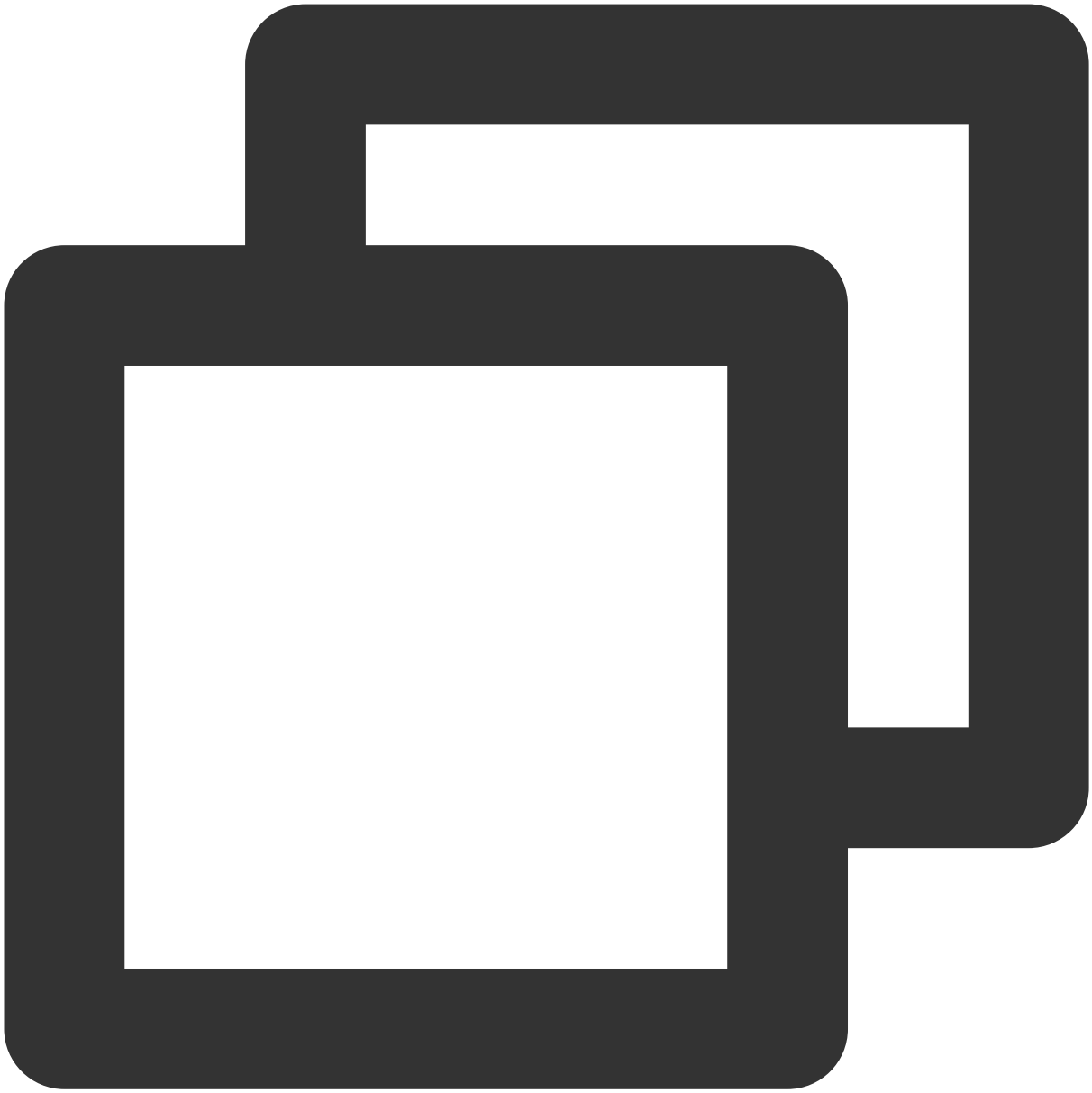
FrameNumber 取值(1,100]，则将图片压缩到指定帧数 （FrameNumber）。



```
transform.gifOptimization(50);
```

输出渐进式 JPG 格式

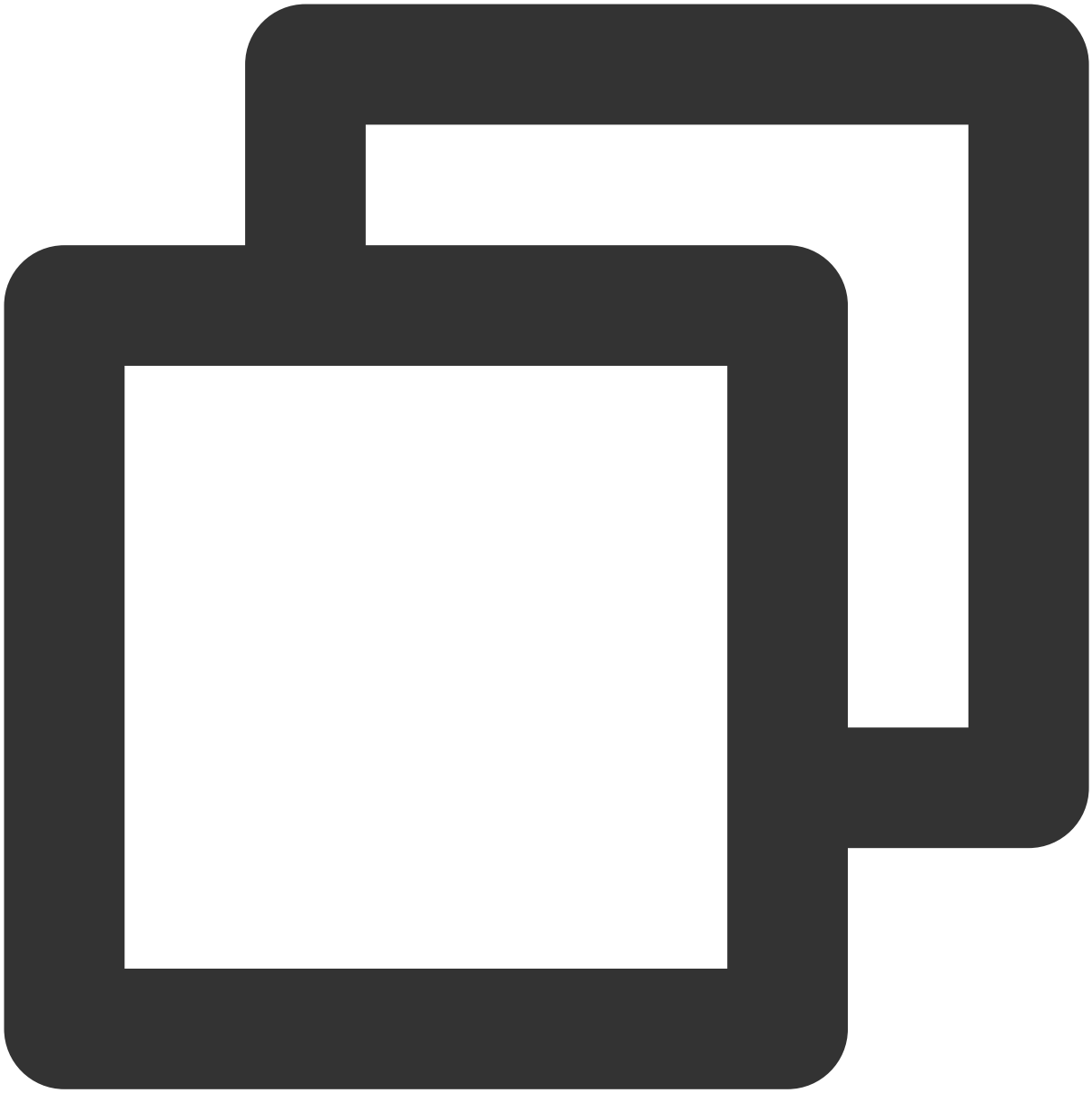
该操作仅在输出图片格式为 JPG 格式时有效。如果输出非 JPG 图片格式，会忽略该操作。



```
transform.jpegInterlaceMode(true);
```

质量变换

数据万象提供三种变换类型：绝对变换、相对变换、最低质量变换，可调节图片质量。API 文档请参见 质量变换接口。



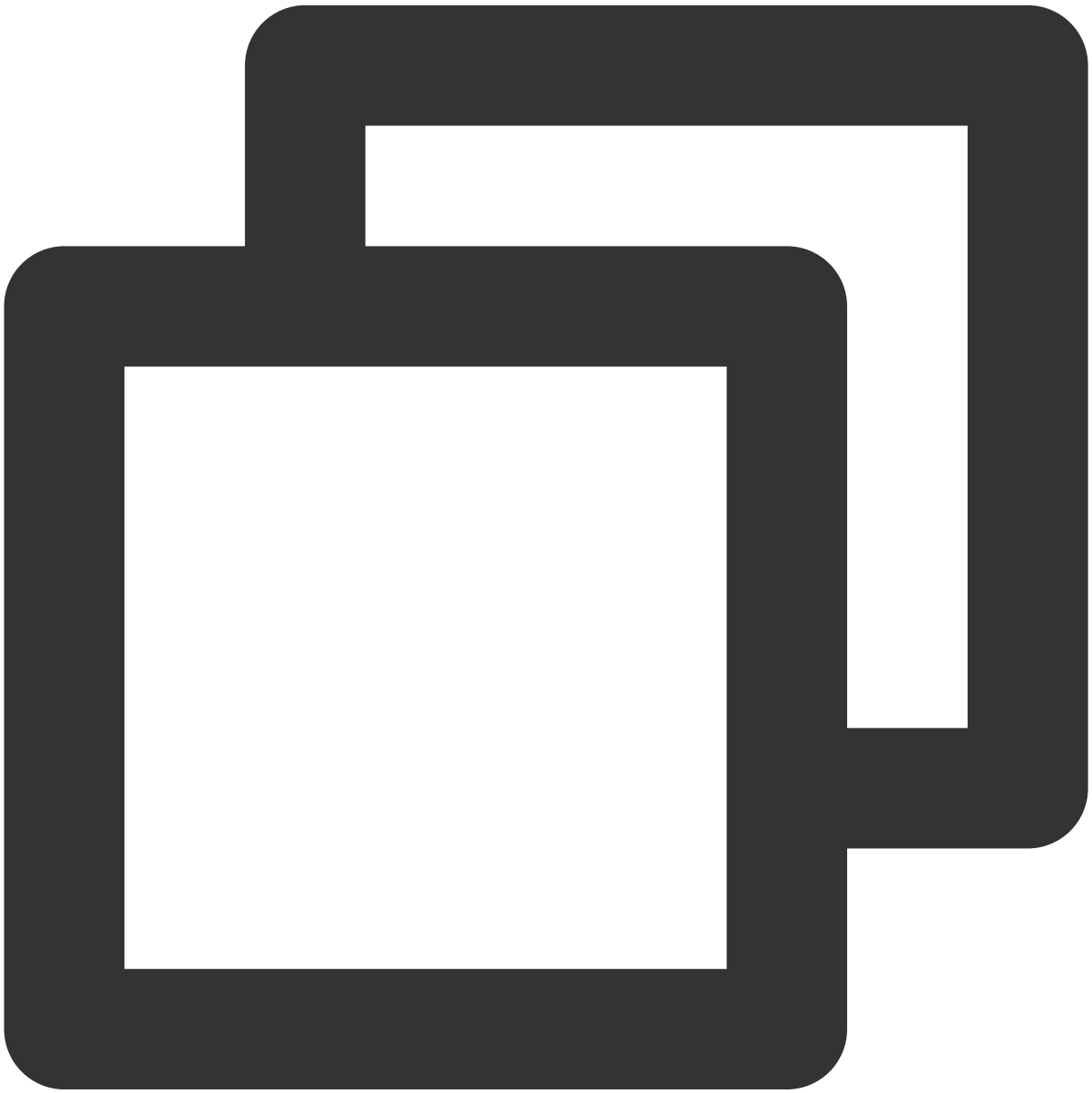
```
//设置图片的绝对质量，取值范围0 - 100 ，默认值为原图质量；取原图质量和指定质量的最小值
transform.quality(90);
// 图片的最低质量，取值范围0 - 100 ，设置结果图的质量参数最小值。
// 例如原图质量为85，将 lquality 设置为80后，处理结果图的图片质量为85。
// 例如原图质量为60，将 lquality 设置为80后，处理结果图的图片质量会被提升至80
transform.lowestQuality(90);
//设置图片的相对质量，取值范围0 - 100 ，数值以原图质量为标准。例如原图质量为80，将 rquality 设置
transform.relativelyQuality(80);
```

说明：

仅适用于 JPG 和 WEBP 格式的图片。

高斯模糊

高斯模糊功能，模糊半径，取值范围为1 - 50；正态分布的标准差，必须大于0。API 文档请参见 高斯模糊接口。

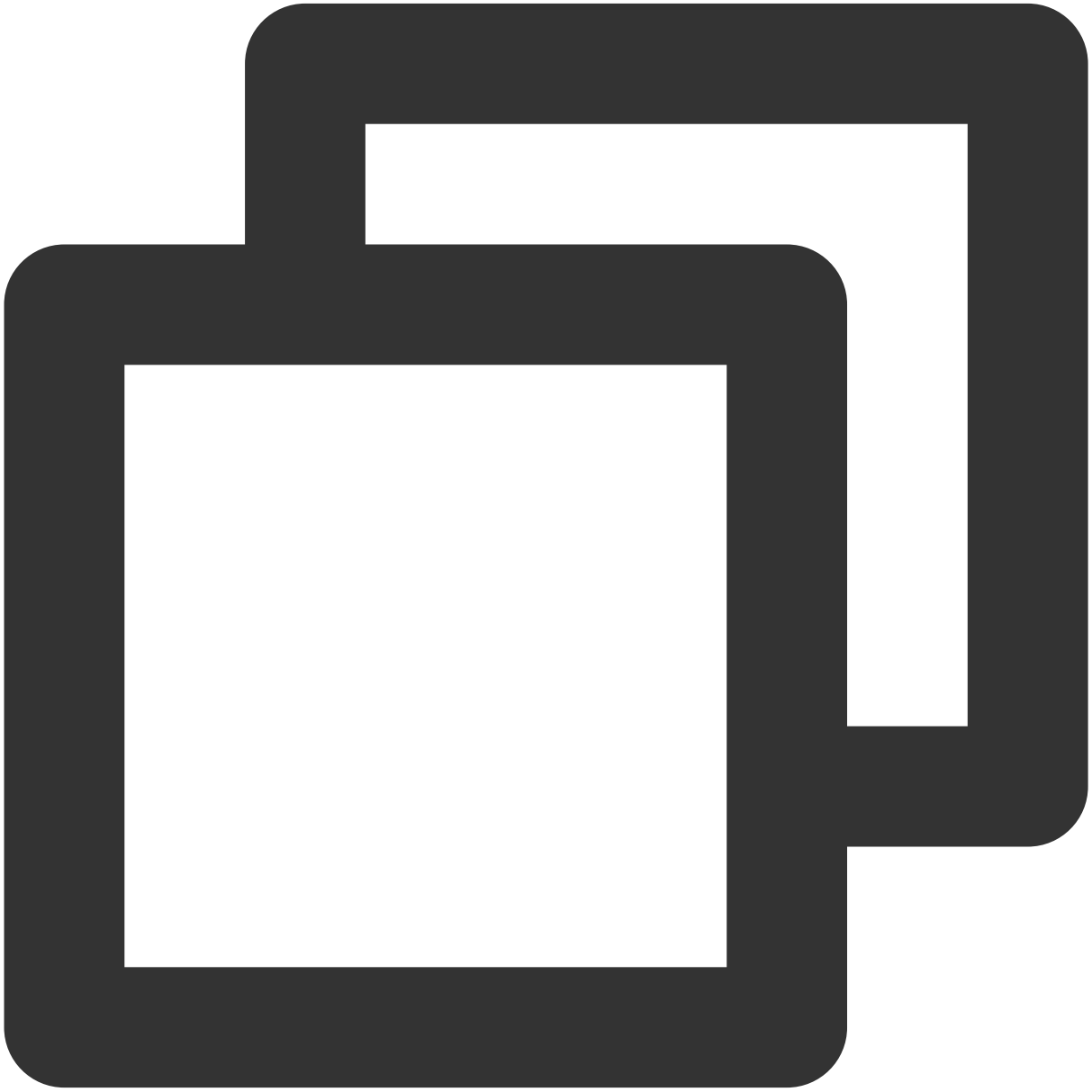


```
// 模糊半径20，正态分布的标准差20为例
transform.blur(20, 20);
```

说明：
图片格式为 GIF 时，不支持该操作。

锐化

图片锐化功能，value 为锐化参数值，取值范围为10 - 300间的整数，推荐使用70。参数值越大，锐化效果越明显。
API 文档请参见 锐化接口。



```
// 以锐化值70为例
```

```
transform.sharpen(70);
```

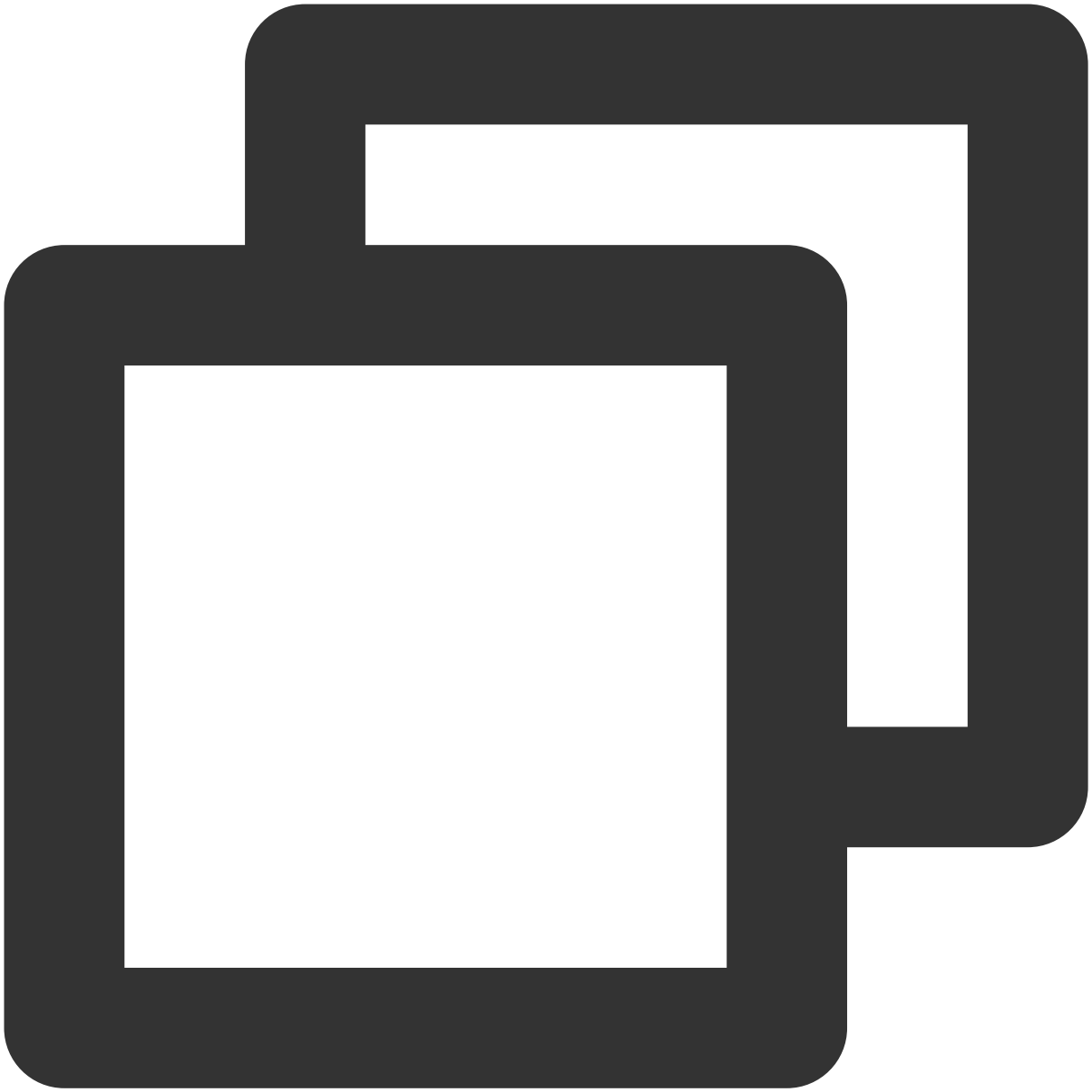
水印

说明：

API 文档请参见 图片水印接口 和 文字水印接口。

方位图说明，请参见 gravity。

图片水印



```
WatermarkImageTransform watermarkImageTransform = new WatermarkImageTransform
        //水印图片地址
        .Builder(imageurl)
        //设置水印九宫格位置
        .setGravity(CIGravity.CENTER)
        //设置水印图适配功能，适用于水印图尺寸过大的场景（如水印墙）
        .setBlogo(1)
        //设置水平（横轴）边距
        .setDistanceX(10)
        //设置垂直（纵轴）边距
        .setDistanceY(10)
```

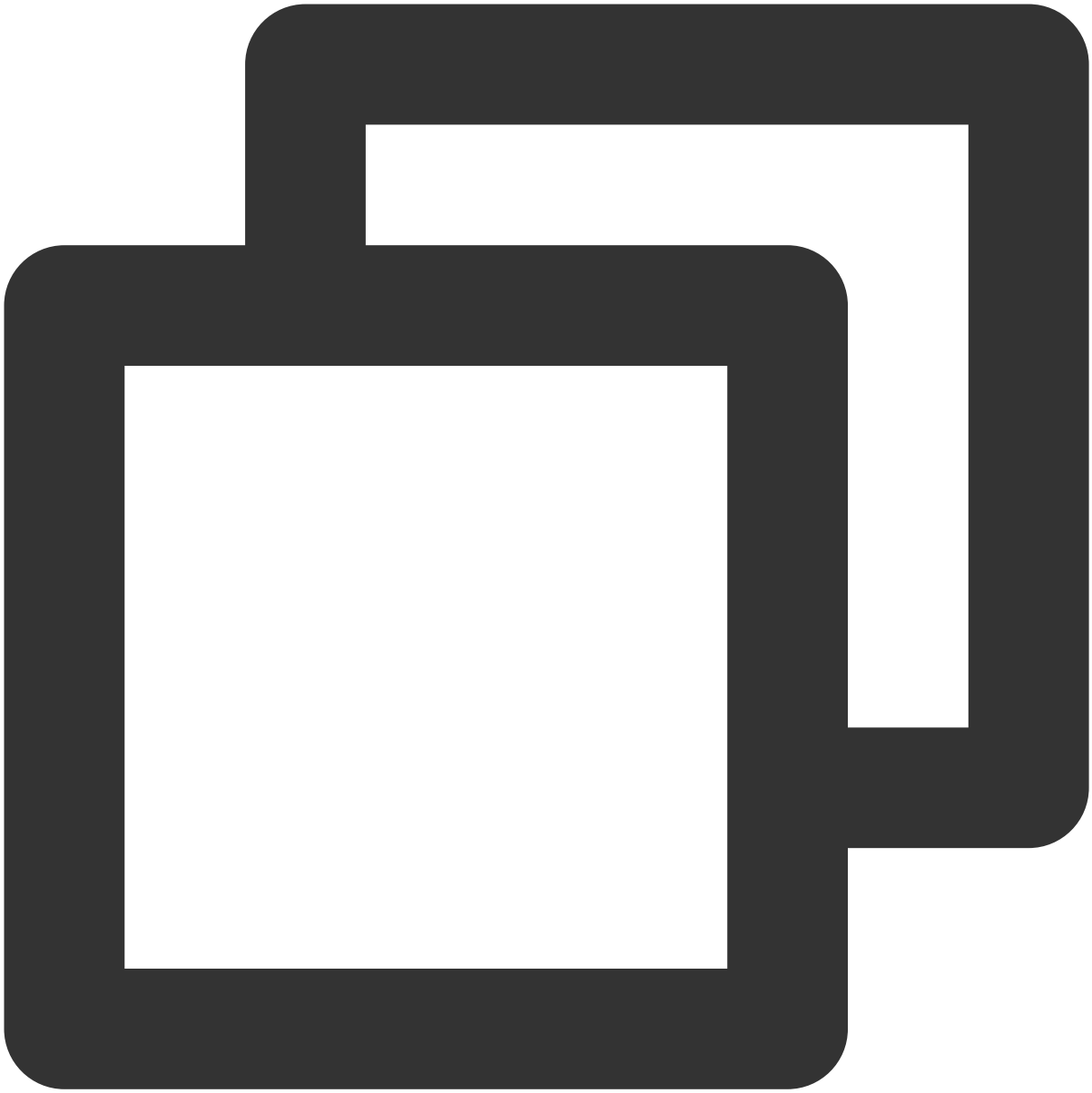
```
            .builder();
    transform.watermarkImage(watermarkImageTransform);
```

说明：

指定的水印图片必须同时满足如下3个条件：

1. 水印图片与源图片必须位于同一个存储桶下。
2. URL 需使用 COS 域名（不能使用 CDN 加速域名，例如 examplebucket-1250000000.file.myqcloud.com/shuiyin_2.png 不可用 ），且需保证水印图可访问。如果水印图读取权限为私有，则需要携带有效签名）。
3. URL 必须以 `http://` 开始，不能省略 HTTP 头，也不能填 HTTPS 头，例如 `examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` ， `https://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` 为非法的水印 URL。

文字水印

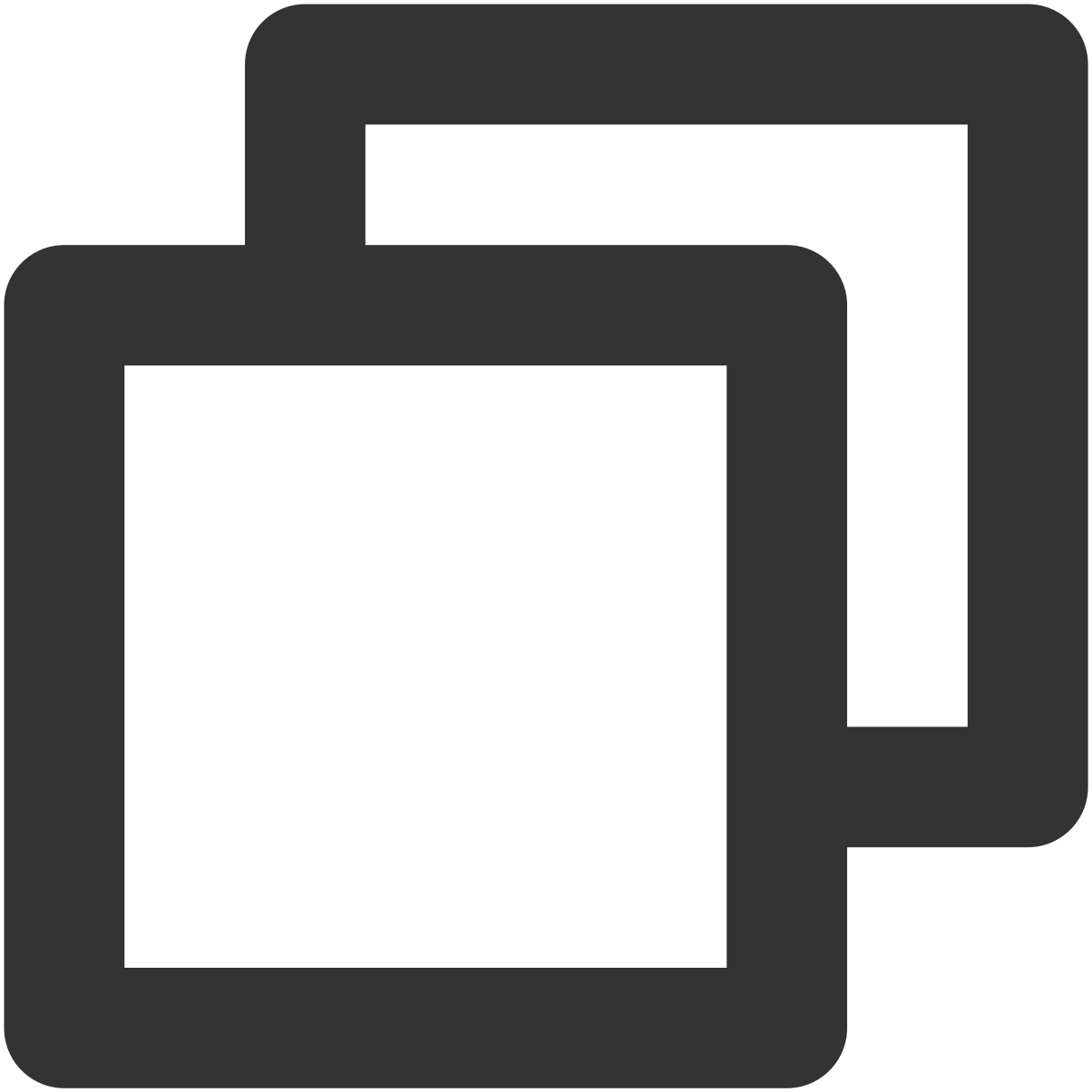


```
WatermarkTextTransform watermarkImageTransform = new WatermarkTextTransform
        //水印内容
        .Builder("水印")
        //设置水印九宫格位置
        .setGravity(CIGravity.CENTER)
        //设置水印字体
        .setFont("tahoma.ttf")
        //设置水印文字字体大小
        .setFontSize(13)
        //设置水印字体颜色
        .setFill("#FF0000")
```

```
            //设置水印文字透明度
            .setDissolve(80)
            //设置水平（横轴）边距
            .setDistanceX(10)
            //设置垂直（纵轴）边距
            .setDistanceY(10)
            //打开平铺水印功能，可将文字水印平铺至整张图片
            .openBatch()
            //旋转水印
            .rotate(45)
            .builder();
transform.watermarkText(watermarkImageTransform);
```

去除图片元信息

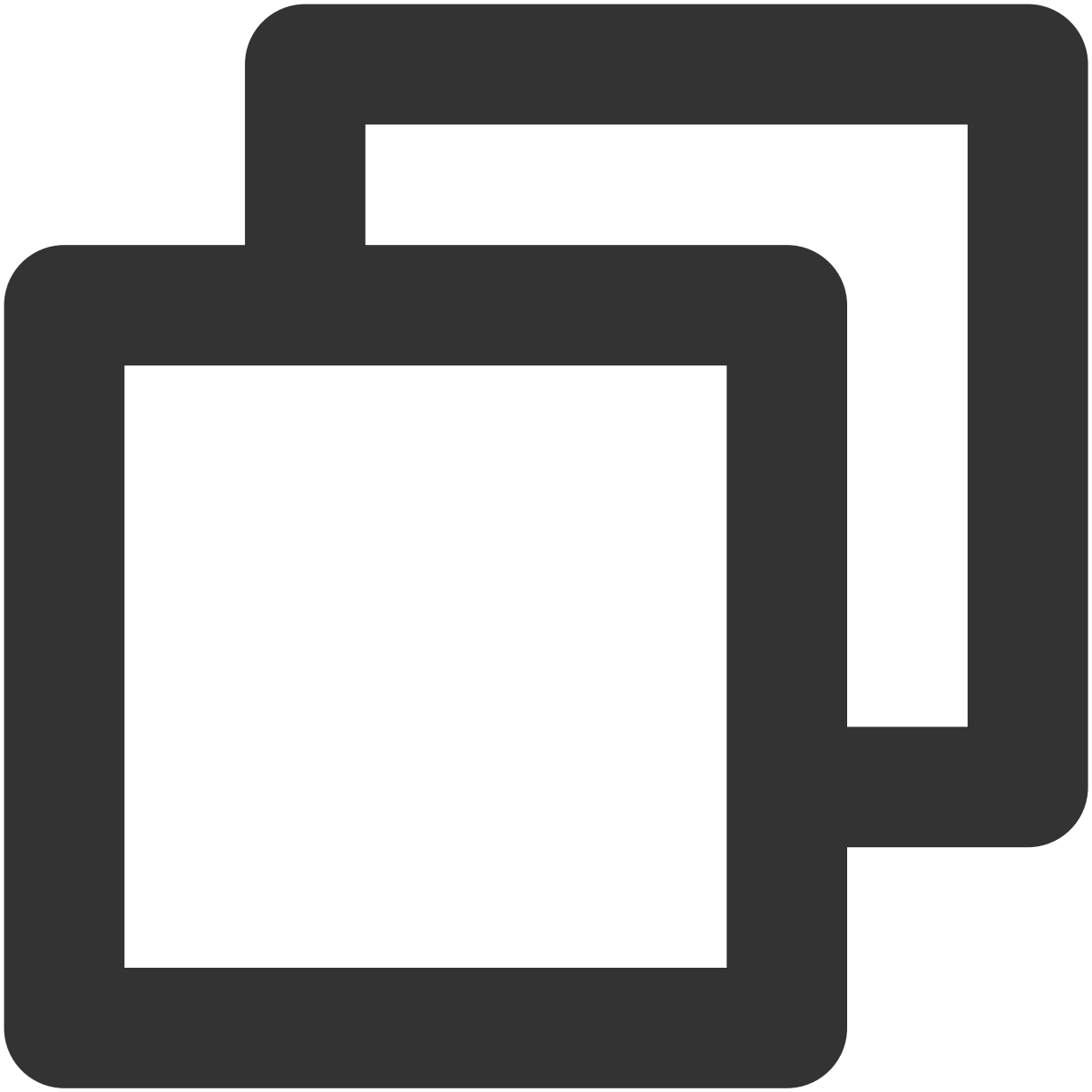
腾讯云数据万象通过 imageMogr2 接口可去除图片元信息，包括 EXIF 信息。API 文档请参见 去除图片元信息接口。



```
transform.strip();
```

组合操作

Transform 支持多操作组合，并保持有序。



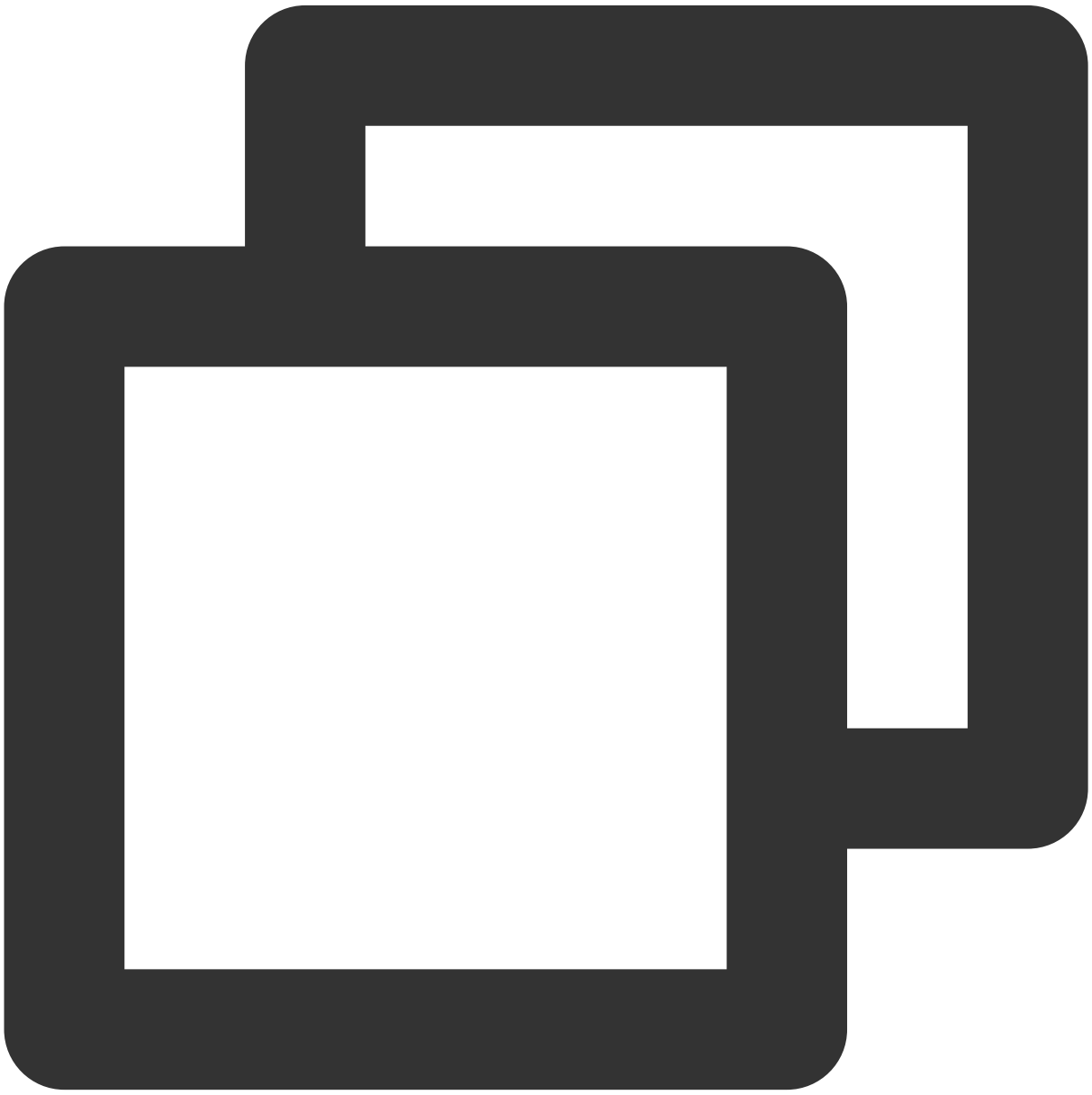
```
transform
        .thumbnailByScale(80)
        .iradius(100)
        .blur(20,20)
        .strip();
```

使用 Glide 加载图片

最近更新时间：2024-03-13 15:17:11

本文主要介绍如何使用 Glide 加载图片。

安装 Glide

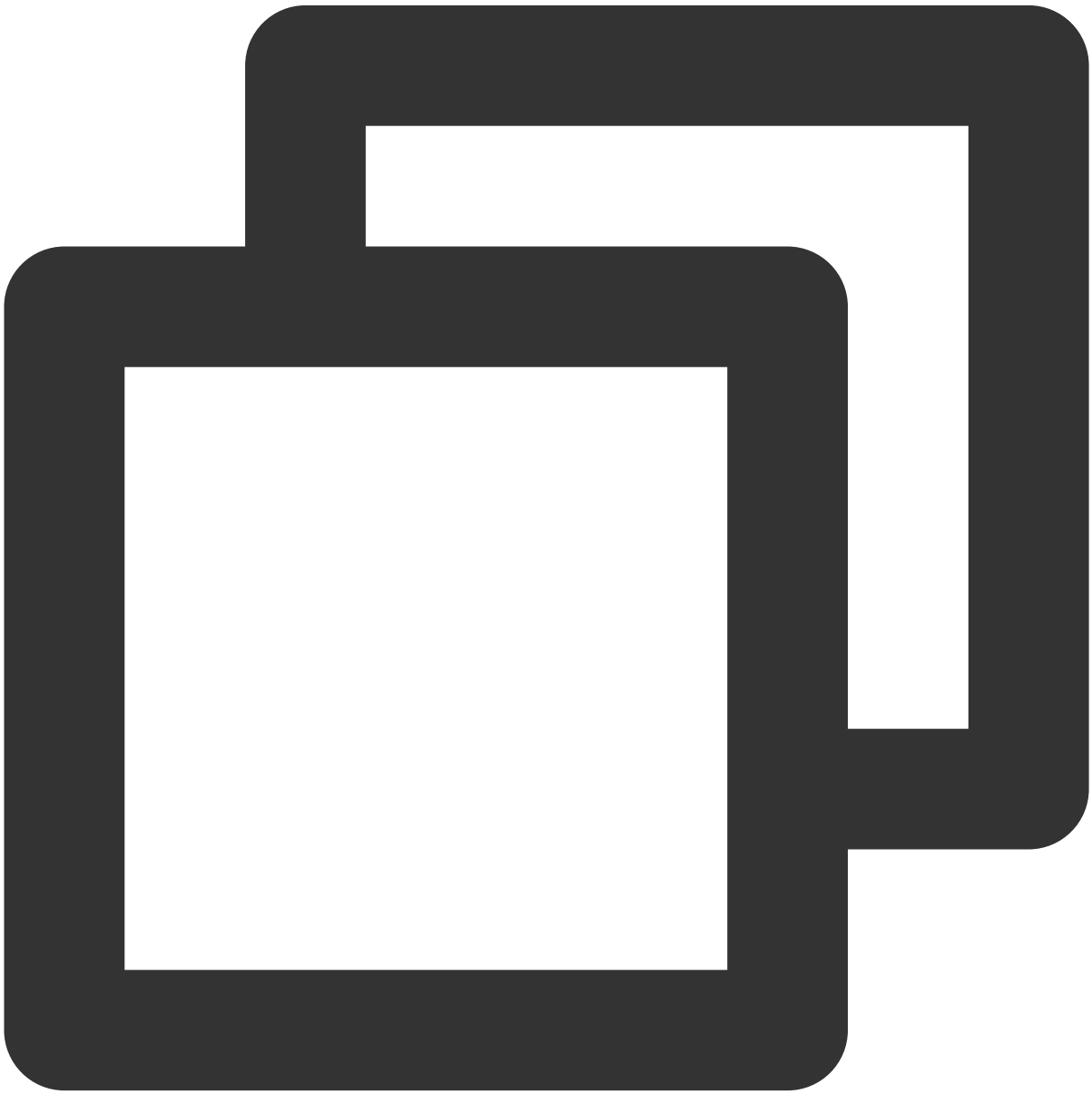


```
implementation 'com.github.bumptech.glide:glide:version'
```

基础图片处理

与 Glide 配合使用数据万象基础图片处理操作（除 TPG 和 AVIF 相关功能外）。

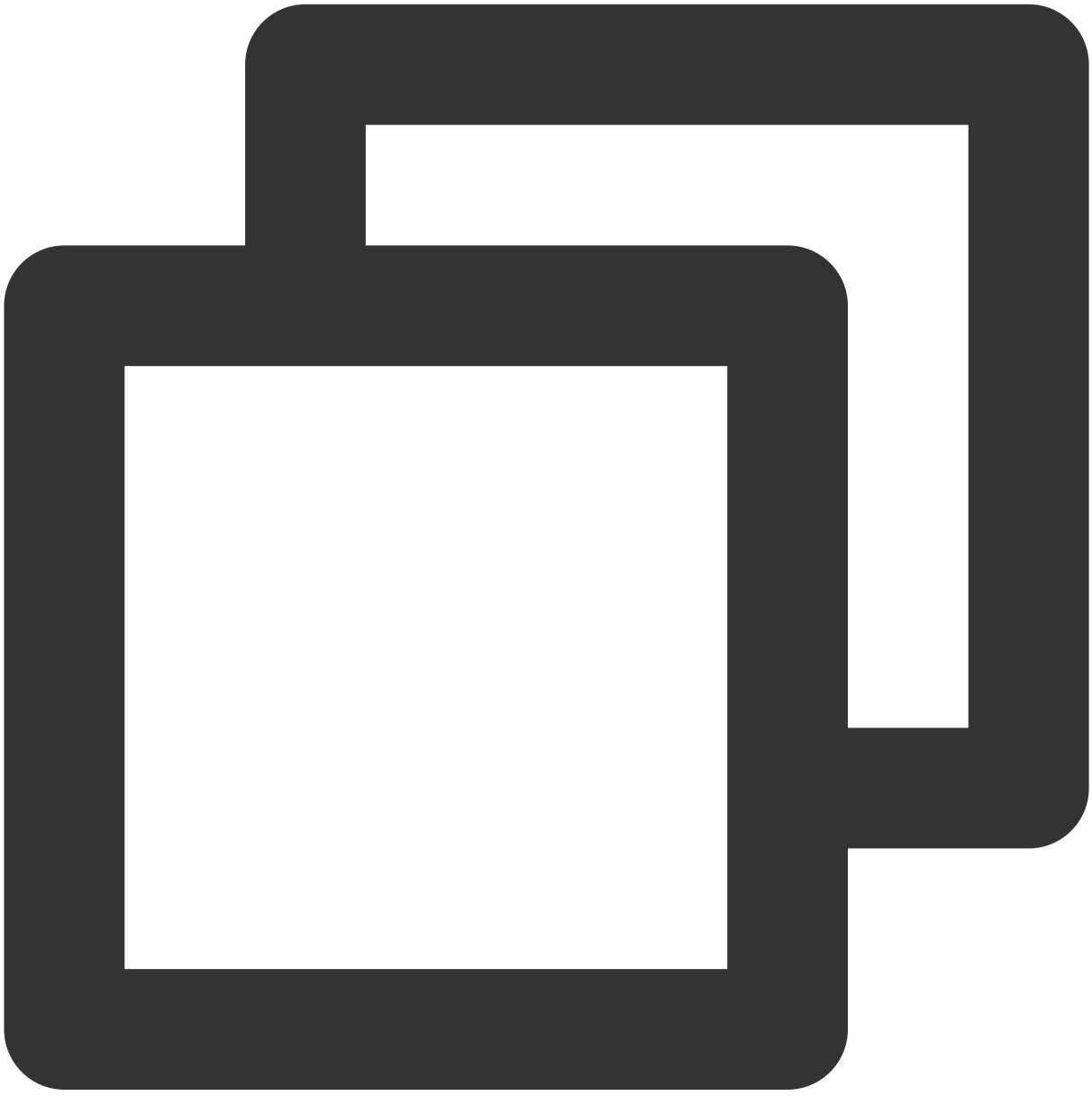
1. 使用 CloudInfinite 和 CITransformation 构建 CIImageLoadRequest。



```
CloudInfinite cloudInfinite = new CloudInfinite();
```

```
CITransformation transform = new CITransformation();
transform.thumbnailByScale(50).iradius(60);
CIImageLoadRequest request = cloudInfinite.requestWithBaseUrlSync(url, transform);
```

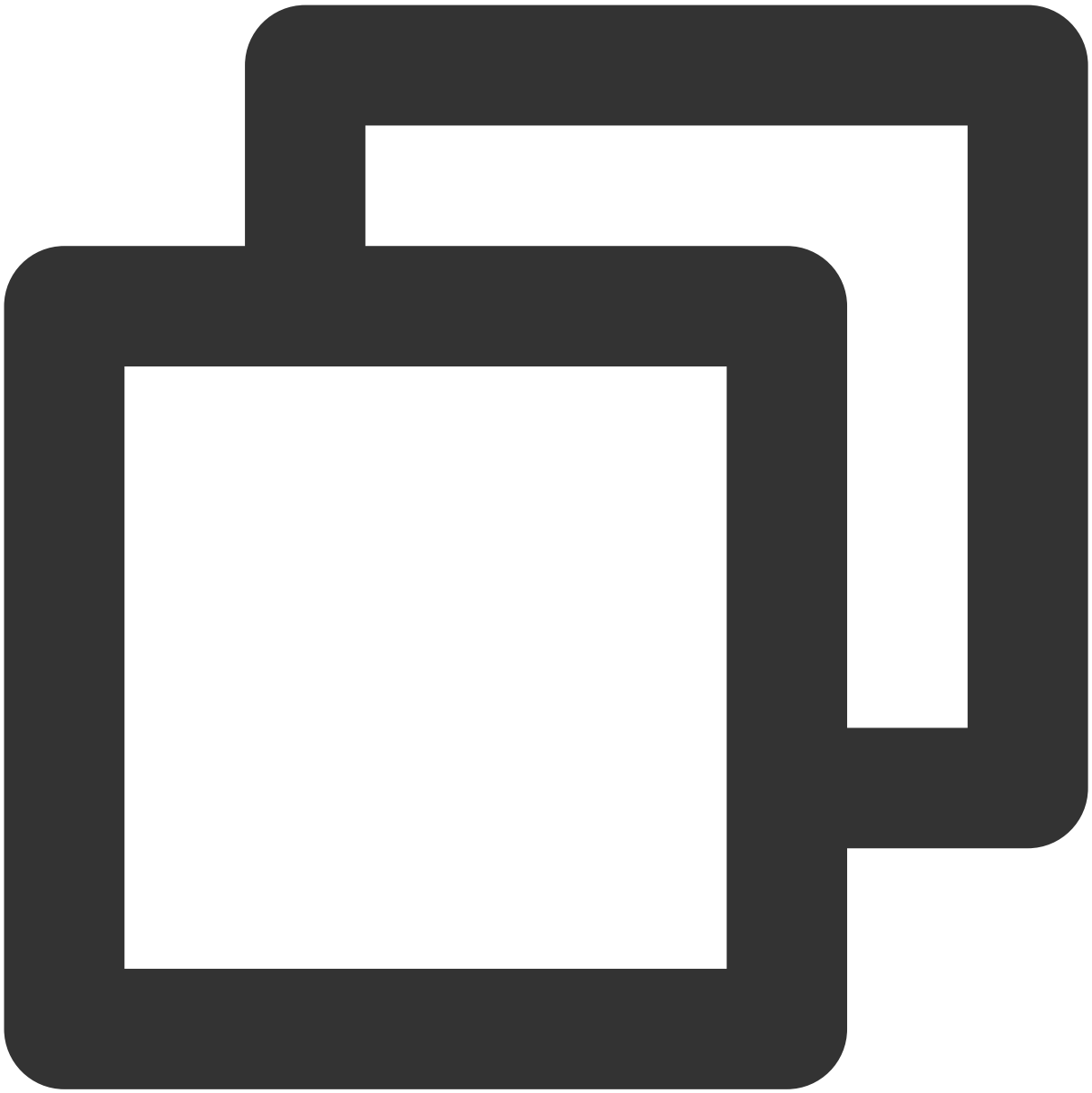
2. 通过得到的 CIImageLoadRequest 获取 URL，使用 Glide 进行加载。



```
Glide.with(activity).load(request.getUrl().toString()).into(imageview);
```

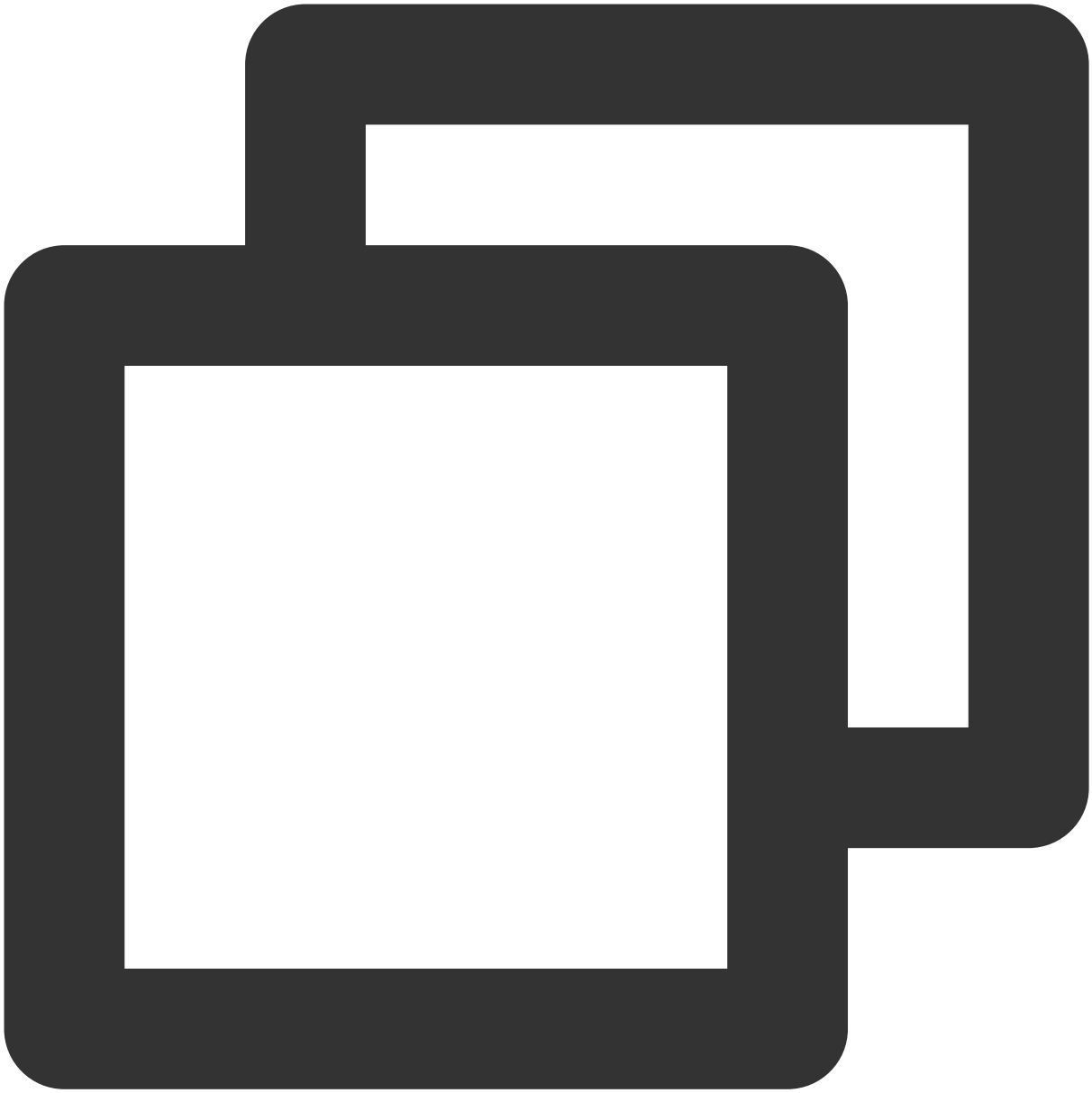
使用数据万象 Glide 功能

安装 cloud-infinite-glide SDK 以及 glide:compiler。



```
implementation 'com.qcloud.cos:cloud-infinite-glide:1.2.1'
annotationProcessor 'com.github.bumptech.glide:compiler:version'
```

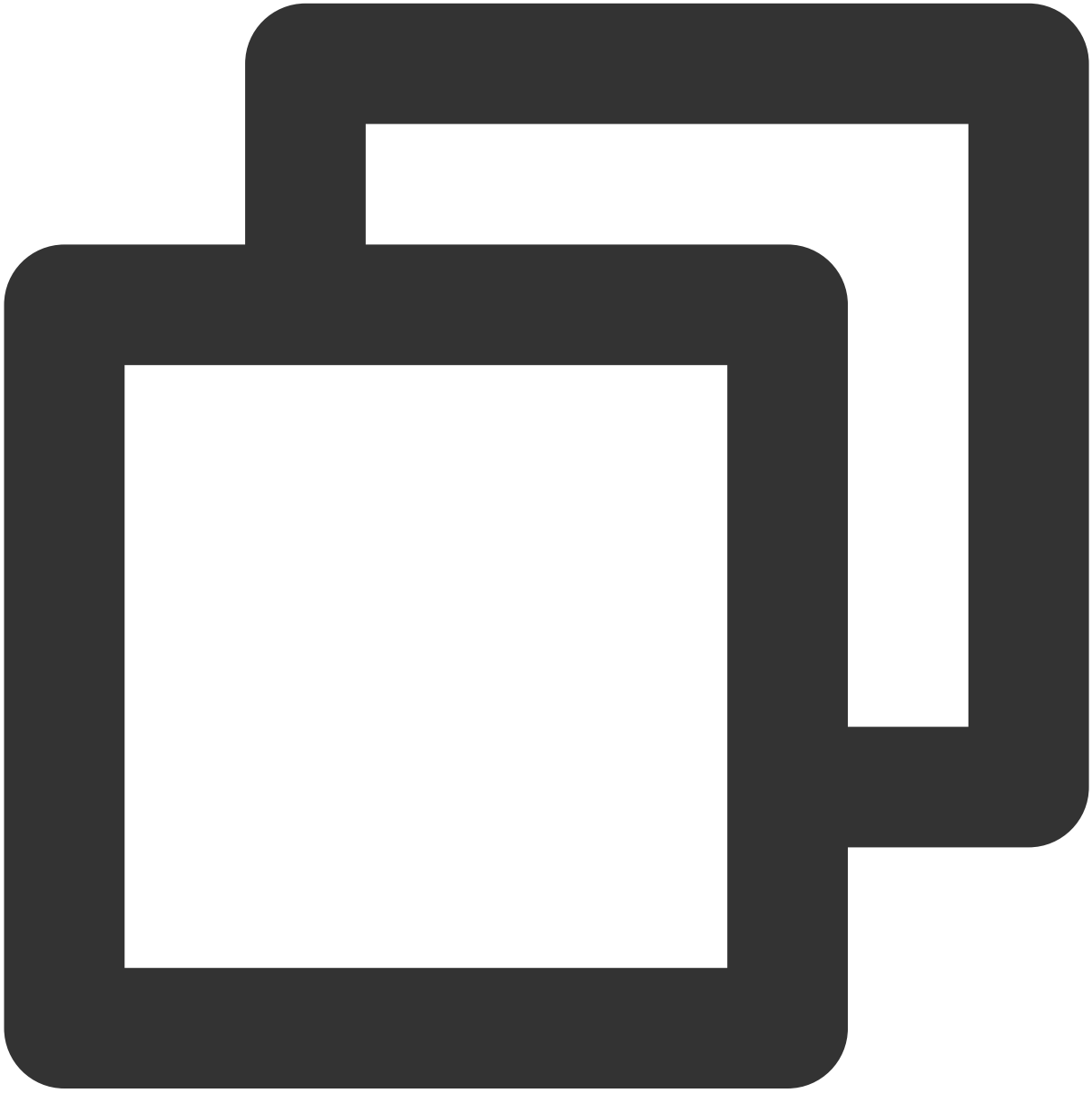
通过 AppGlideModule 注册相关解码器和 loader 实现相应功能。



```
 // 注册自定义 GlideModule
 // 开发者应该创建此类注册 CIImageRequestModelLoader 和 ImageAveDecoder<br>
 // 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,
        //注册支持 CIImageLoadRequest 的 loader
        registry.prepend(CIImageLoadRequest.class, InputStream.class, new CIImageRe

        //注册主色解码器
```

```
        registry.prepend(InputStream.class, Bitmap.class, new ImageAveDecoder(glide
    }
}
```

使用图片主色预加载

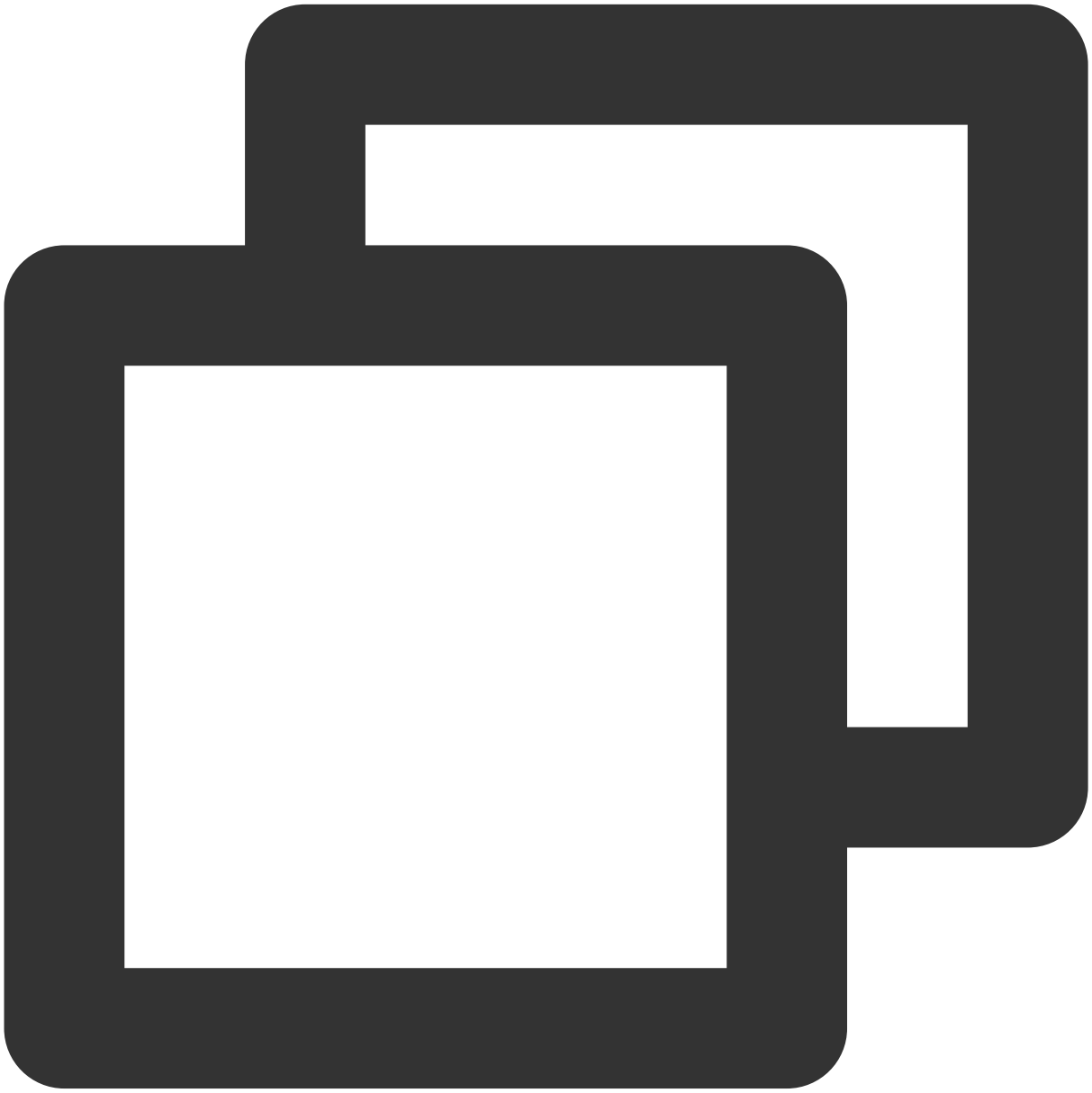


```
//先在 AppGlideModule 中注册主色解码器
registry.prepend(InputStream.class, Bitmap.class, new ImageAveDecoder(glide.getBitm

//使用 glide 的 thumbnail 进行主色预加载
Glide.with(context)
```

```
                .load(imageUrl)
                .thumbnail(CloudInfiniteGlide.getImageAveThumbnail(context, imageUrl
                .into(imageview);
```

直接加载 CIImageLoadRequest

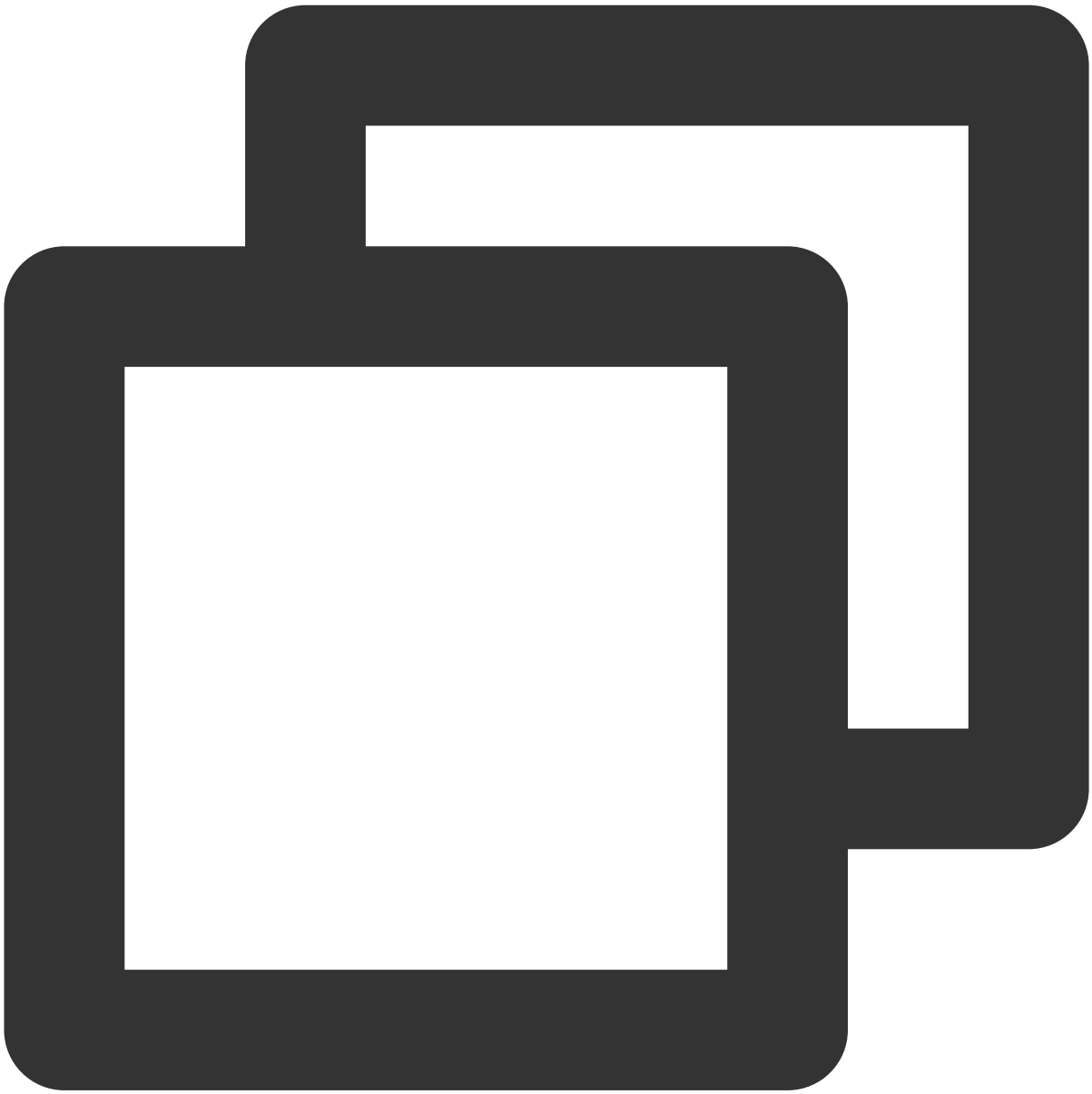
目前仅用于格式转换时，header 传输目标图片格式的情况。



```
registry.prepend(CIImageLoadRequest.class, InputStream.class, new CIImageRequestMod
```

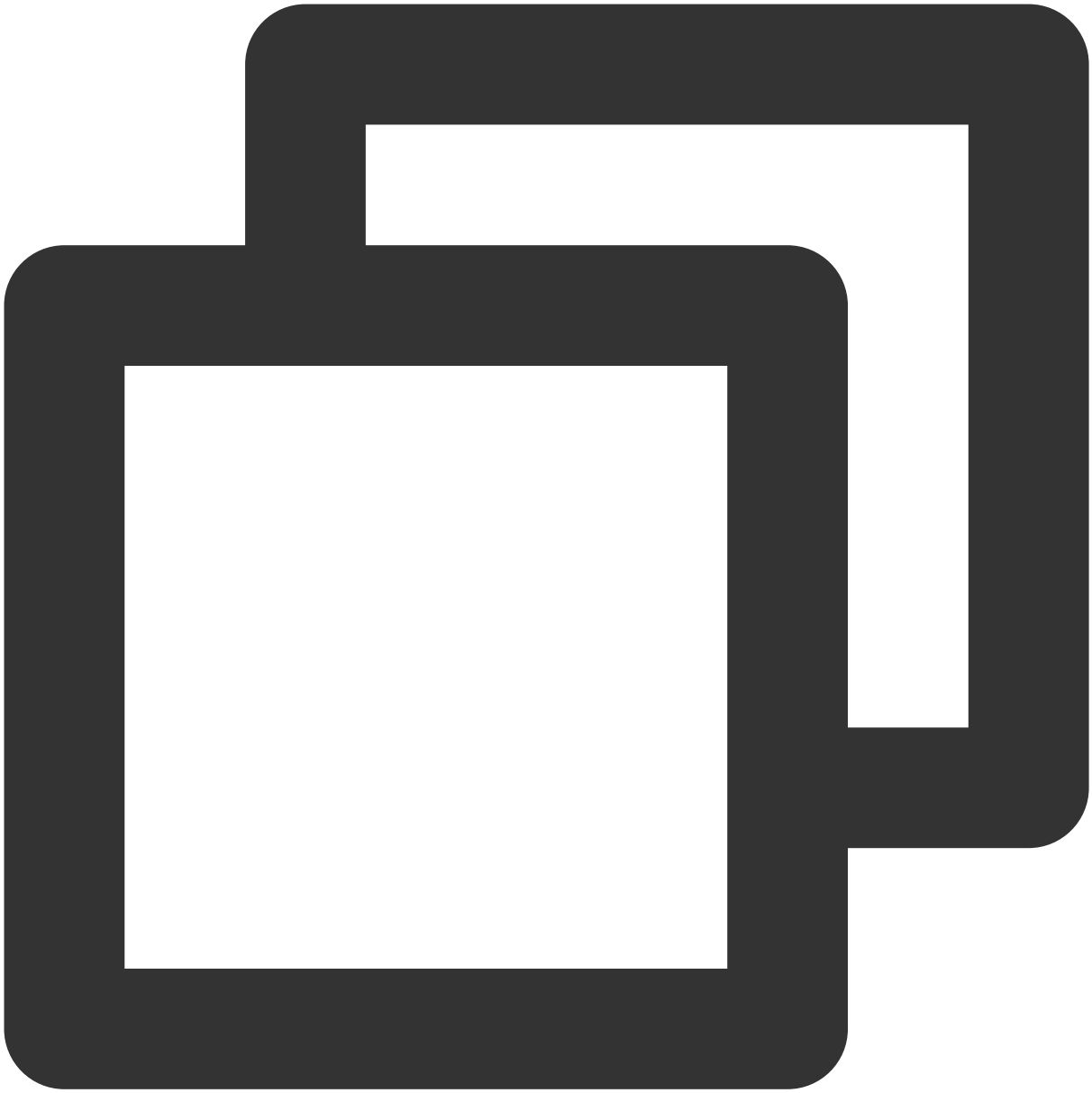
使用数据万象 TPG 功能

安装 TPG SDK 以及 glide:compiler。



```
implementation 'com.qcloud.cos:tpg:1.3.2'
annotationProcessor 'com.github.bumptech.glide:compiler:version'
```

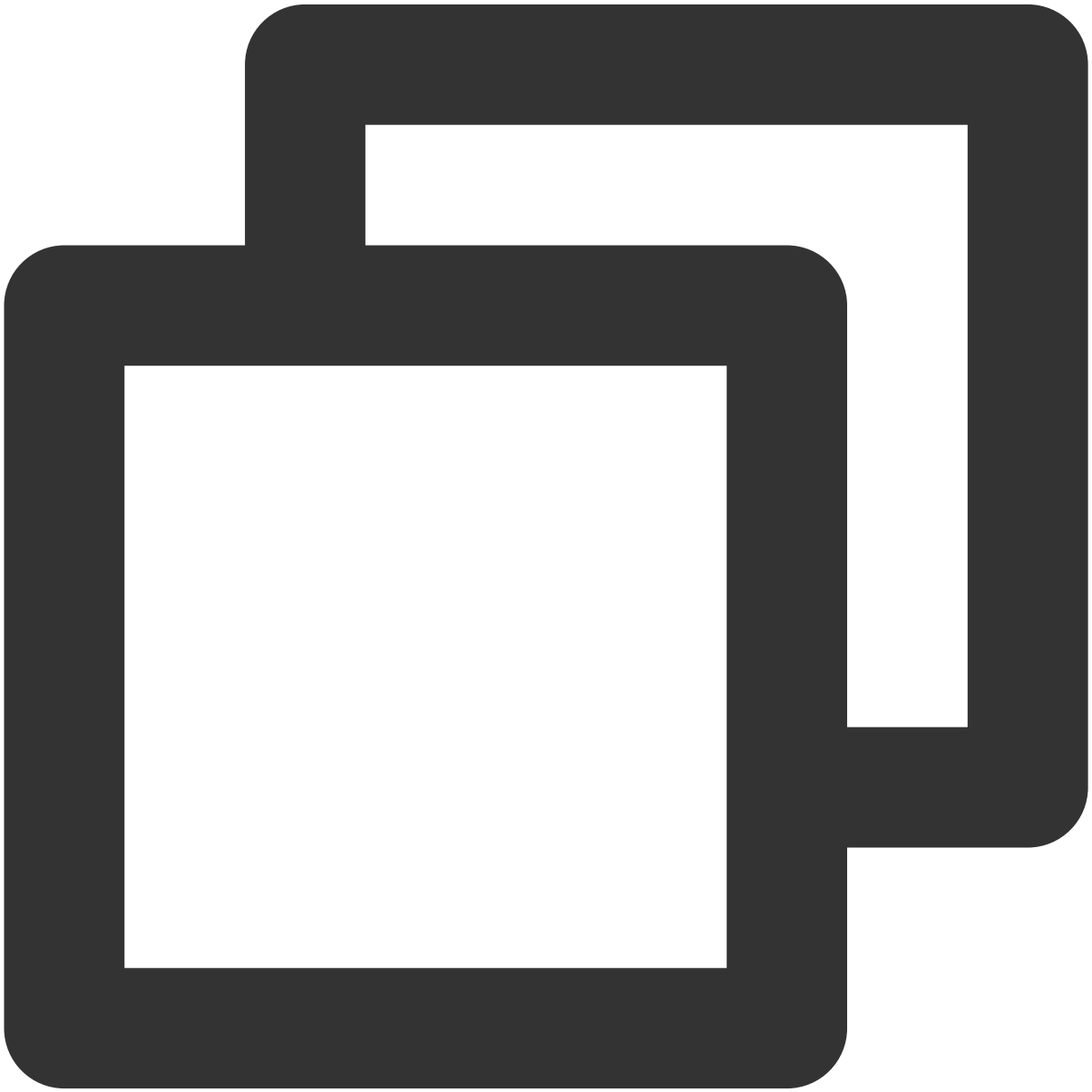
通过 AppGlideModule 注册相关解码器实现相应功能。



```
 // 注册自定义 GlideModule
 // 开发者应该创建此类注册TpgDecoder、ByteBufferTpgGifDecoder<br>
 // 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,
        /*-------------------解码器 开始--------------------------*/
        //注册 TPG 静态图片解码器
        registry.prepend(InputStream.class, Bitmap.class, new TpgDecoder(glide.getB
        //注册 TPG 动图解码器
```

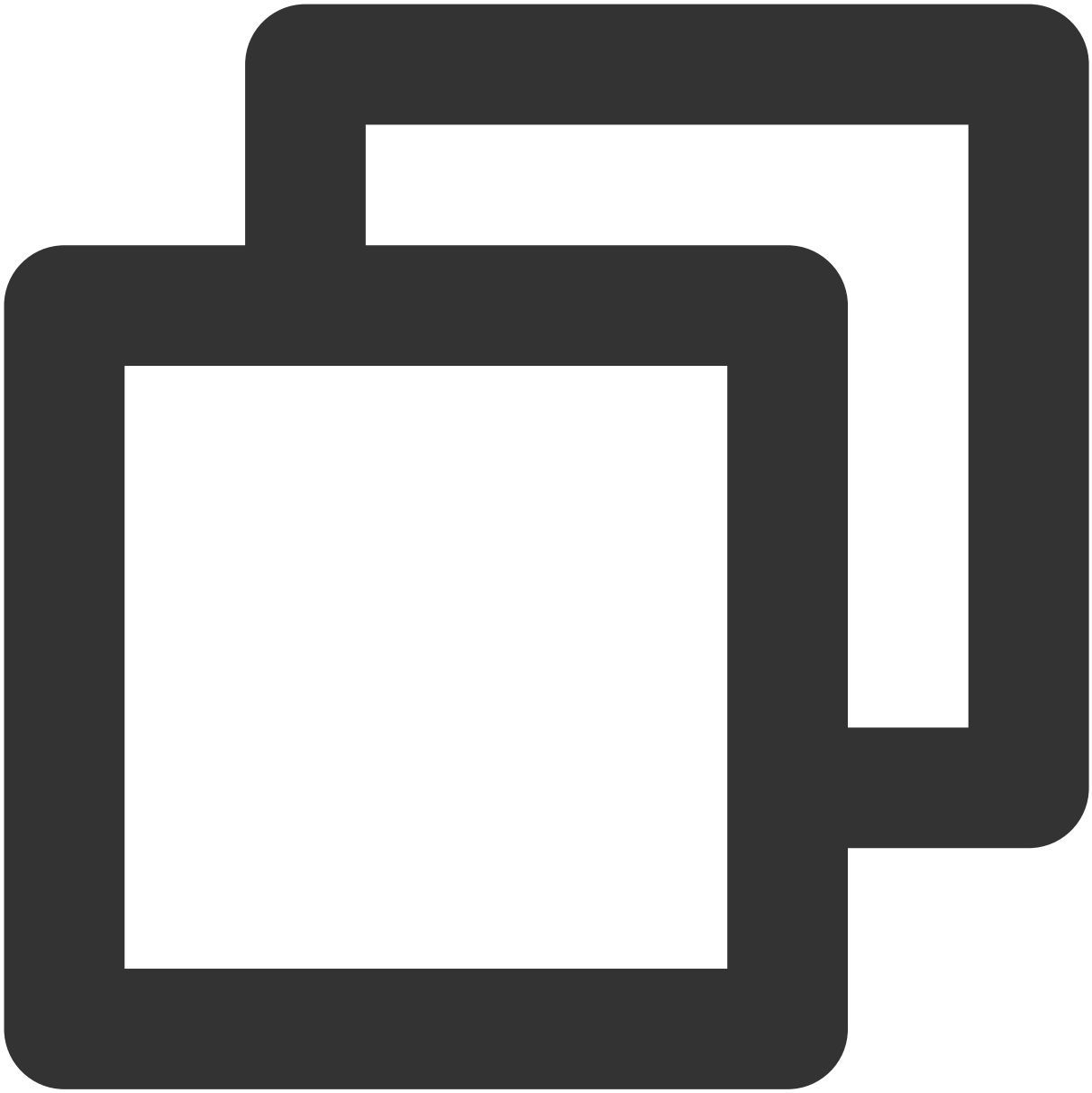
```
        ByteBufferTpgGifDecoder byteBufferTpgGifDecoder = new ByteBufferTpgGifDecod
        registry.prepend(InputStream.class, GifDrawable.class, new StreamTpgGifDeco
        registry.prepend(ByteBuffer.class, GifDrawable.class, byteBufferTpgGifDecod
        /*------------------解码器 结束-------------------------*/
    }
}
```

加载静态 TPG 图片



```
registry.prepend(InputStream.class, Bitmap.class, new TpgDecoder(glide.getBitmapPoo
```

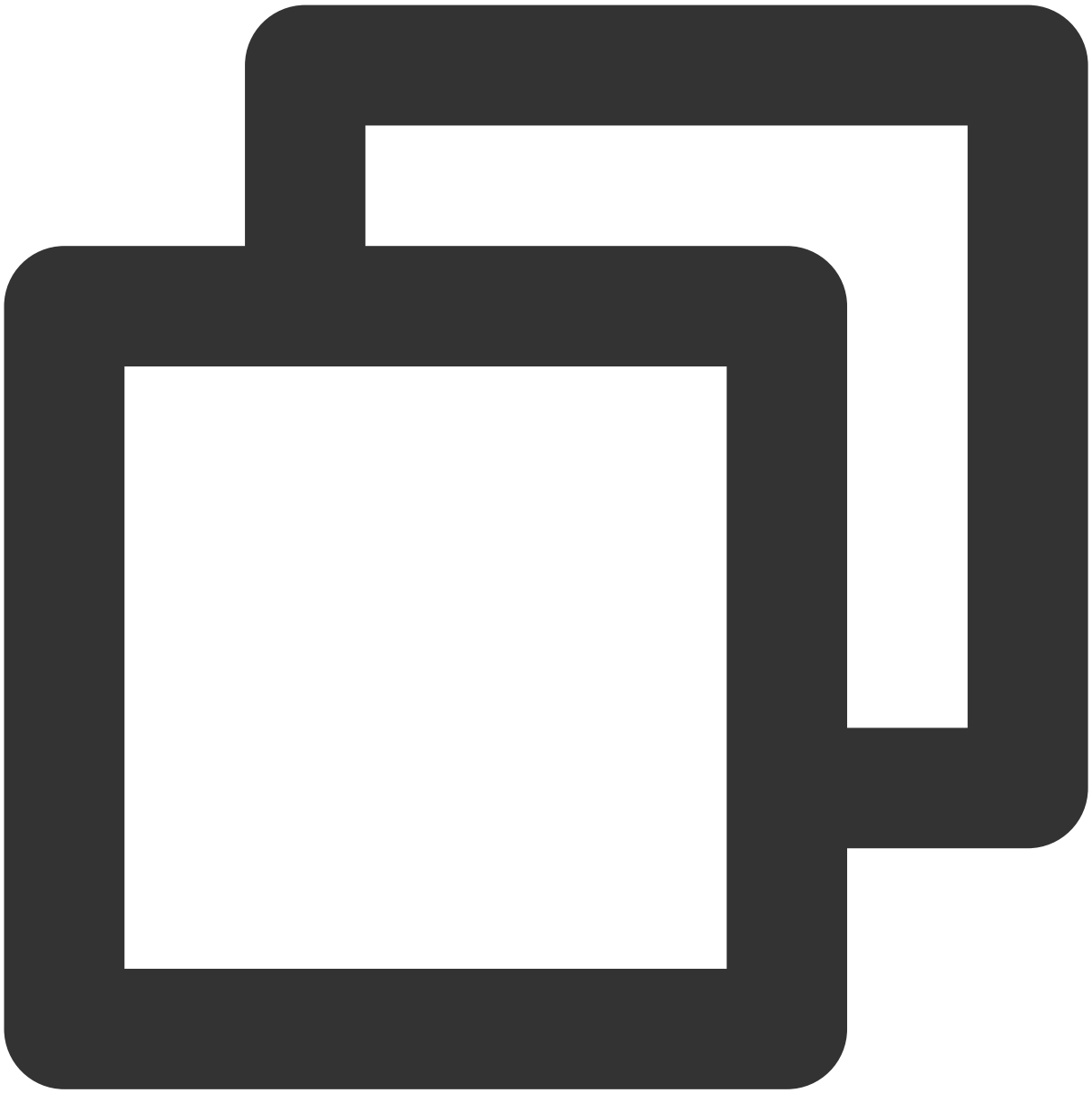
加载动图类型 TPG 图片



```
ByteBufferTpgGifDecoder byteBufferTpgGifDecoder = new ByteBufferTpgGifDecoder(conte
registry.prepend(InputStream.class, GifDrawable.class, new StreamTpgGifDecoder(byte
registry.prepend(ByteBuffer.class, GifDrawable.class, byteBufferTpgGifDecoder);
```

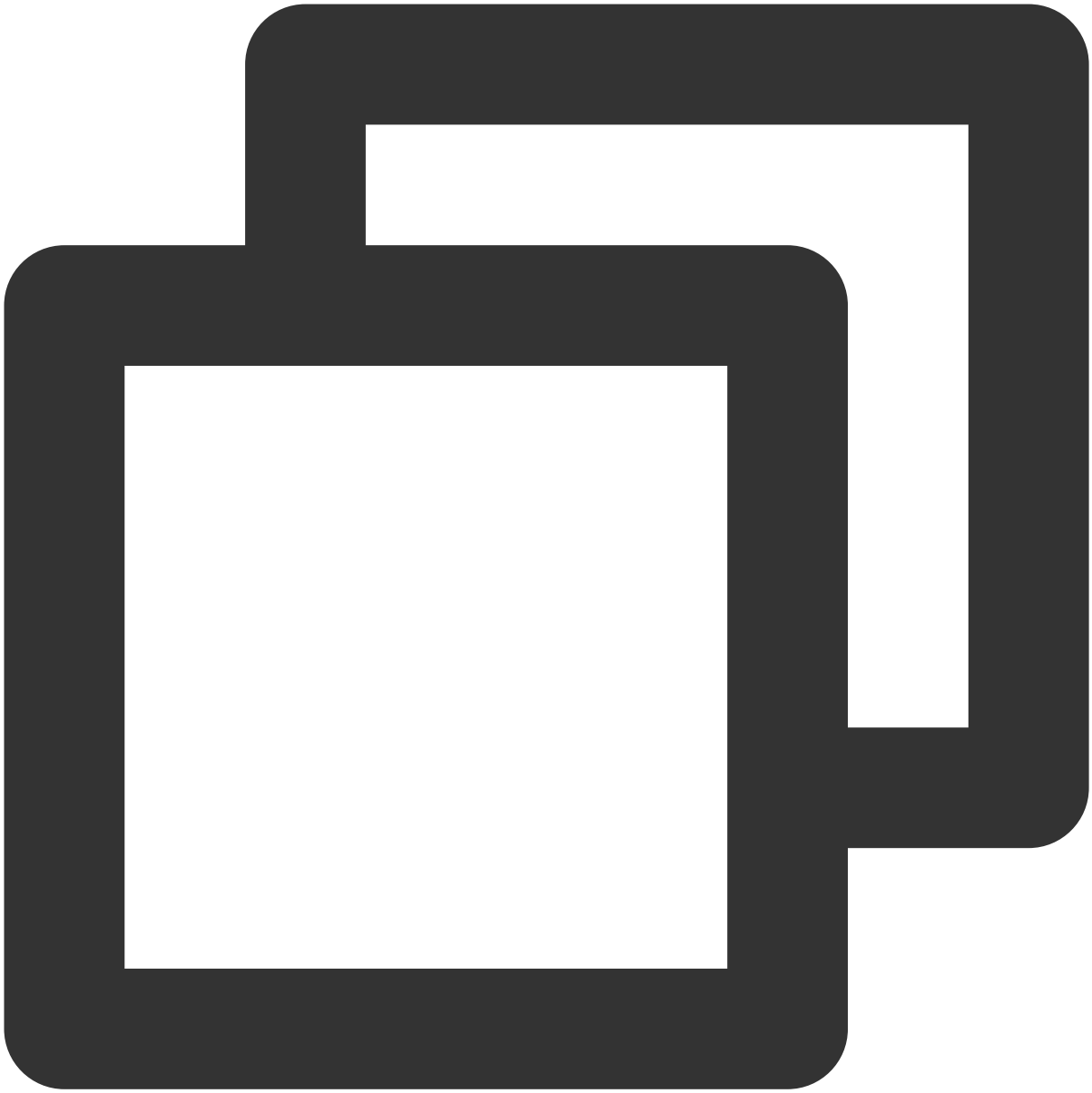
使用数据万象 AVIF 功能

安装 AVIF SDK 以及 glide:compiler。



```
implementation 'com.qcloud.cos:avif:1.0.0'
annotationProcessor 'com.github.bumptech.glide:compiler:version'
```

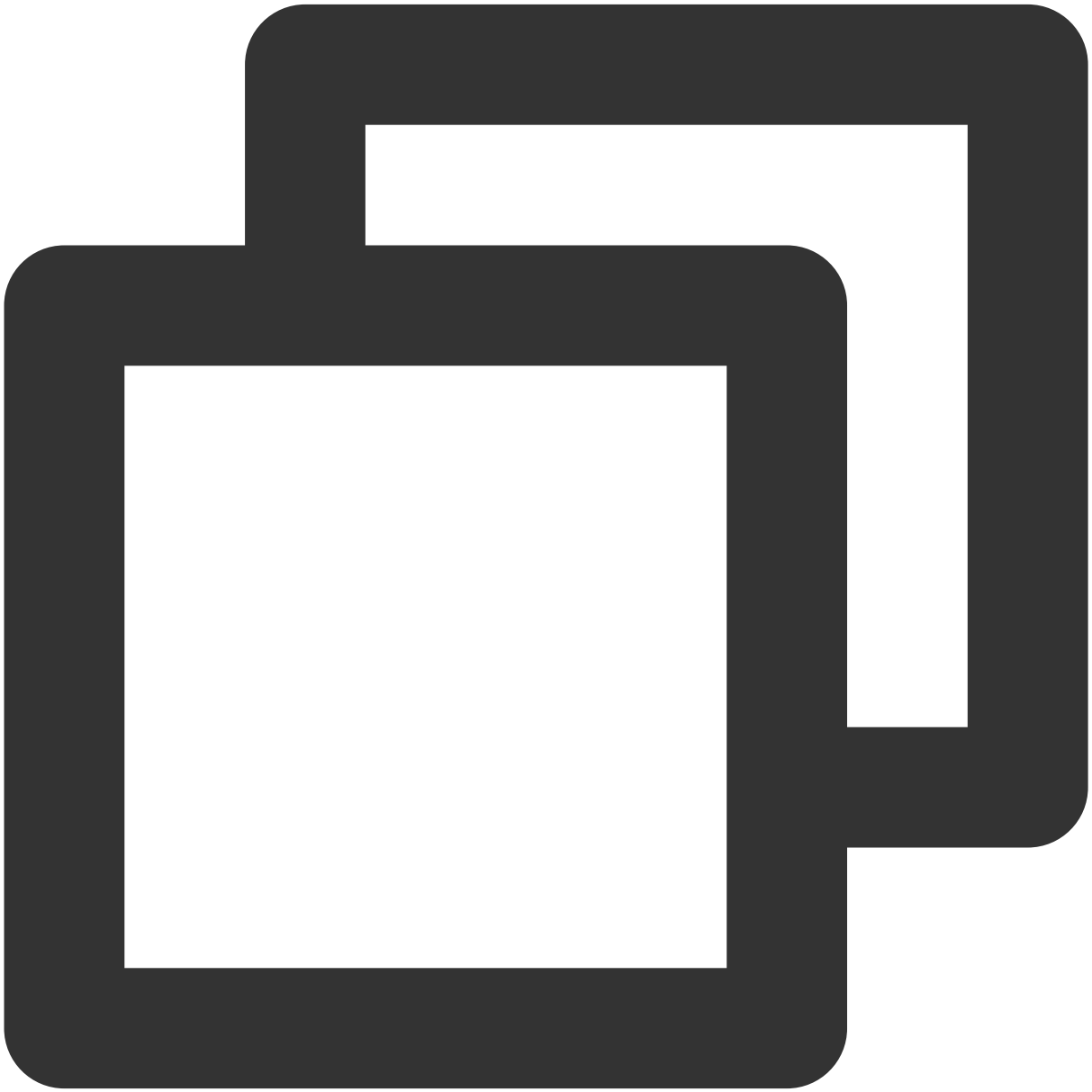
通过 AppGlideModule 注册相关解码器实现相应功能。



```
 // 注册自定义 GlideModule
 // 开发者应该创建此类注册相关解码器<br>
 // 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,
        /*-------------------解码器 开始--------------------------*/
        //注册 AVIF 静态图片解码器
        registry.prepend(Registry.BUCKET_BITMAP, InputStream.class, Bitmap.class, n
        registry.prepend(Registry.BUCKET_BITMAP, ByteBuffer.class, Bitmap.class, ne
```

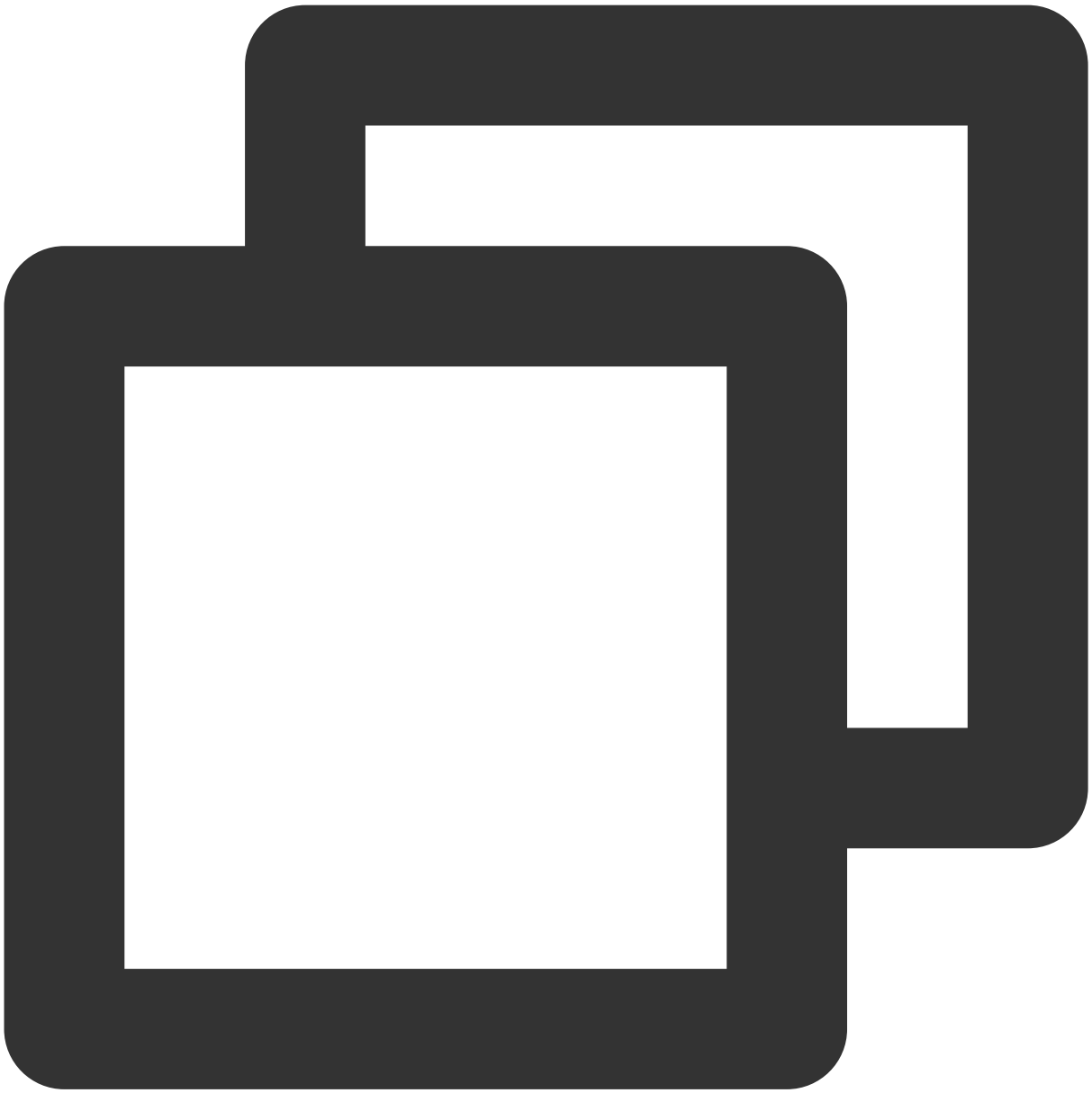
```
        //注册 AVIF 动图解码器
        registry.prepend(InputStream.class, AvifSequenceDrawable.class, new StreamA
        registry.prepend(ByteBuffer.class, AvifSequenceDrawable.class, new ByteBuff
        /*------------------解码器 结束-------------------------*/
    }
}
```

加载静态 AVIF 图片



```
registry.prepend(Registry.BUCKET_BITMAP, InputStream.class, Bitmap.class, new Strea
registry.prepend(Registry.BUCKET_BITMAP, ByteBuffer.class, Bitmap.class, new ByteBu
```

加载动图类型 AVIF 图片



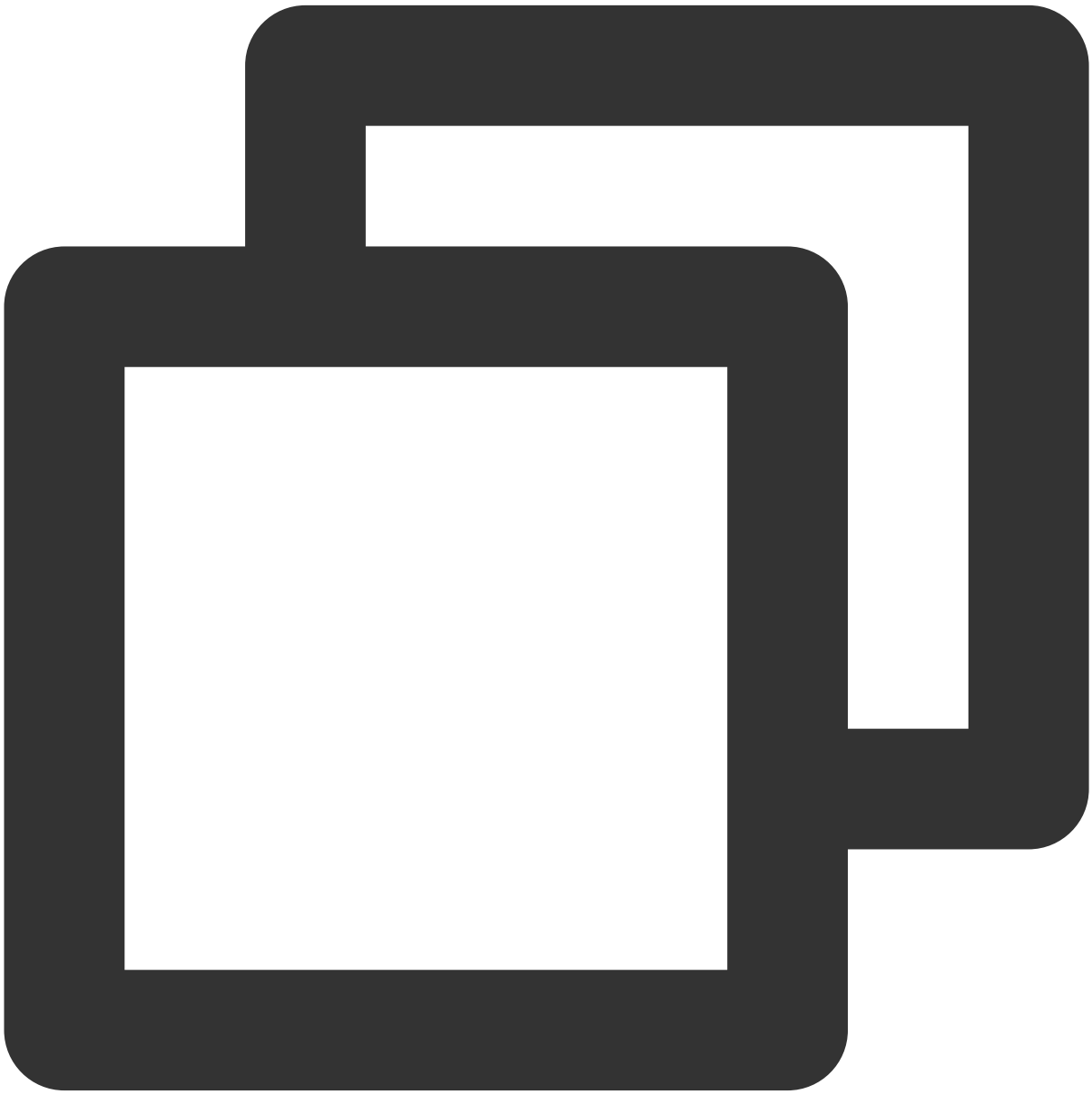
```
registry.prepend(InputStream.class, AvifSequenceDrawable.class, new StreamAvifSeque
registry.prepend(ByteBuffer.class, AvifSequenceDrawable.class, new ByteBufferAvifSe
```

加载 TPG 图片

最近更新时间：2024-03-13 15:17:15

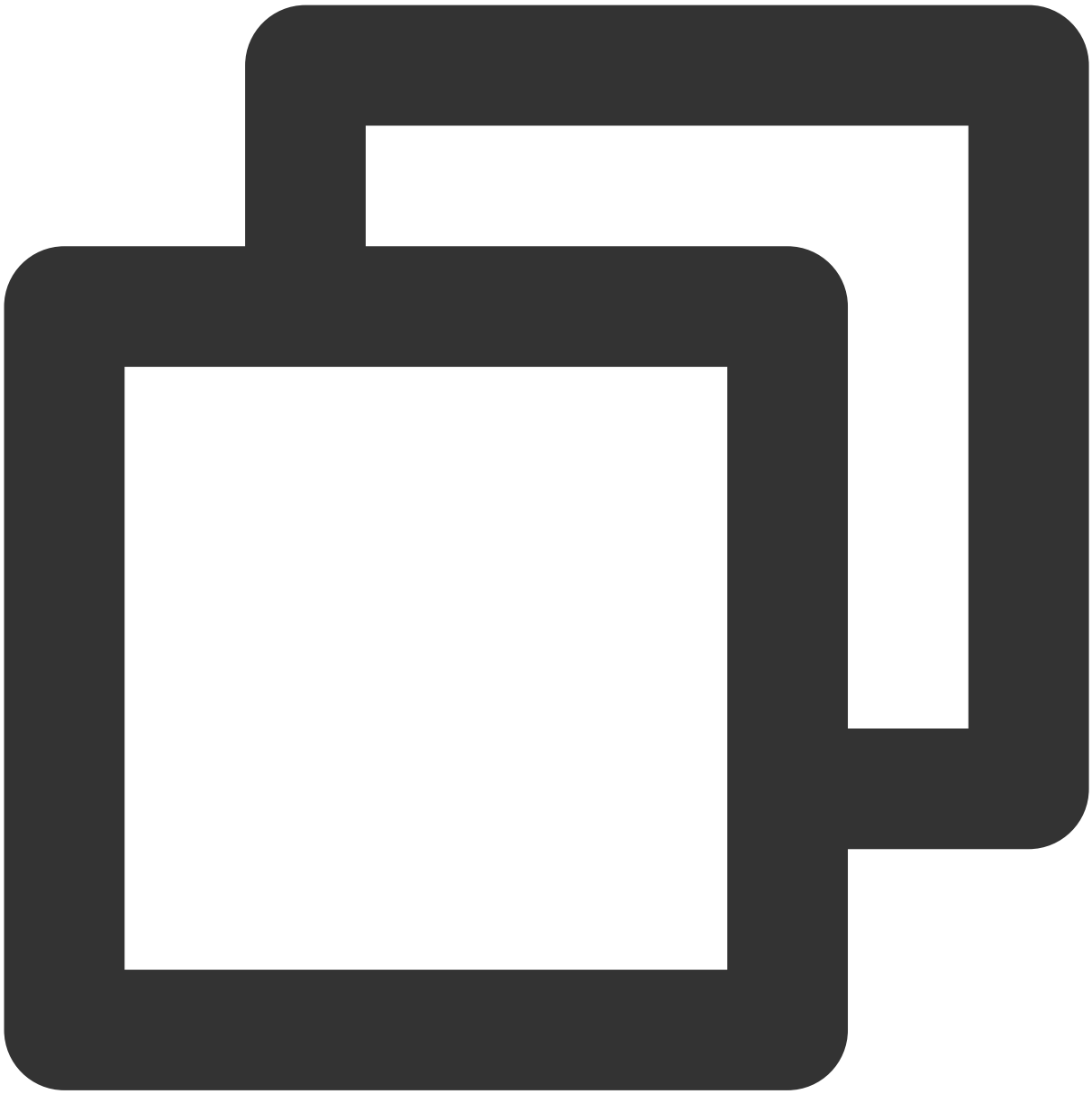
本文提供加载网络以及加载本地两种方式加载 TPG 图片。

安装 TPG SDK



```
implementation 'com.qcloud.cos:tpg:1.3.2'
```

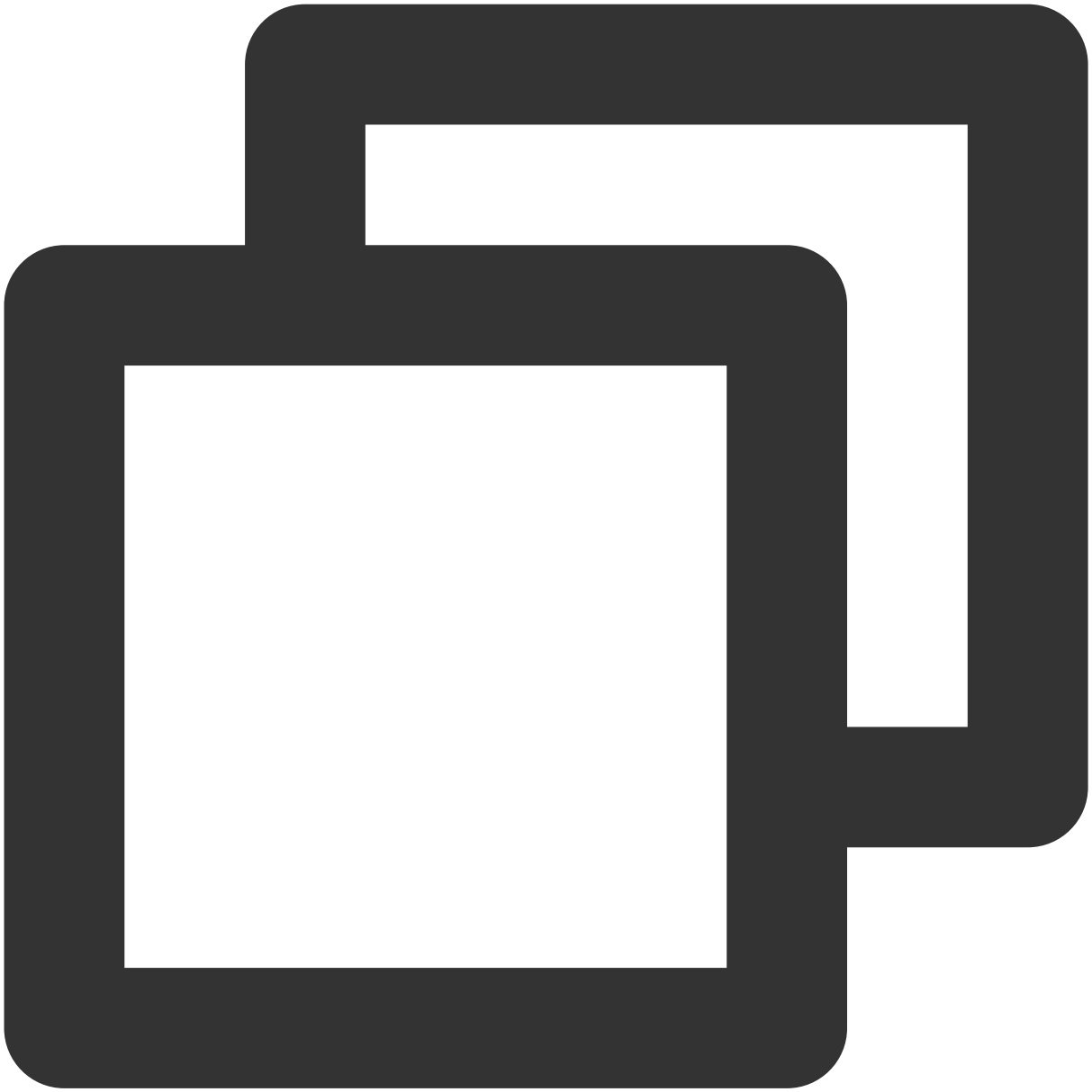
安装时会自动包含 SO 库，建议在 Module 的 build.gradle 文件中使用 NDK 的“abiFilter”配置，设置支持的 SO 库架构。



```
defaultConfig {
    ndk {
        // 设置支持的 SO 库架构
        abiFilters 'armeabi' //, 'x86', 'armeabi-v7a', 'x86_64', 'arm64-v8a'
    }
}
```

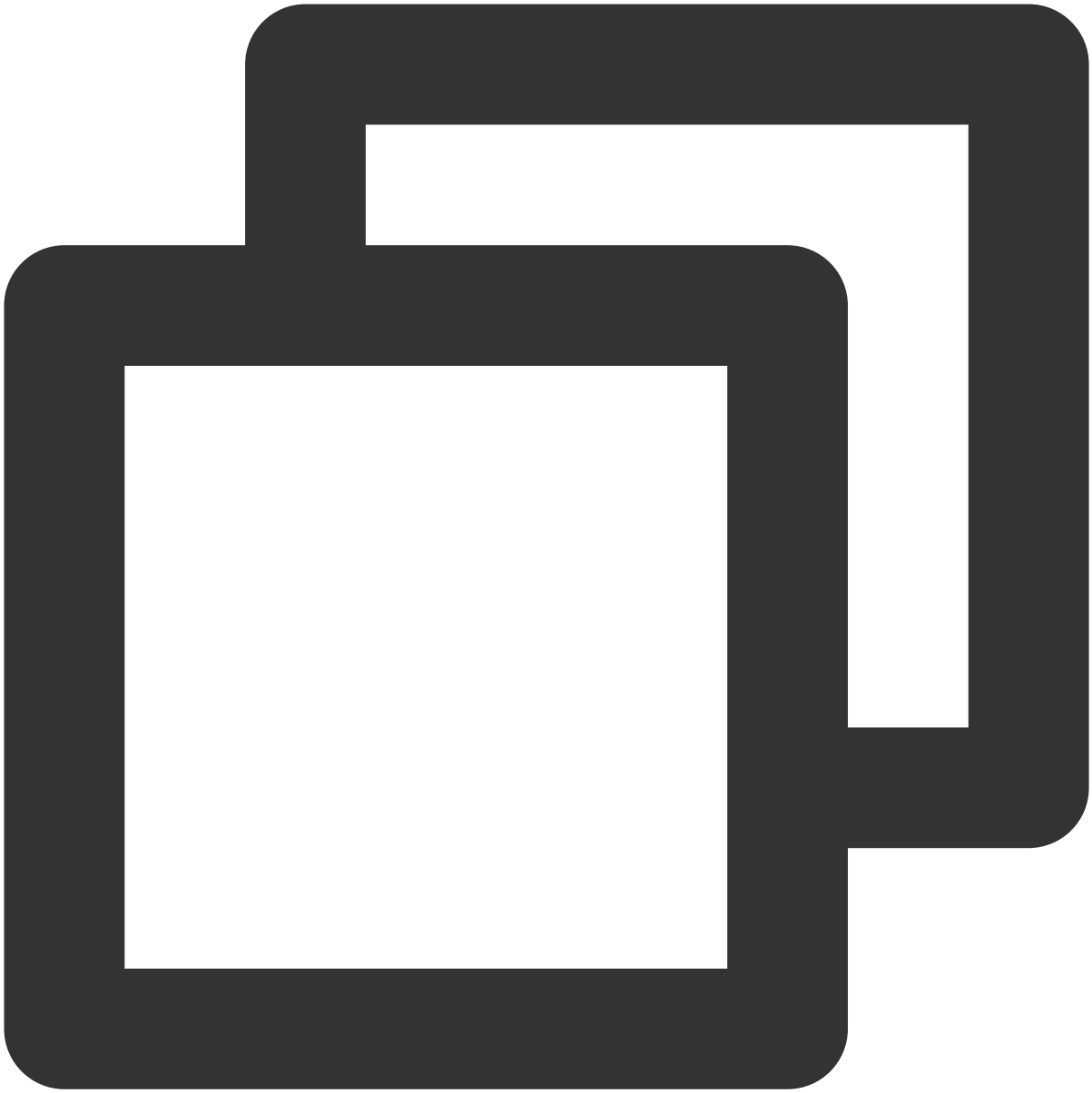
方式一：加载网络 TPG 图片

1. 集成 cloud-infinite SDK。



```
implementation 'com.qcloud.cos:cloud-infinite:1.2.1'
```

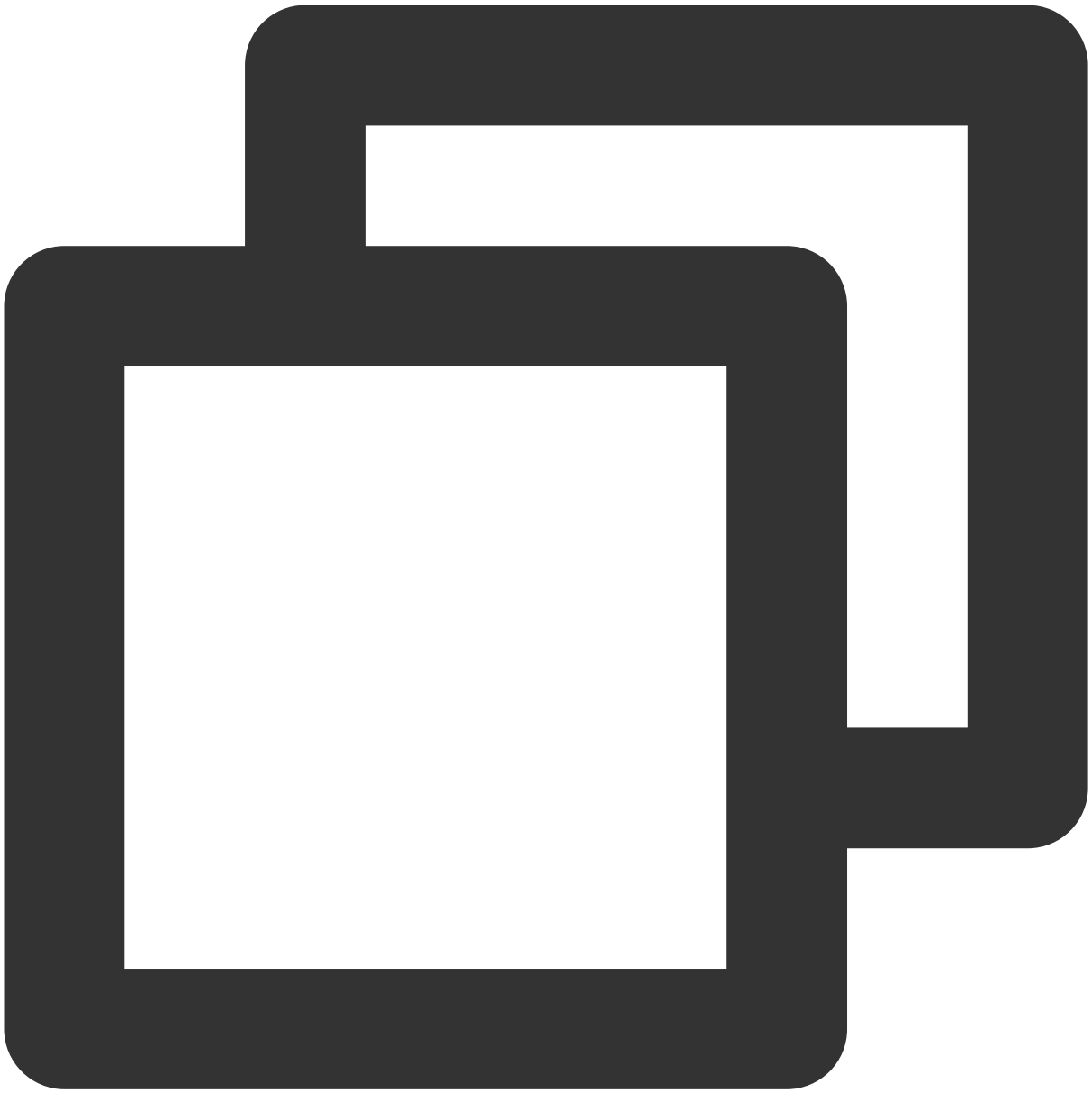
2. 在 cloud-infinite SDK 中构建出请求 TPG 格式图片的链接，然后与 Glide 加载图片 配合使用，加载网络 TPG 图片。



```
// 实例化 CloudInfinite，用来构建请求图片请求连接；
CloudInfinite cloudInfinite = new CloudInfinite();
// 根据用户所选万象基础功能 options 进行 Transformation；
CITransformation transform = new CITransformation();
transform.format(CIImageFormat.TPG, CIImageLoadOptions.LoadTypeUrlFooter);
// 构建图片 CIImageLoadRequest
CIImageLoadRequest request = cloudInfinite.requestWithBaseUrlSync(url, transform);
```

方式二：加载本地 TPG 图片

应用内置的资源，例如 assets、drawable、raw 等，使用 TPG 格式可以有利于减小安装包大小。



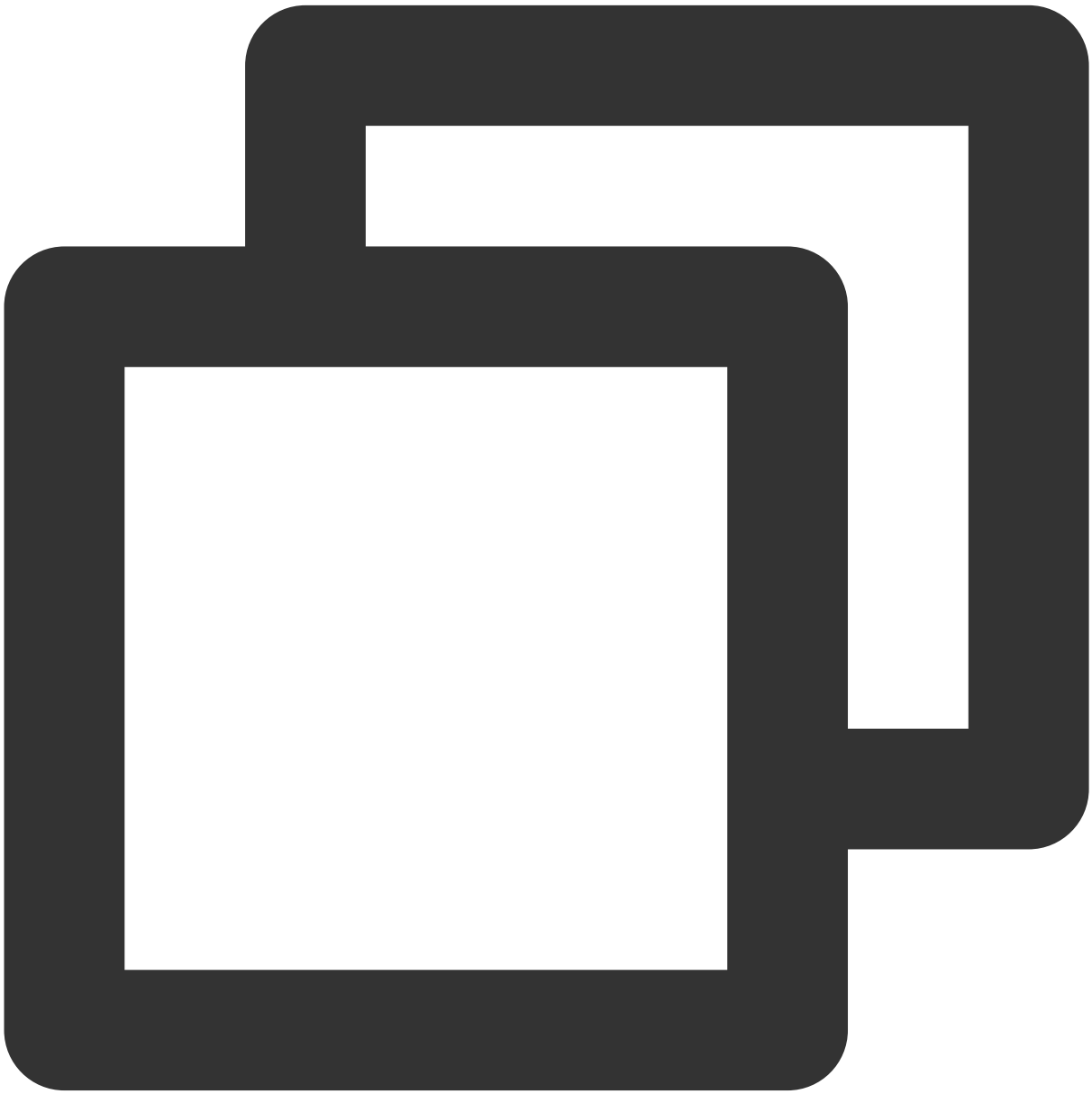
```
//加载 Assets 中的 TPG 图片
TpgImageLoader.displayWithAssets(imageview, assetsName);
//加载 Resource 中的 TPG 图片
TpgImageLoader.displayWithResource(imageview, R.drawable.tpg);
//加载本地文件中的 TPG 图片
TpgImageLoader.displayWithFileUri(imageview, fileUri);
```

加载 AVIF 图片

最近更新时间：2024-03-13 15:17:16

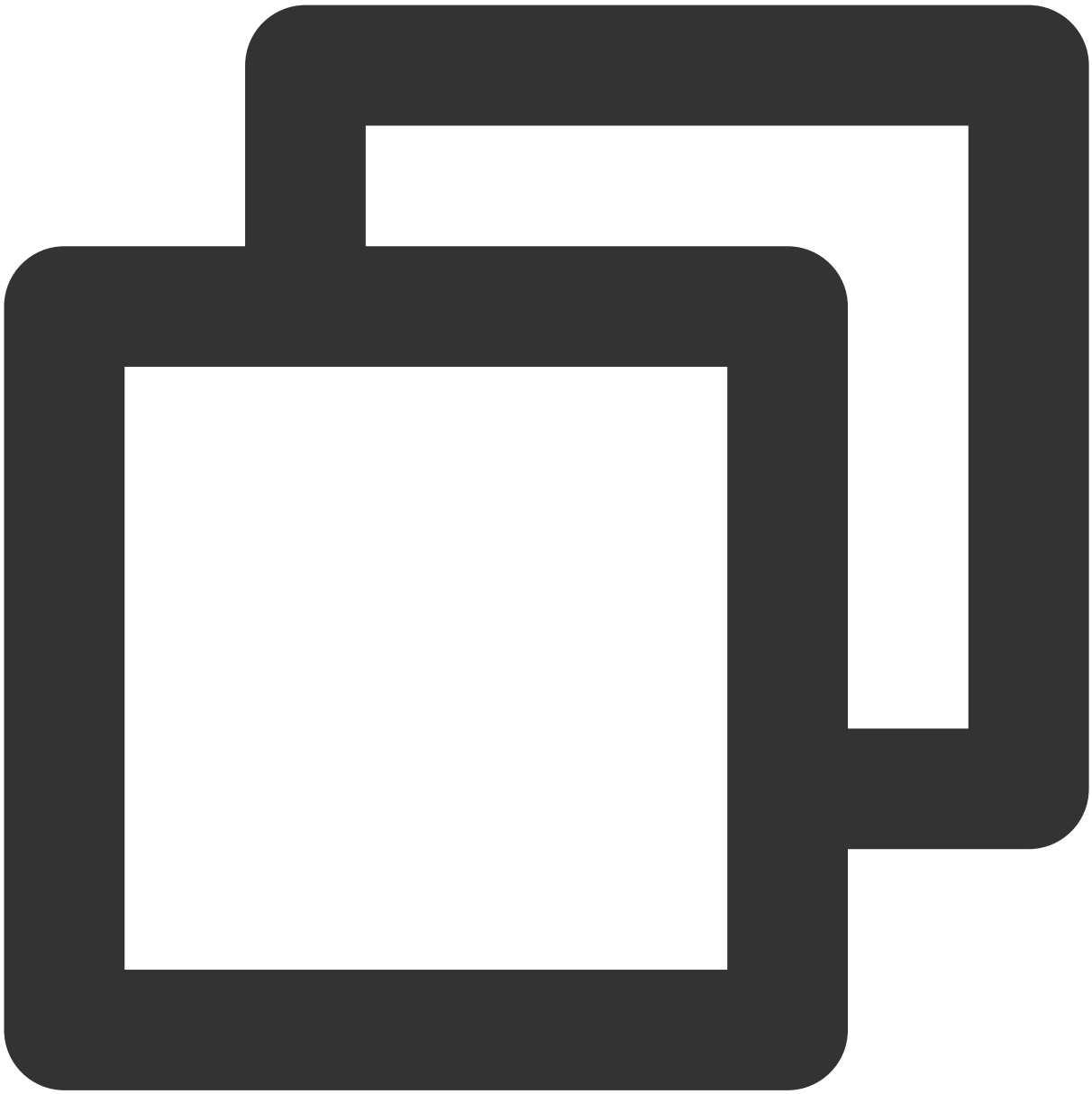
本文提供加载网络以及加载本地两种方式加载 AVIF 图片。

安装 AVIF SDK



```
implementation 'com.qcloud.cos:avif:1.0.0'
```

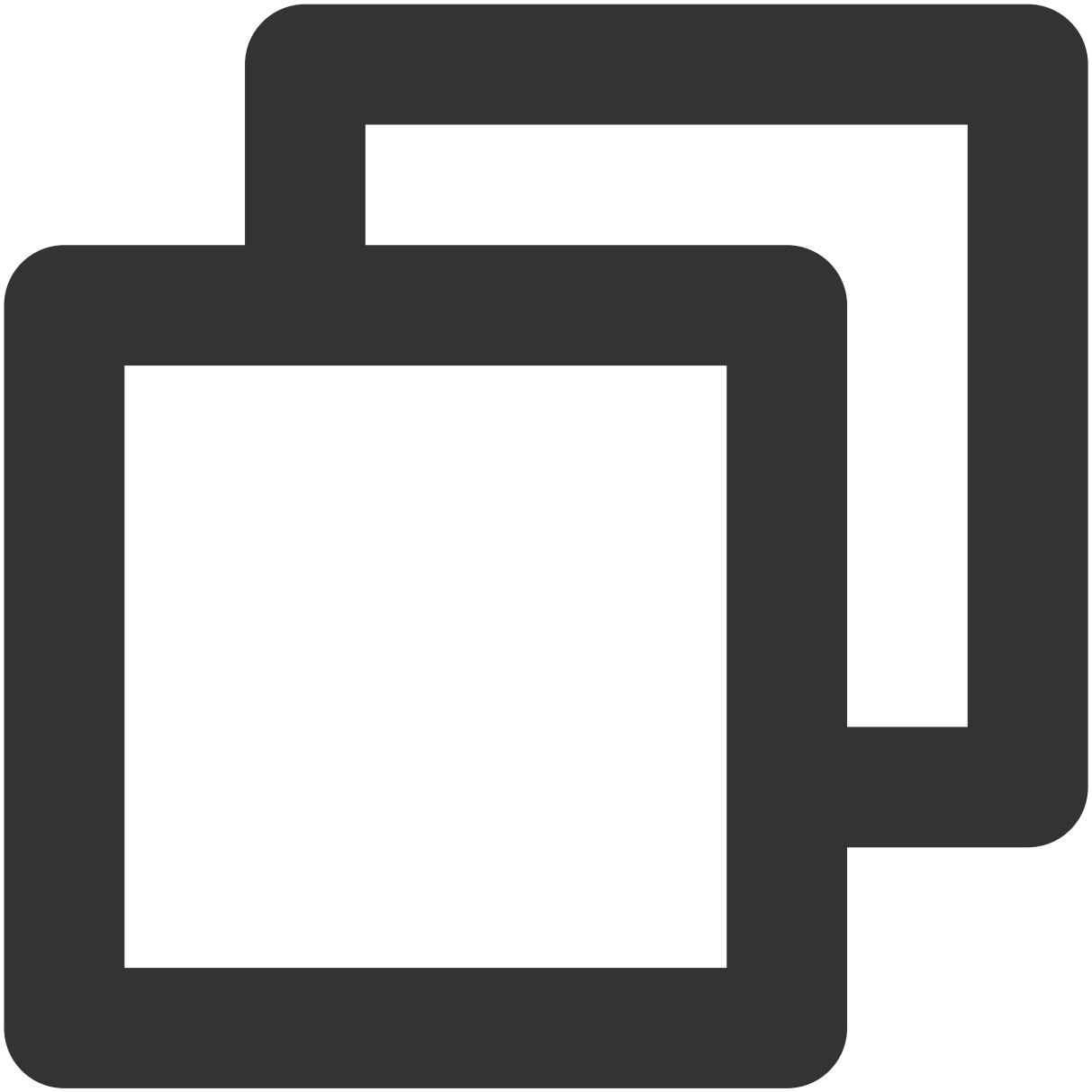
安装时会自动包含 SO 库，建议在 Module 的 build.gradle 文件中使用 NDK 的“abiFilter”配置，设置支持的 SO 库架构。



```
defaultConfig {
    ndk {
        // 设置支持的 SO 库架构
        abiFilters 'armeabi' //, 'x86', 'armeabi-v7a', 'x86_64', 'arm64-v8a'
    }
}
```

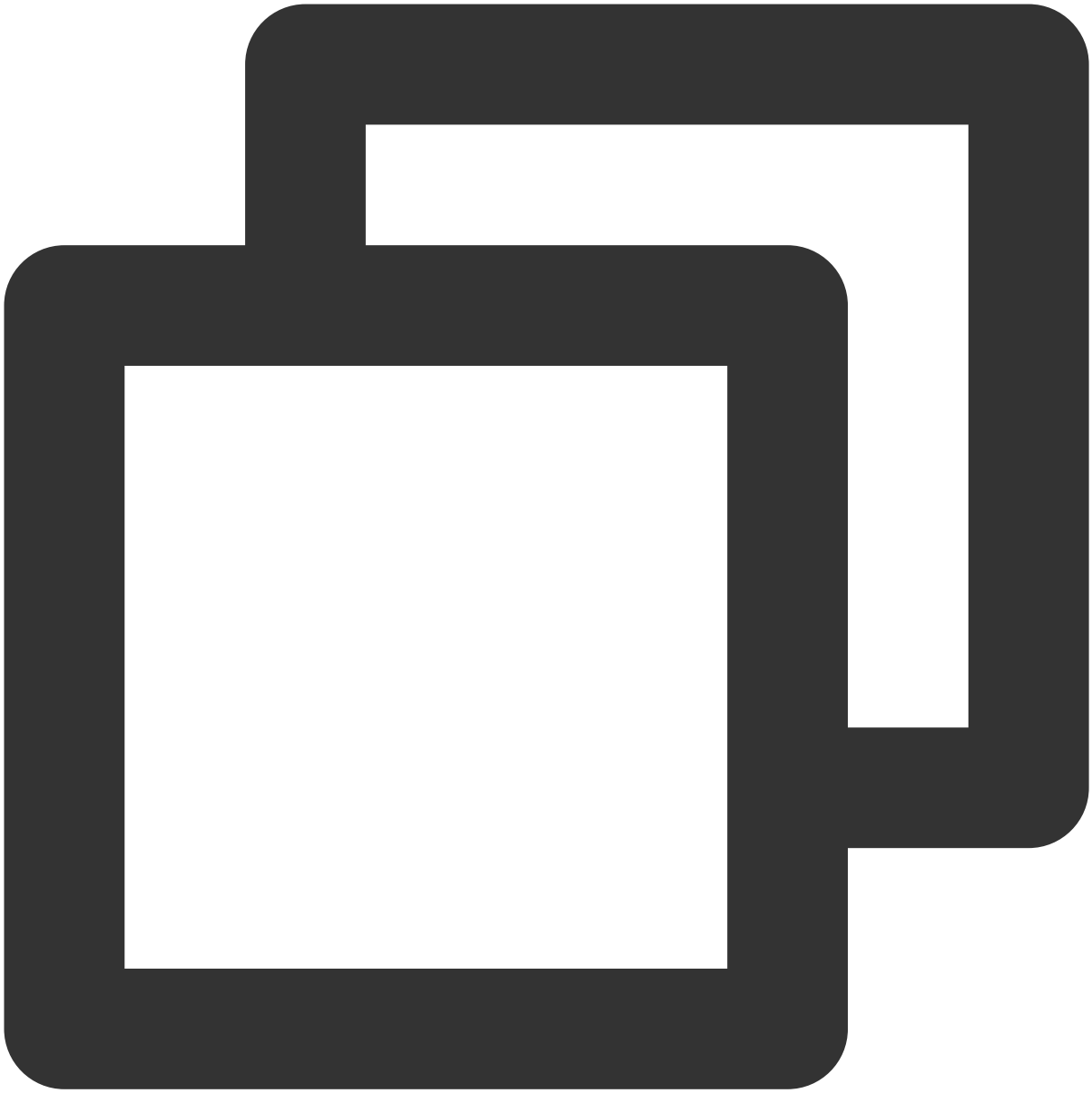
方式一：加载网络 AVIF 图片

1. 集成 cloud-infinite SDK。



```
implementation 'com.qcloud.cos:cloud-infinite:1.2.1'
```

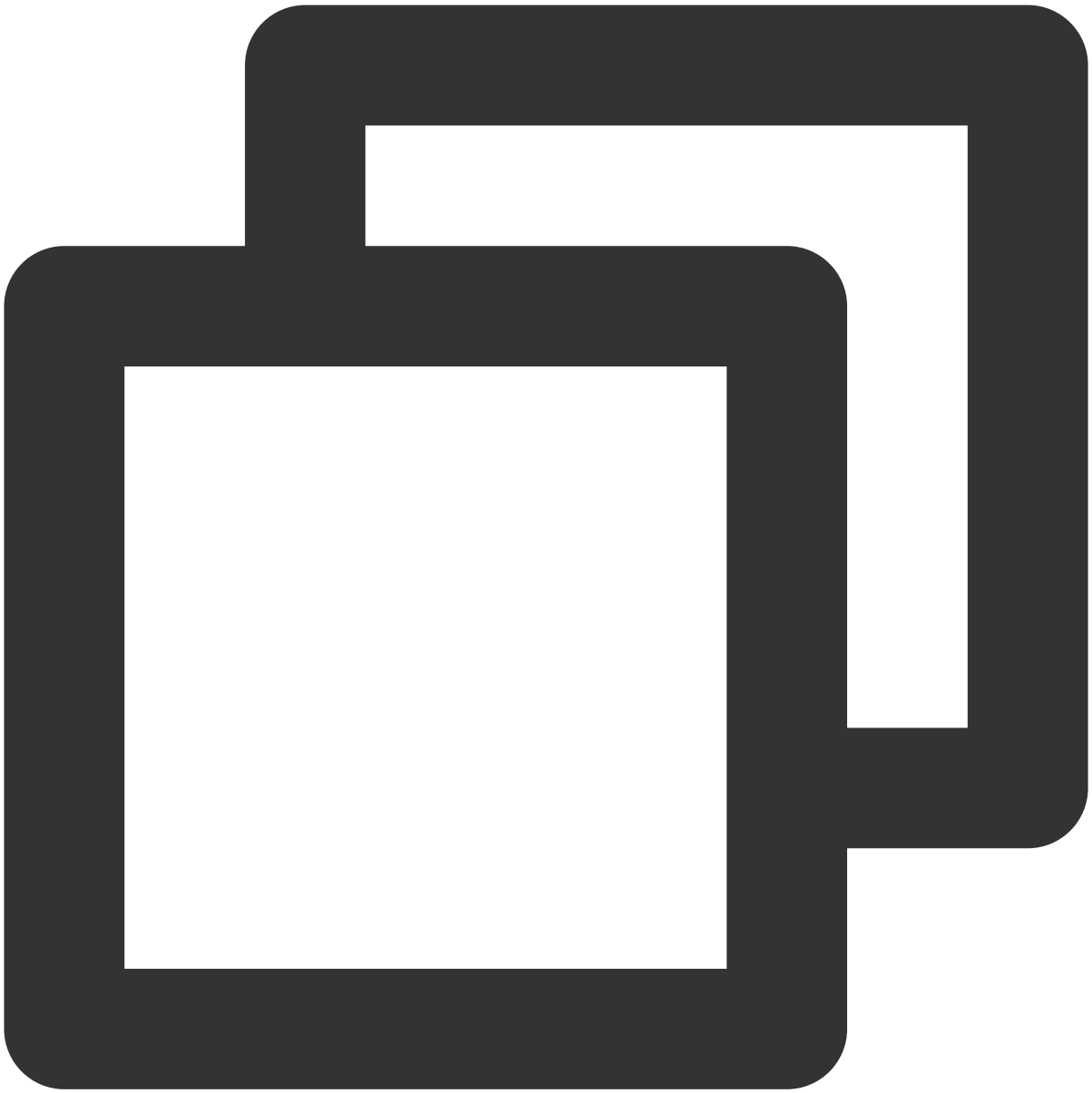
2. 在 cloud-infinite SDK 中构建出请求 AVIF 格式图片的链接，然后与 Glide 加载图片 配合使用，加载网络 AVIF 图片。



```
// 实例化 CloudInfinite，用来构建请求图片请求连接；
CloudInfinite cloudInfinite = new CloudInfinite();
// 根据用户所选万象基础功能 options 进行 Transformation；
CITransformation transform = new CITransformation();
transform.format(CIImageFormat.AVIF, CIImageLoadOptions.LoadTypeUrlFooter);
// 构建图片 CIImageLoadRequest
CIImageLoadRequest request = cloudInfinite.requestWithBaseUrlSync(url, transform);
```

方式二：加载本地 AVIF 图片

应用内置的资源，例如 assets、drawable、raw 等，使用 AVIF 格式可以有利于减小安装包大小。



```
//加载 Assets 中的 AVIF 图片
AVIFImageLoader.displayWithAssets(imageview, assetsName);
//加载 Resource 中的 AVIF 图片
AVIFImageLoader.displayWithResource(imageview, R.drawable.avif);
//加载本地文件中的 AVIF 图片
AVIFImageLoader.displayWithFileUri(imageview, fileUri);
```

版权所有：腾讯云计算（北京）有限责任公司

图片处理 iOS SDK 快速入门

最近更新时间：2024-03-14 09:36:58

简介

万象图片终端 SDK 主要提供了 AVIF 和 TPG 的图片解码器以及万象的图片基础处理功能。

模块	功能
AVIF	AVIF 图片解码器以及 SDWebImage 生态封装。
TPG	TPG 图片解码器以及 SDWebImage 等生态封装。
CloudInfinite	该模块包含数据万象的基础图片处理操作，并支持各个操作能够相互组合，构建 URL 进行网络请求。

说明：

图片压缩瘦身介绍请参见 图片压缩瘦身。

CDN 云闪图片分发介绍请参见 云闪图片分发。

关于 TPG 的更多介绍，请参见 TPG 功能介绍。

关于 AVIF 的更多介绍，请参见 AVIF 功能介绍。

相关资源

SDK 源码和 Demo 请参见 CI iOS SDK。

SDK 更新日志请参见 ChangeLog。

准备工作

1. 您需要一个 iOS 应用，这个应用可以是您现有的工程，也可以是您新建的一个空的工程。
2. 请确保应用基于 iOS 8.0 及以上版本的 SDK 构建。

集成与使用

根据您需要的模块查看具体的使用文档，进行集成和使用。

TPG 解码器：

TPG 解码器基础使用请参见 TPG 解码器。

数据万象的图片基础处理：

图片基础处理请参见 图片处理。

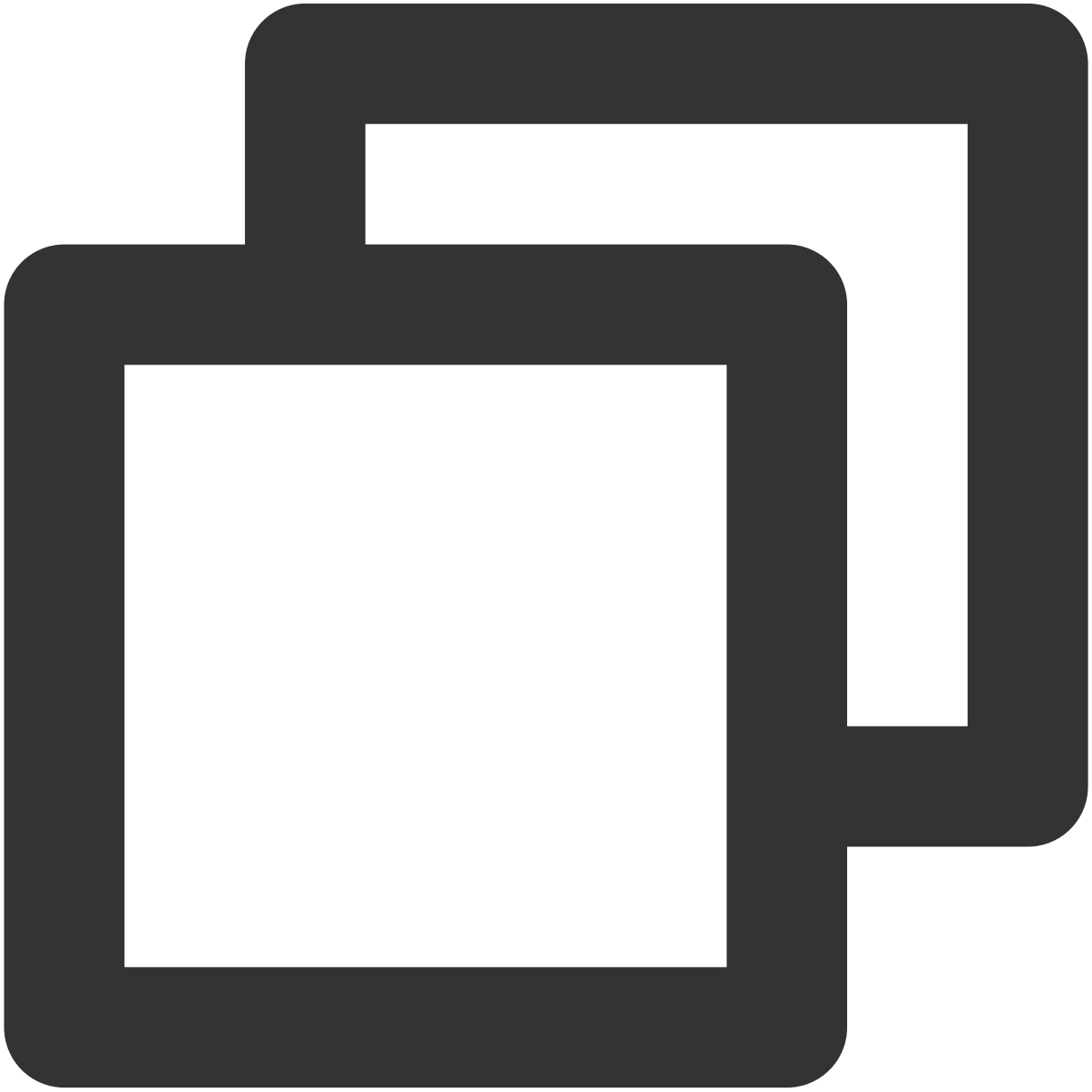
加载 TPG 图片

最近更新时间：2024-03-13 15:17:15

本文提供加载网络以及使用 TPG 模块两种方式加载 TPG 图片。

方式一：加载网络 TPG 图片

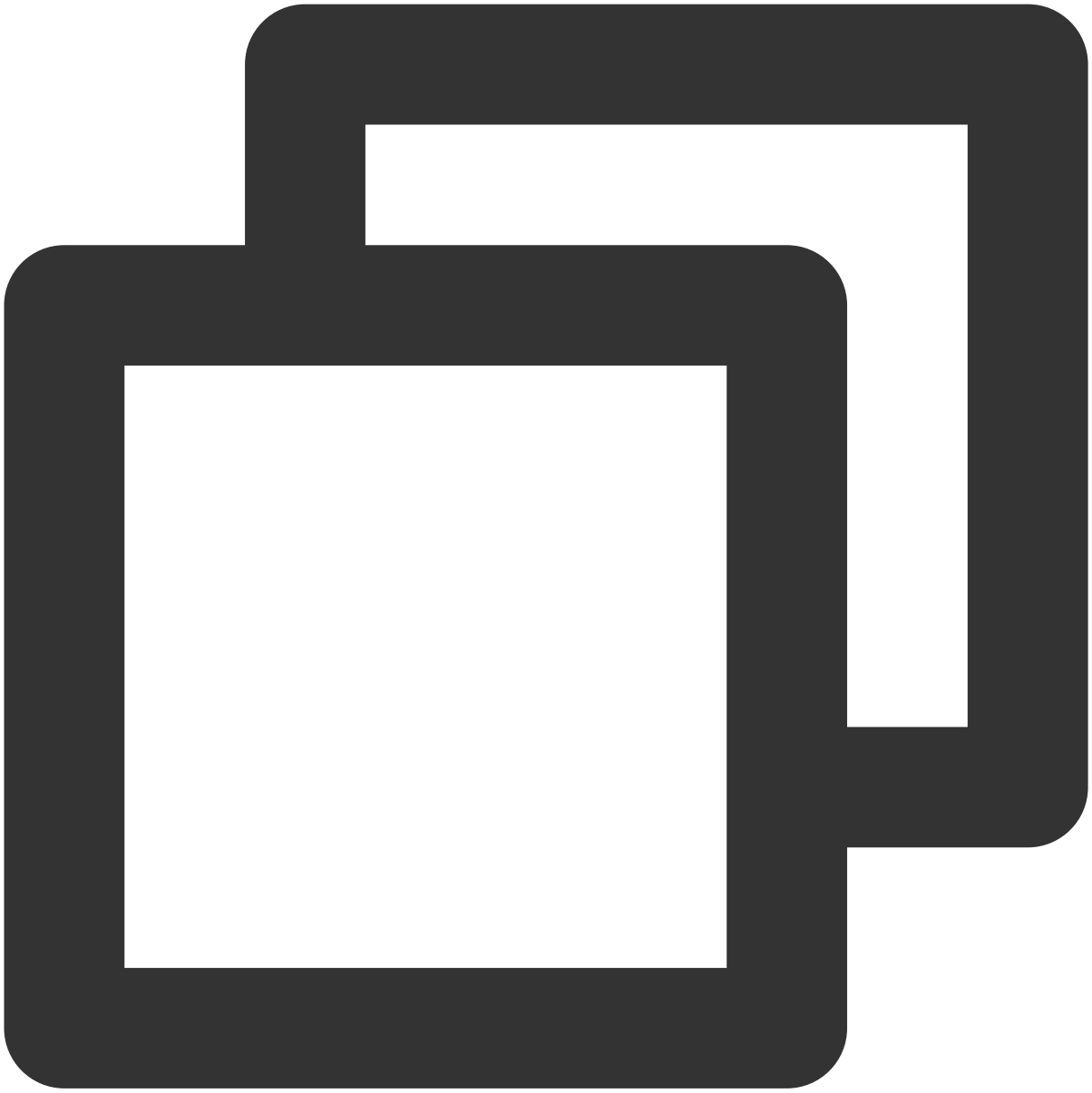
1. 首先集成 CloudInfinite。



```
pod 'CloudInfinite'
```

2. 在 CloudInfinite 模块中构建出请求 TPG 格式图片的链接，然后与SDWebImage 配合使用，加载网络 TPG 图片。

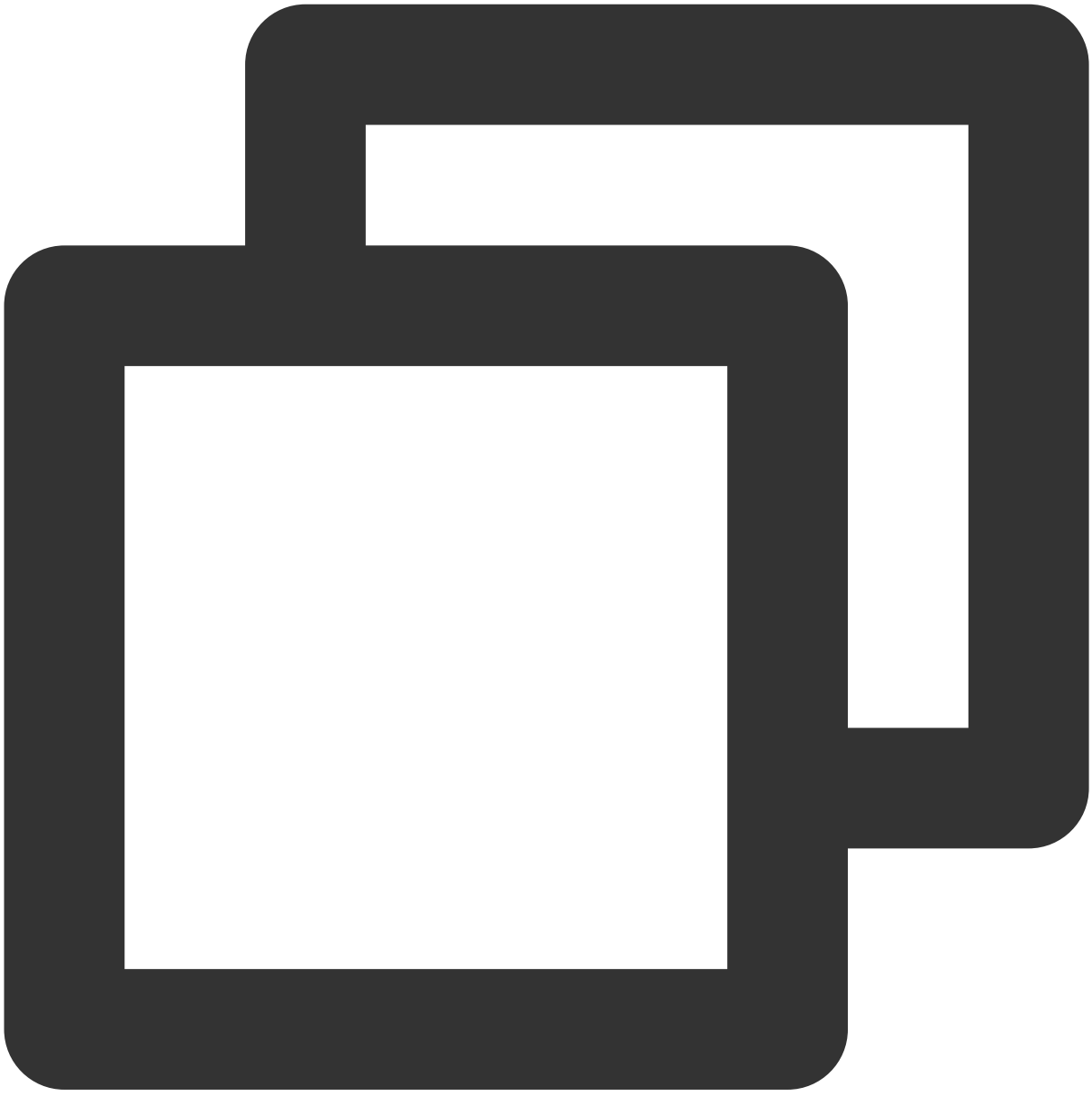
Objective-C



```
// 实例化 CloudInfinite，用来构建请求图片请求连接；
CloudInfinite * cloudInfinite = [CloudInfinite new];

// 根据用户所选万象基础功能 options 进行构建 CIImageLoadRequest；
CITransformation * transform = [CITransformation new];
[transform setFormatWith:CIImageTypeTPG options:CILoadTypeUrlFooter];

// 构建图片 CIImageLoadRequest
```

```
[cloudInfinite requestWithBaseUrl:@"图片链接" transform:transform request:^(CIImageL
    // request 构建成功的 CIImageLoadRequest 实例,
}];
```

swift



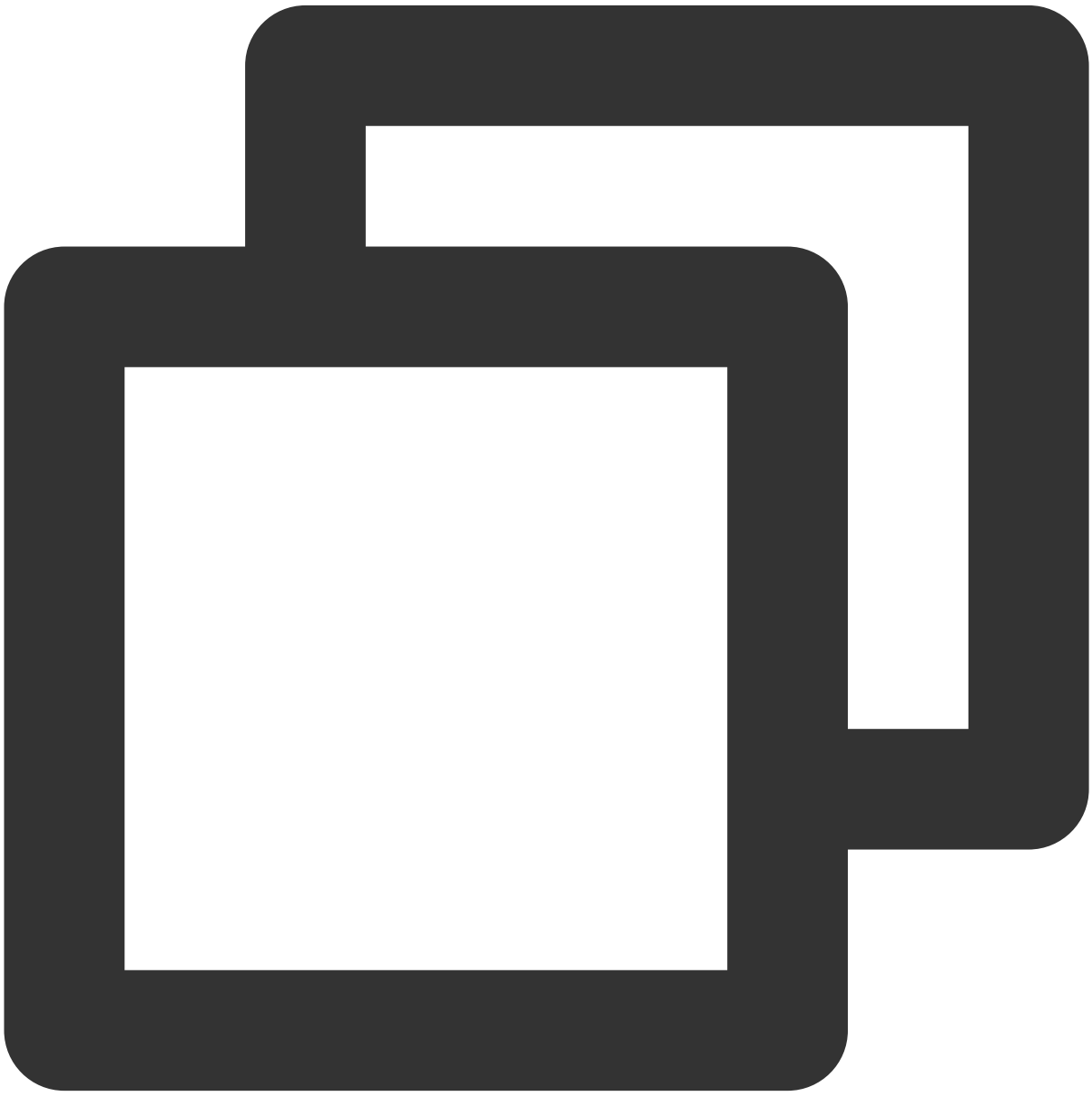
```
// 实例化 CloudInfinite, 用来构建请求图片请求连接；
let cloudInfinite = CloudInfinite();

// 根据用户所选的数据万象基础功能 options 进行构建 CIImageLoadRequest；
let transform = CITransformation();
transform.setFormatWith(CIImageFormat.typeTPG, options: CILoadTypeEnum.urlFoote
```

```
// 构建图片 CIImageLoadRequest
cloudInfinite.request(withBaseUrl: "图片链接", transform: transform) { (request)
    // request 构建成功的 CIImageLoadRequest 实例,
}
```

方式二：使用 TPG 模块加载 TPG 图片

使用 TPG 模块加载 TPG 图片 Data 数据，支持加载 TPG 动图，无需额外处理。

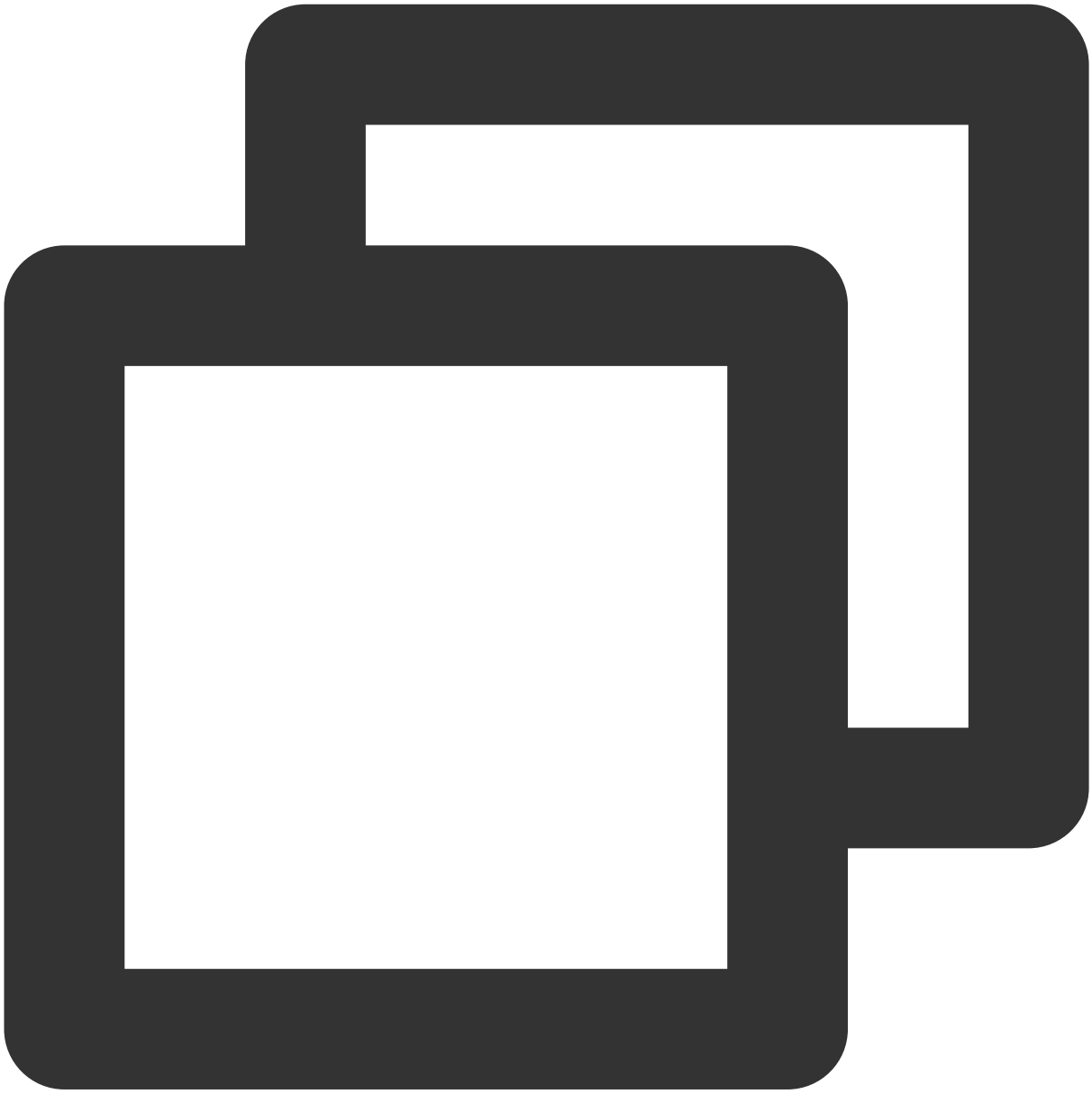
1. 首先集成 TPG 模块。



```
pod 'CloudInfinite/TPG'
```

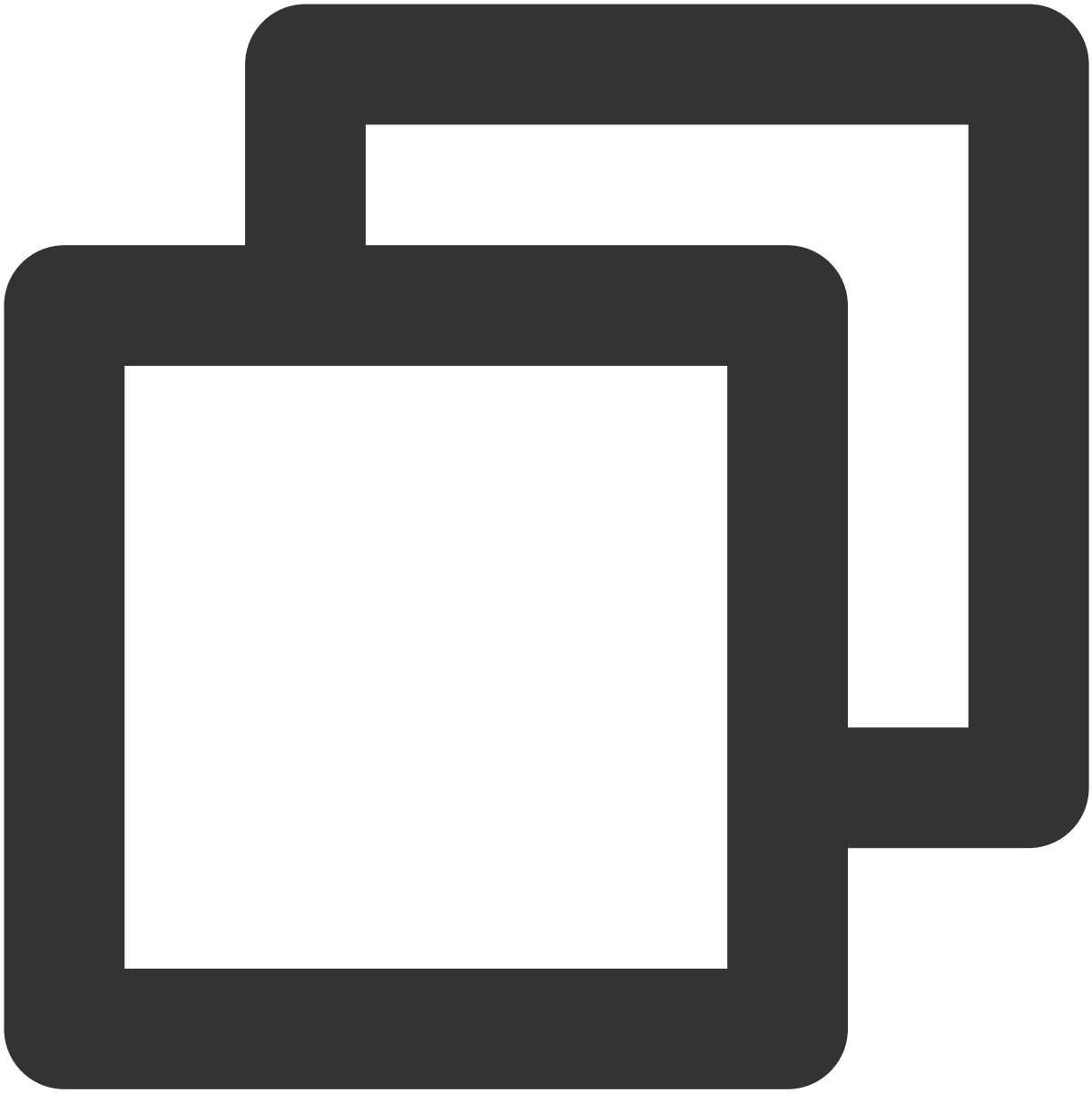
2. 如果已经获取到 TPG 图片 data 数据，则直接使用 TPG 模块 UIImageView+TPG 类进行解码并显示。

Objective-C



```
    [self.tpgImageView setTpgImageWithData:data loadComplete:^(NSData * _Nullable
}];
```

swift



```
    imageView.setTpgImageWith(data) { (data, image, error) in
}
```

图片基础处理
图片处理

最近更新时间：2024-03-13 15:17:11

简介

数据万象提供两种方式进行图片处理：图片自适应处理 CIResponsiveTransformation 和自定义转换处理 CITransformation。

说明：

支持格式：支持处理 JPG、BMP、GIF、PNG、WEBP 格式，并且支持 HEIF 格式的解码和处理操作。

体积限制：处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

动图帧数限制：gif 帧数限300帧。

相关资源

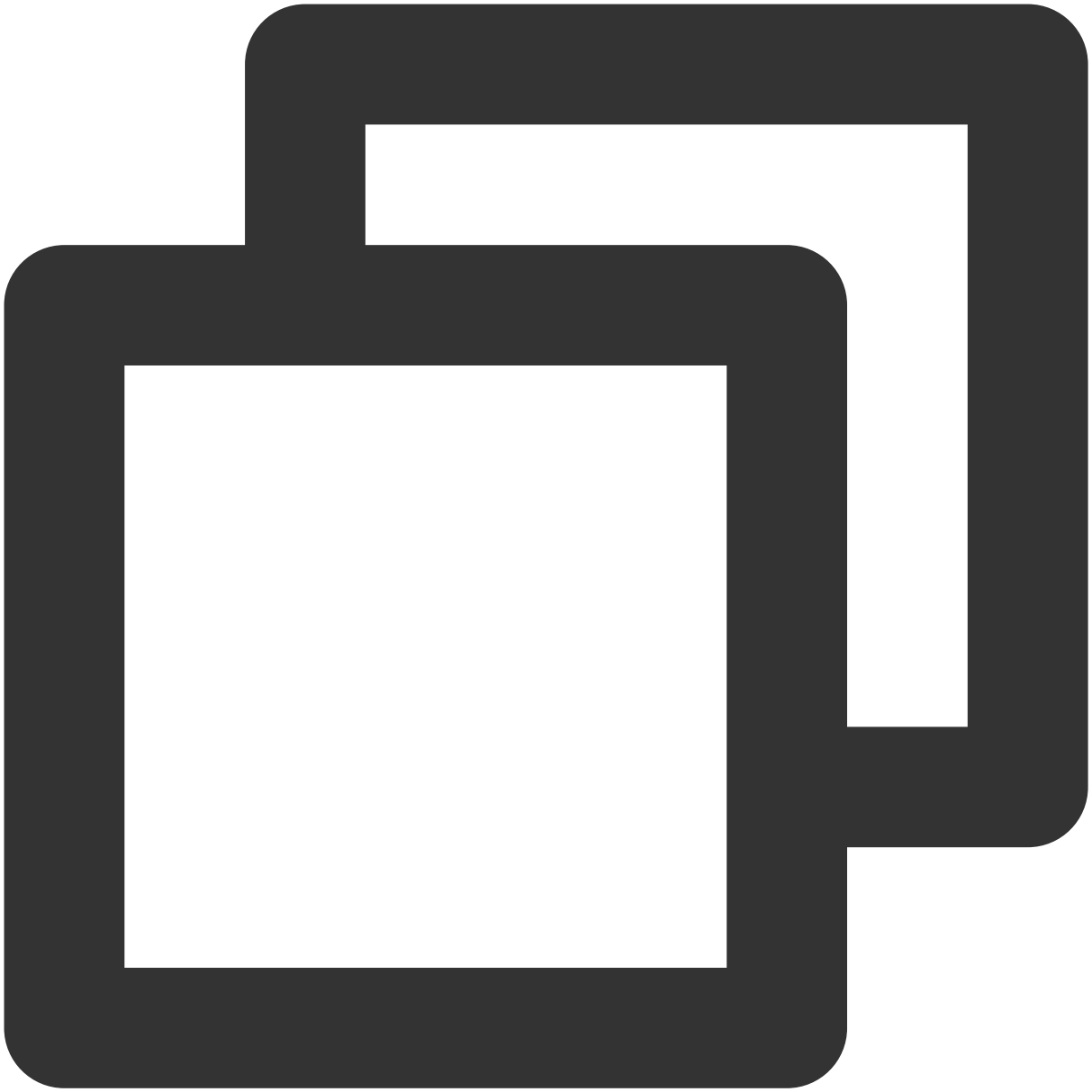
SDK 接口与参数文档请参见 SDK API。

图片自适应处理 CIResponsiveTransformation

图片自适应处理包括了格式转换（根据系统版本自动变换格式）和缩放（根据图片控件尺寸以及 ScaleType 进行缩放），相关说明如下：

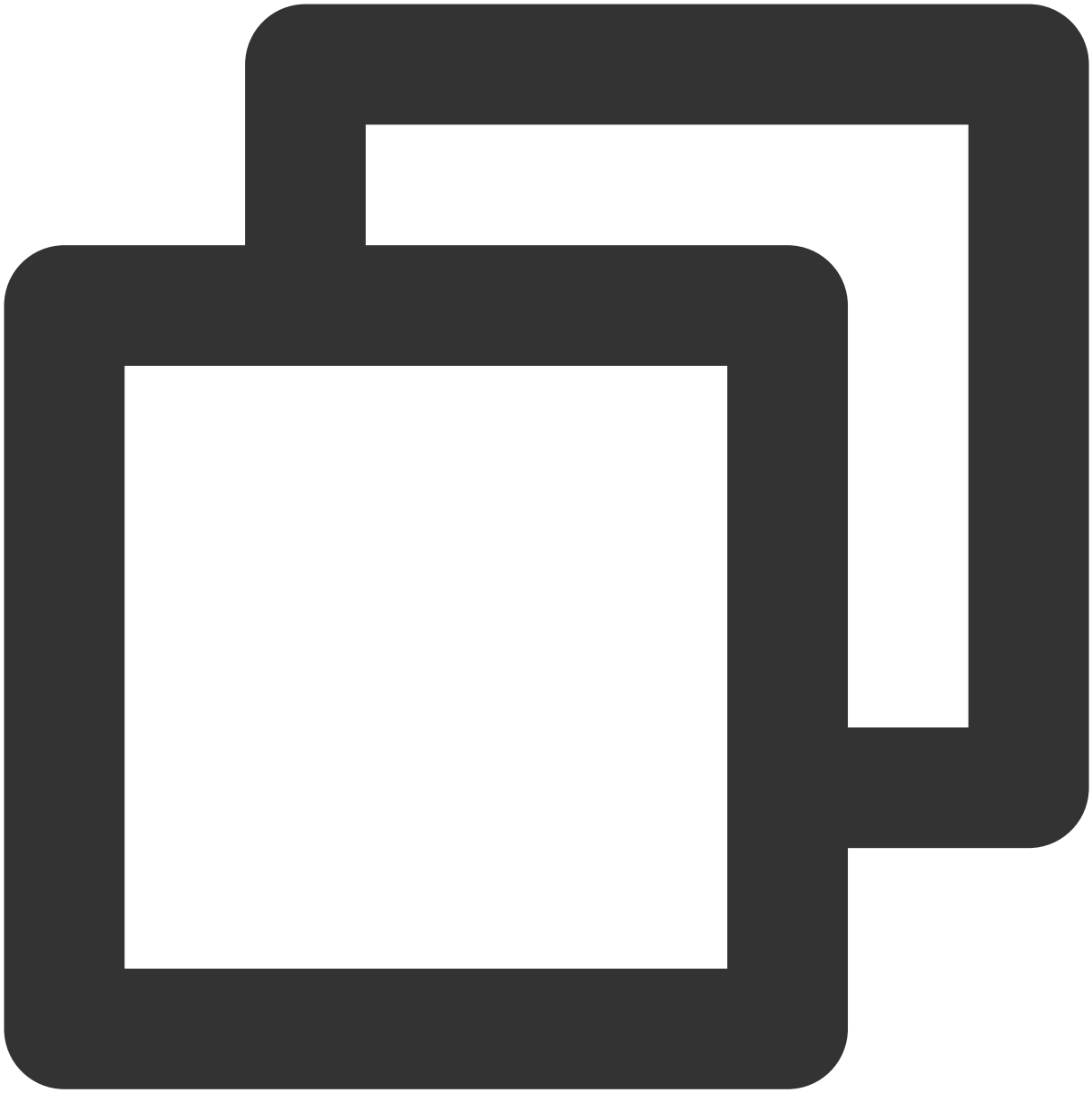
1. 自适应加载 CIResponsiveTransformation。
2. 根据当前 imageView 控件尺寸自动调整图片大小。
3. 根据系统版本自动调整最优图片格式，iOS 11以下加载原格式，iOS 11及以上加载 HEIC 格式。

Objective-C



```
CIResponsiveTransformation * sTransform = [[CIResponsiveTransformation alloc]initWi
```

swift



```
let sTransform = CIResponsiveTransformation(view: nil, scale: ScaleType.autoFit)
```

自定义转换处理 CITransformation

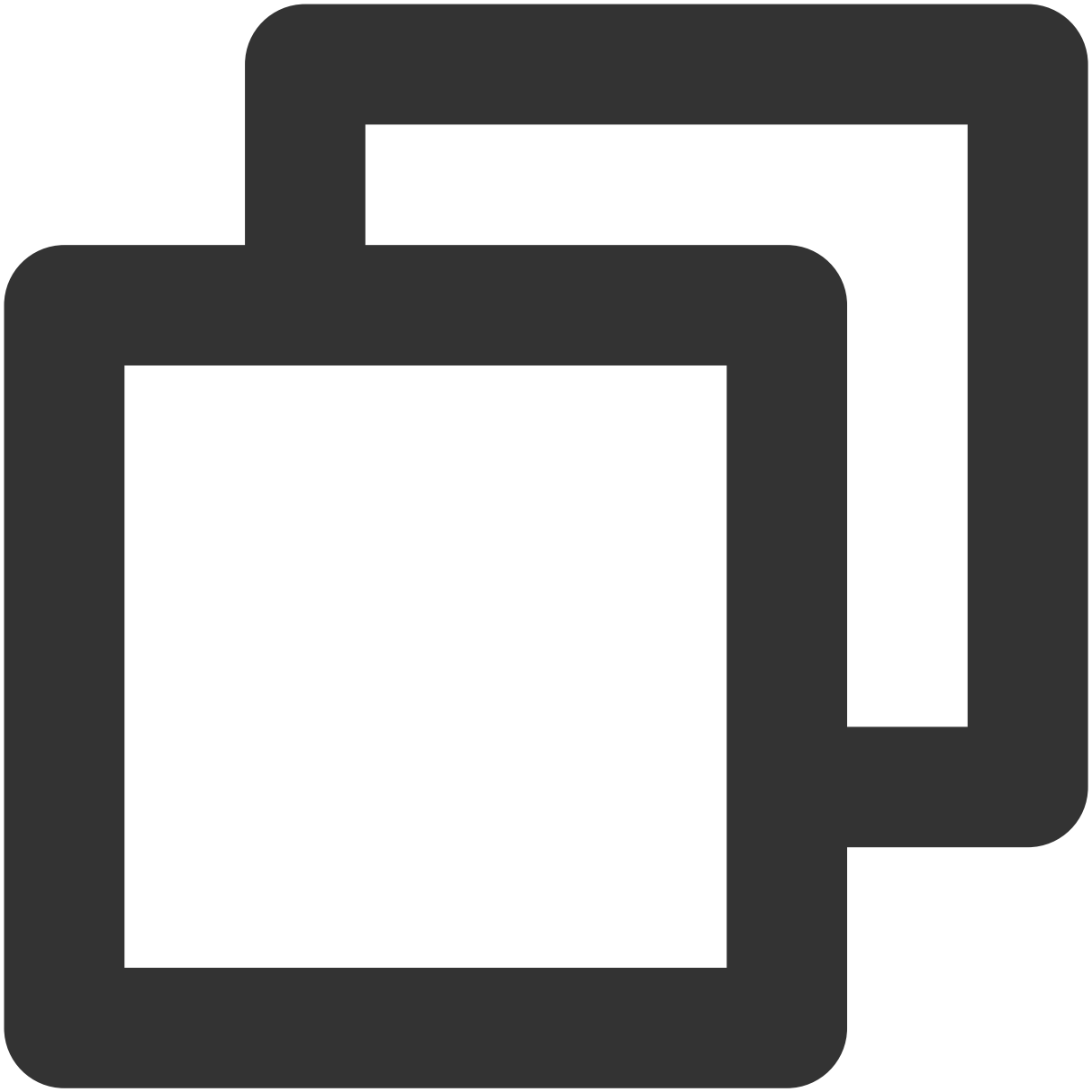
CloudInfinite SDK 中的 CITransformation 类封装了数据万象的基础图片处理功能其中包含如下功能：

缩放
裁剪
旋转

格式转换
质量变换
高斯模糊
锐化
水印
获取图片主题色
去除图片元信息
组合操作

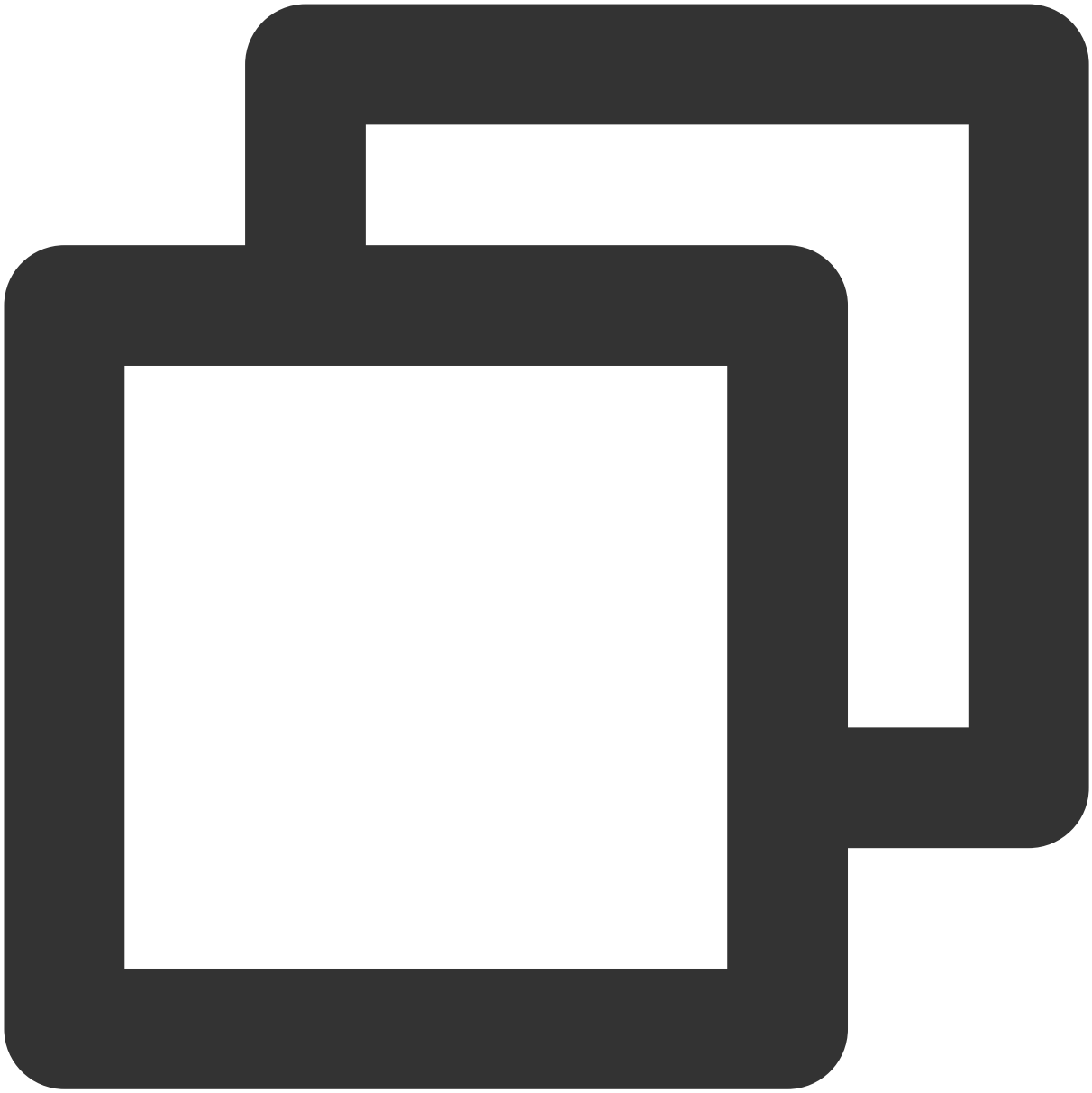
在使用数据万象基础图片处理功能时首先实例化 CITransformation 类，下面所有操作一致，不再重复说明。

Objective-C



```
CITransformation * transform = [CITransformation new];
```

swift



```
let transform = CITransformation();
```

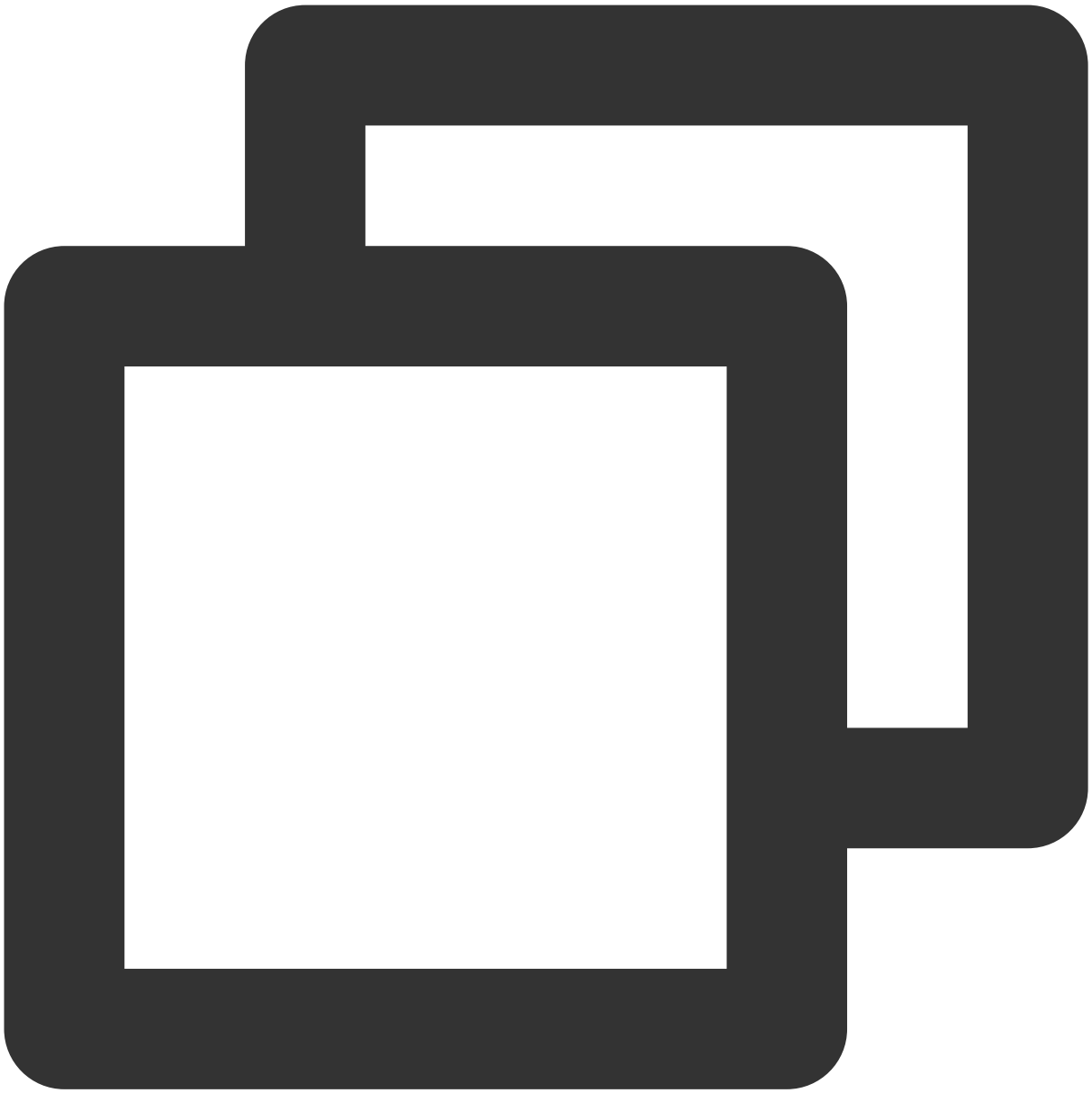
缩放

说明：

API 文档请参见 缩放功能接口。

按百分比缩放

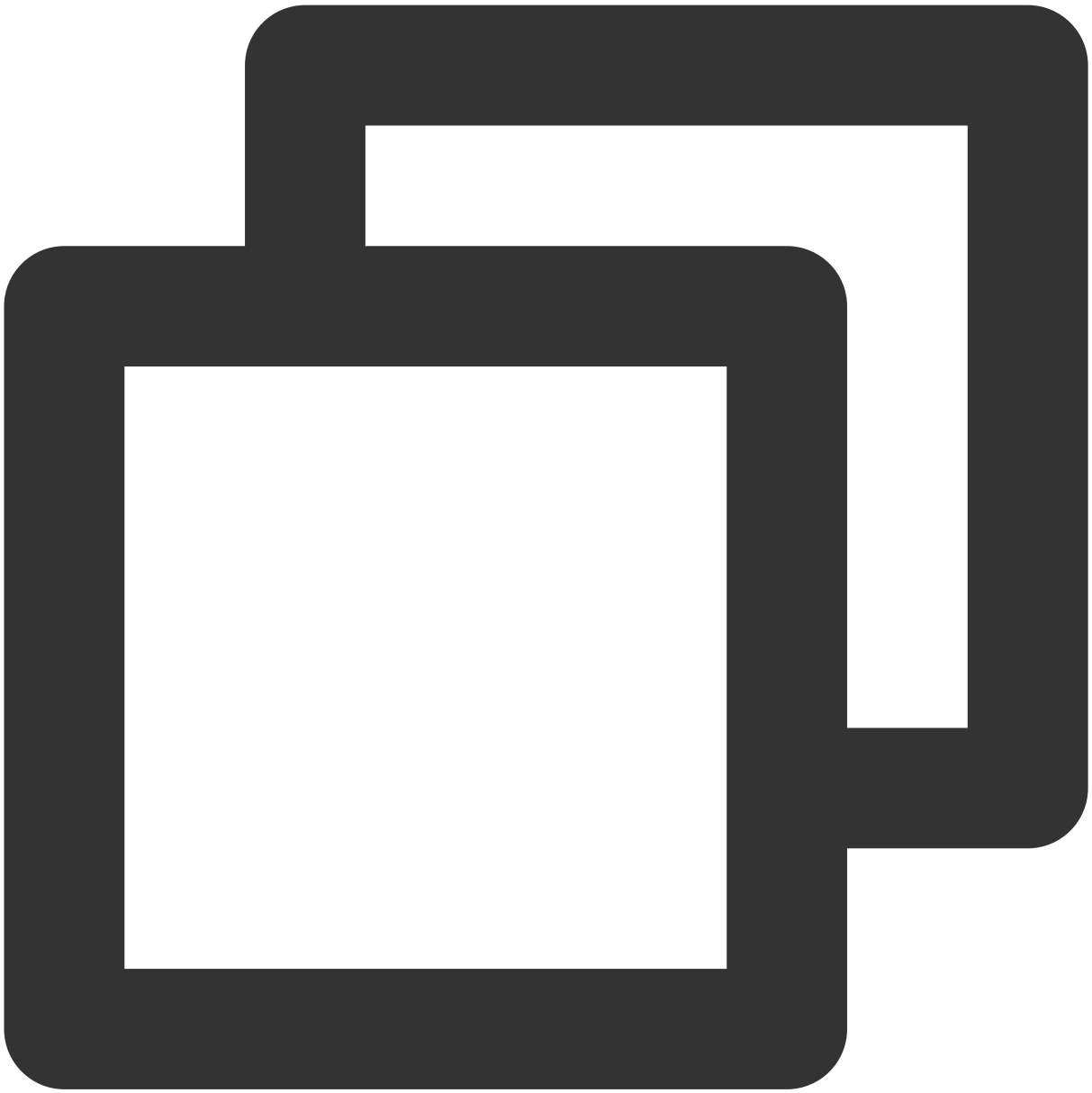
Objective-C



```
// 以缩放百分之50为例
[transform setZoomWithPercent:50 scaleType:ScalePercentTypeALL];

// scaleType 可以指定如下类型：
    // 仅缩放宽度，高度不变
    ScalePercentTypeOnlyWidth = 1,
    // 仅缩放高度，宽度不变
    ScalePercentTypeOnlyHeight,
    //指定图片的宽高同时缩放
```

```
    ScalePercentTypeALL,
```

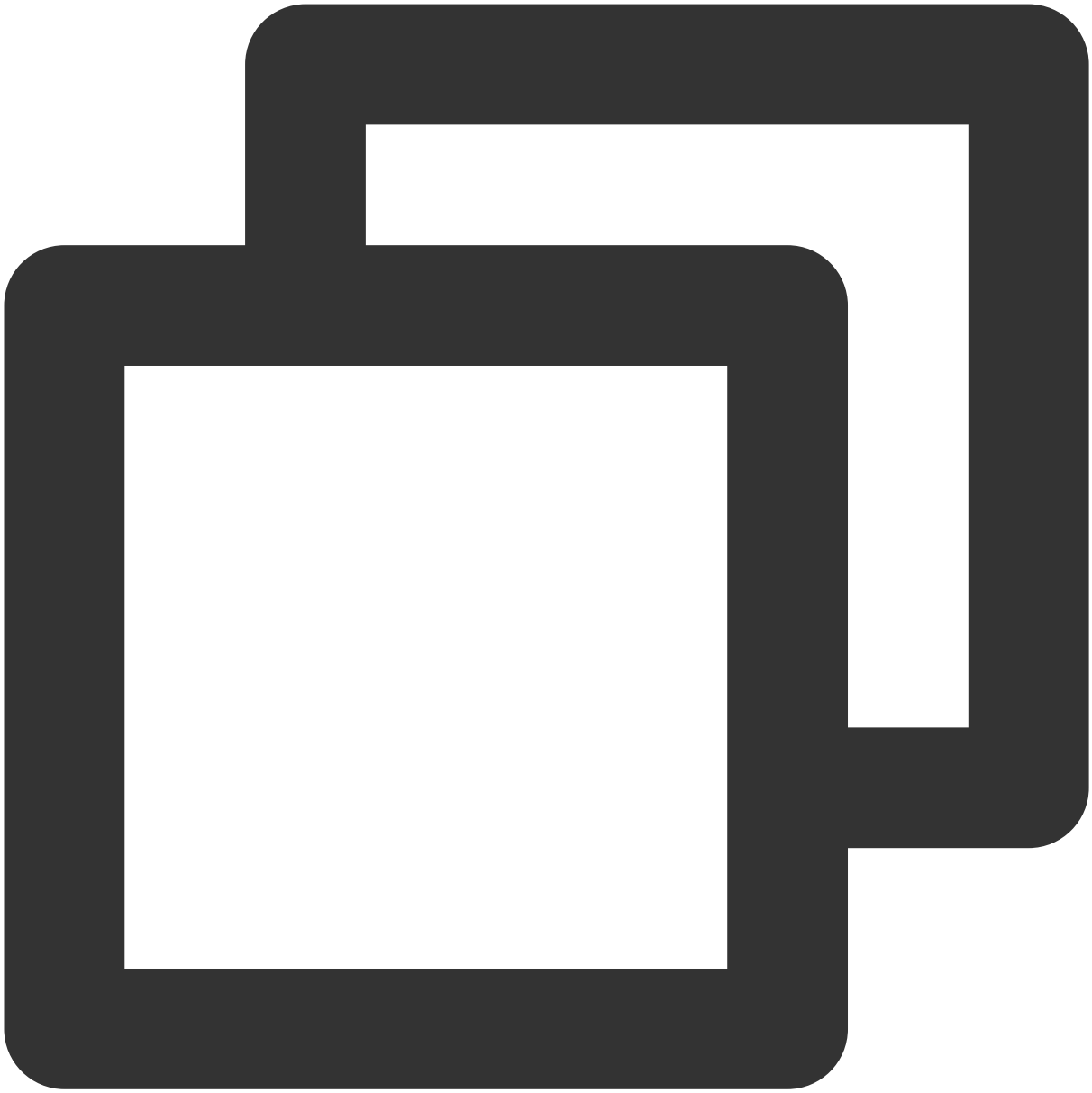
swift



```
transform.setZoomWithPercent(50, scale: ScalePercentType.ALL)
```

指定宽高缩放

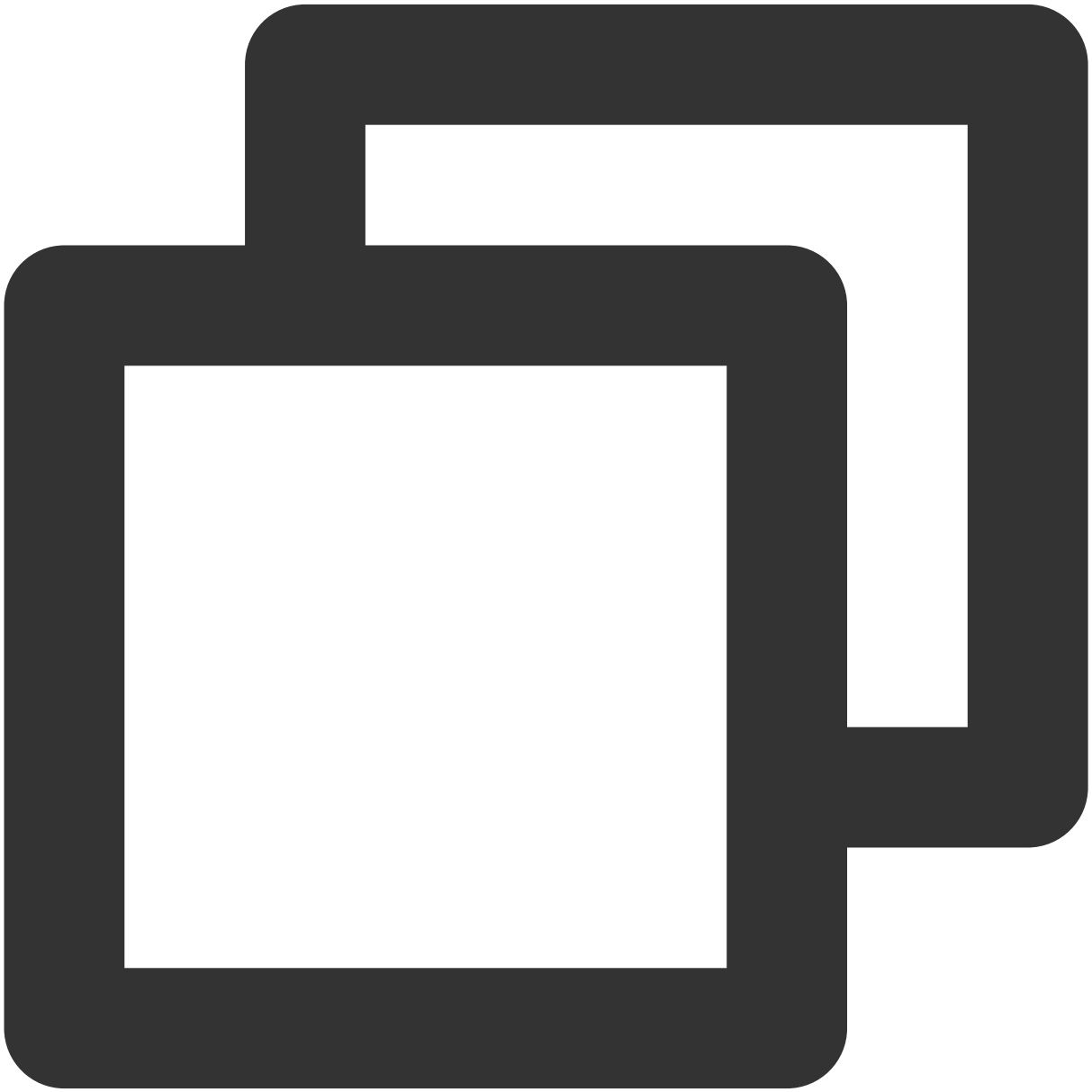
Objective-C



```
// 以宽高都指定为100 并且缩放类型为 ScaleTypeAUTOFit 等比缩放
[transform setZoomWithWidth:100 height:100 scaleType:ScaleTypeAUTOFit];

// 在指定宽高缩放时，scaleType 可以指定的类型：
    // 忽略原图宽高比例，指定图片宽度为 Width，高度为 Height ，强行缩放图片，可能导致目标图片变
    ScaleTypeAUTOFill = 1,

    // 限定缩略图的宽度和高度的最大值分别为 Width 和 Height，进行等比缩放
    ScaleTypeAUTOFit,

    // 限定缩略图的宽度和高度的最小值分别为 Width 和 Height，进行等比缩放
```

```
    ScaleTypeAUTOFITWithMin,

    // 指定目标图片宽度为 Width，高度传入0， 高度等比压缩
    ScaleTypeOnlyWidth,

    // 指定目标图片高度为 Height，宽度传0，宽度等比压缩
    ScaleTypeOnlyHeight
```

swift

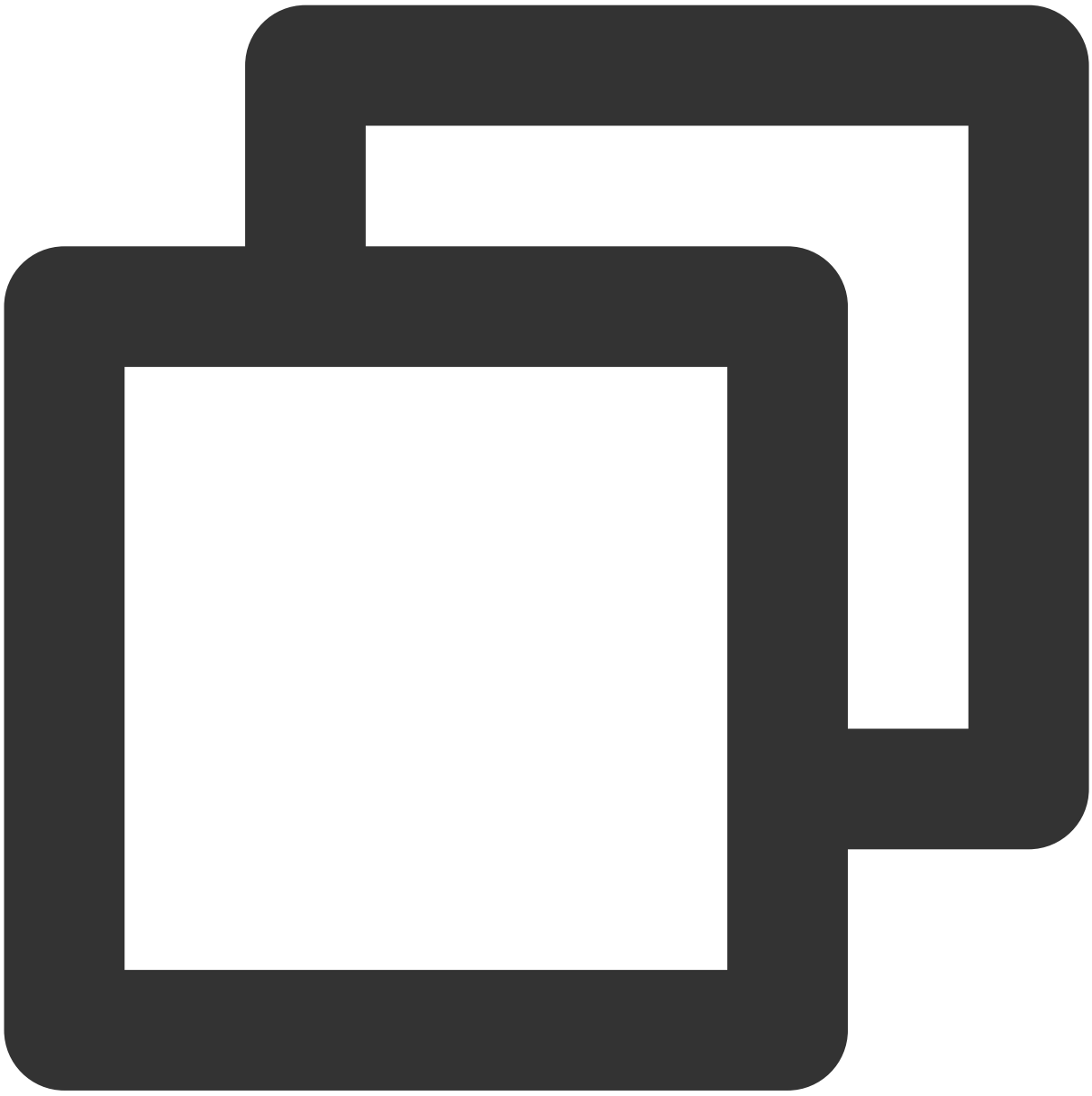


```
transform.setZoomWithWidth(10, height: 100, scaleType: ScaleType.autoFit)
```

等比缩放

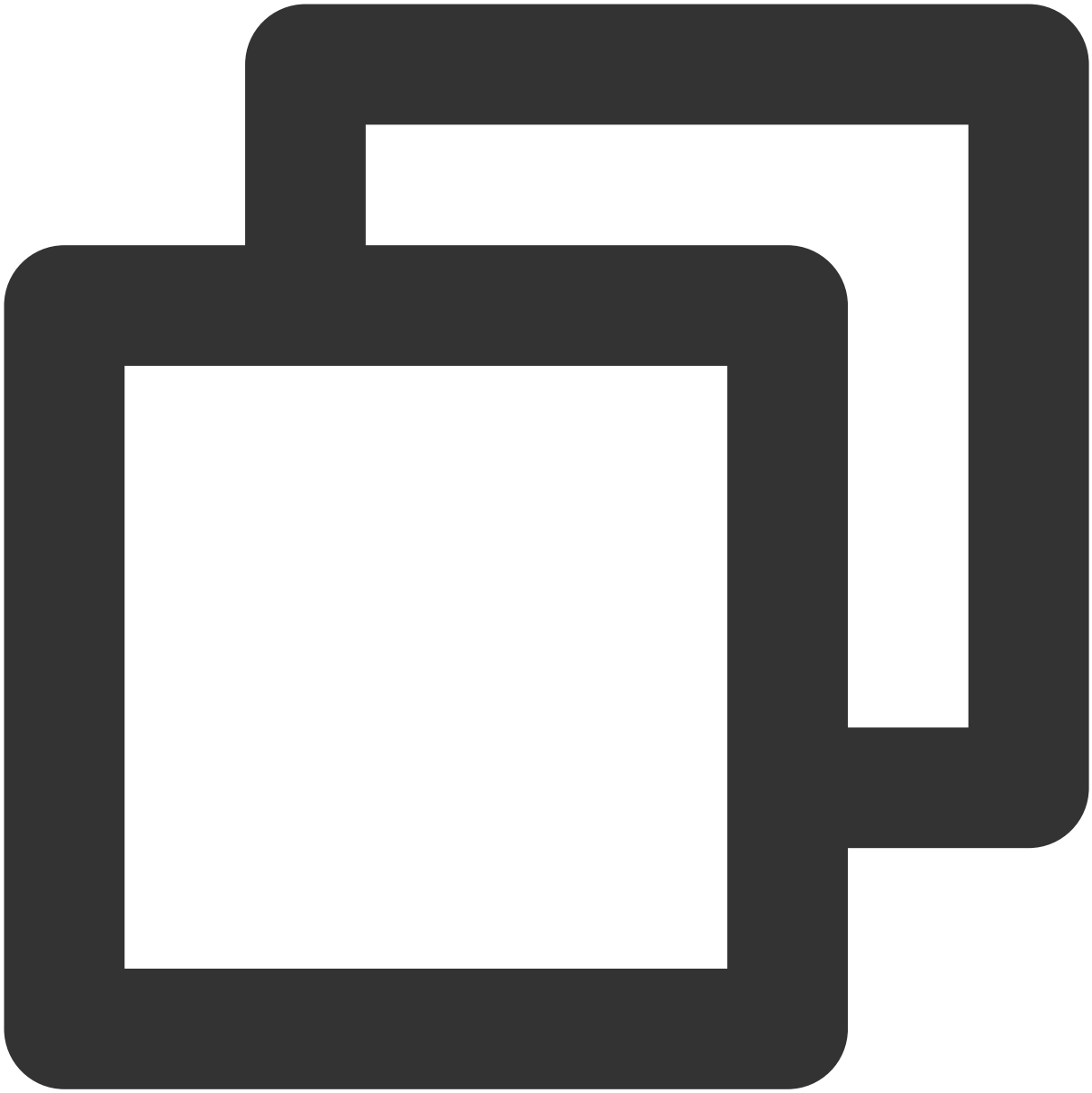
限制像素缩放，缩放后的图像，总像素数量不超过指定数量。

Objective-C



```
// 缩放后的图像，总像素数量不超过1000 为例；
[transform setZoomWithArea:1000];
```

swift



```
transform.setZoomWithArea(1000)
```

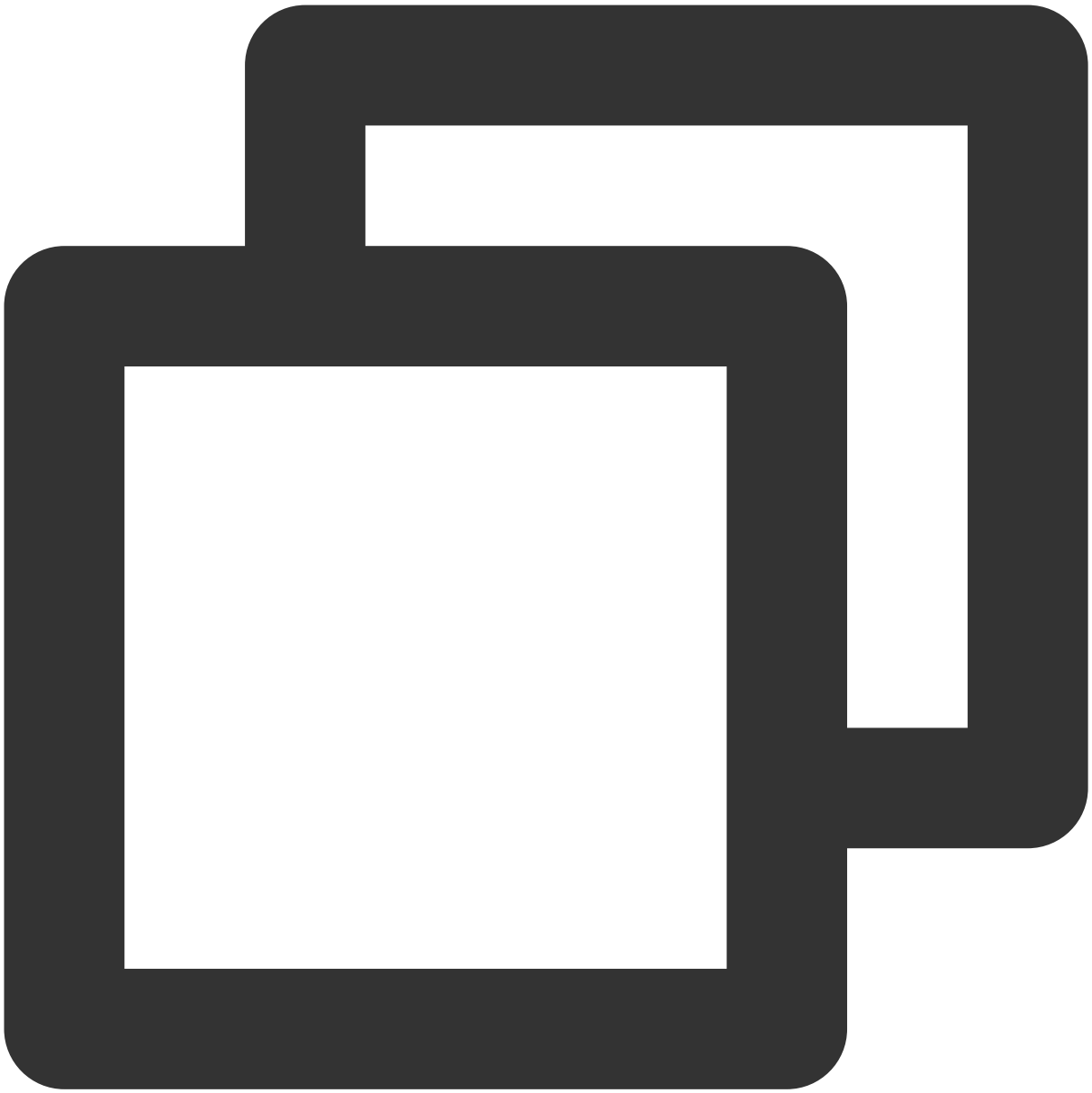
裁剪

说明：

API 文档请参见 裁剪功能接口。

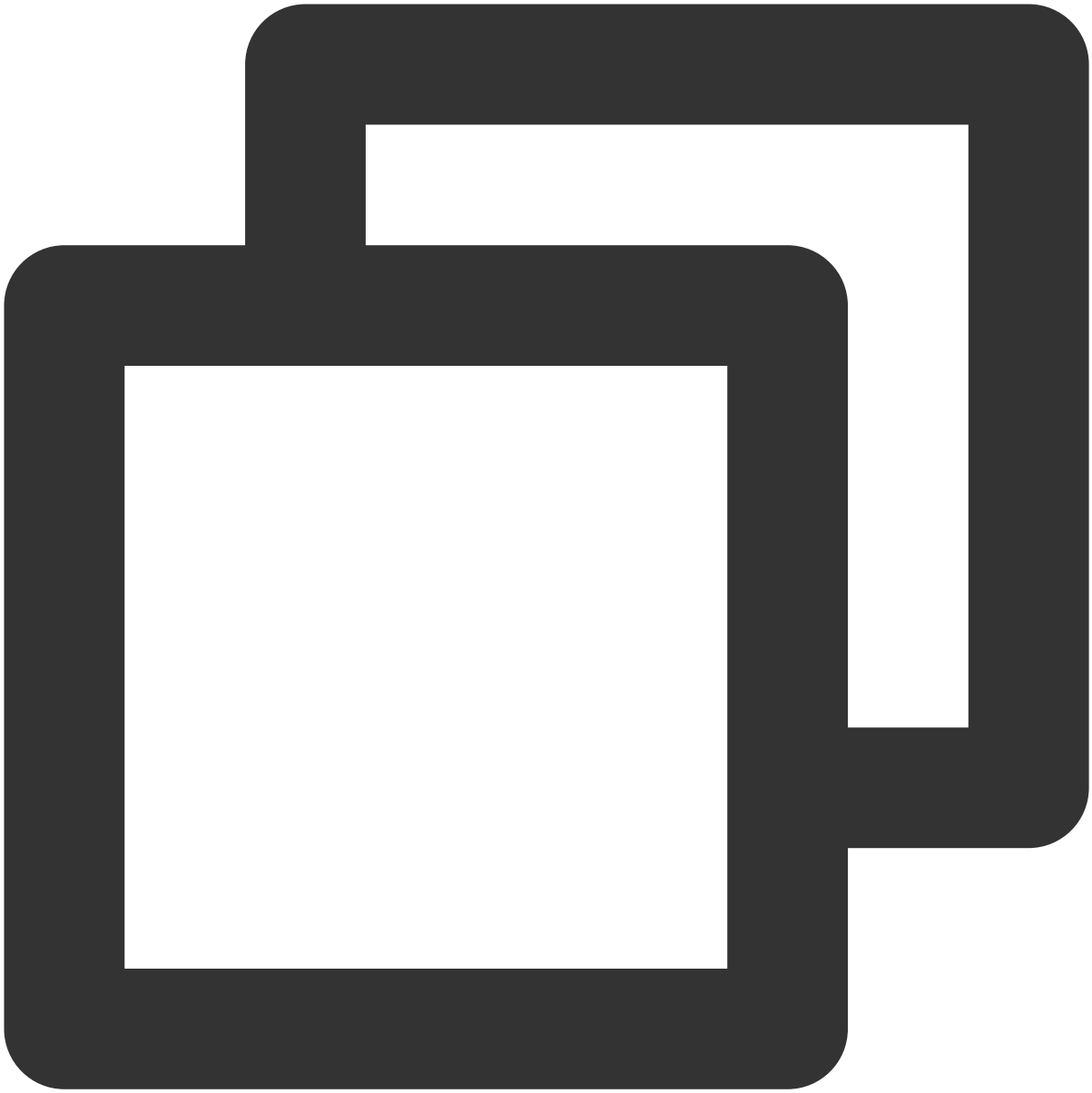
普通裁剪

Objective-C



```
// 指定目标图片宽度、高度、相对于图片左上顶点水平向右偏移、相对于图片左上顶点水平向下偏移进行裁剪为
[transform setCutWithWidth:100 height:100 dx:30 dy:30];
```

swift

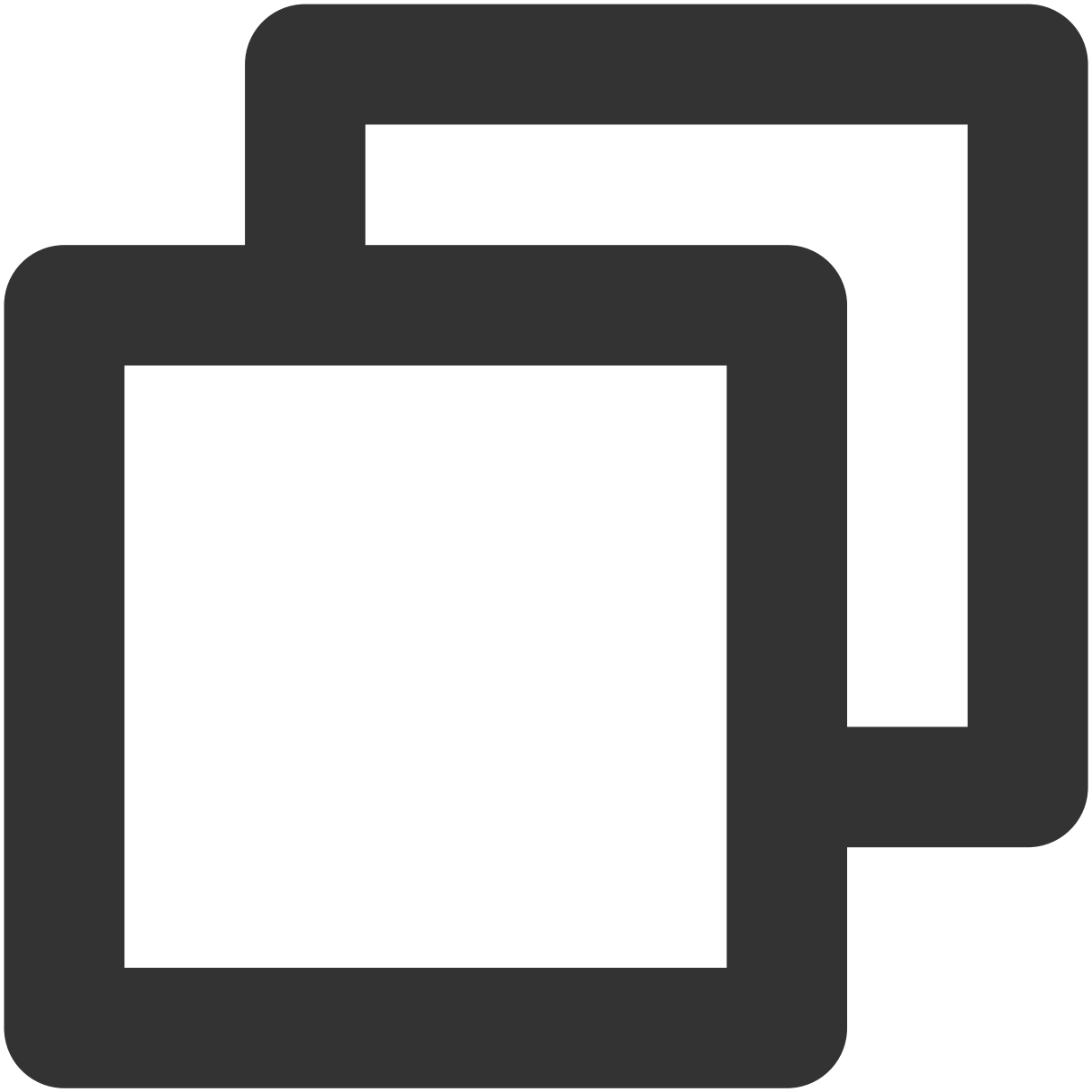


```
transform.setCutWithWidth(100, height: 100, dx: 30, dy: 30)
```

内切圆裁剪

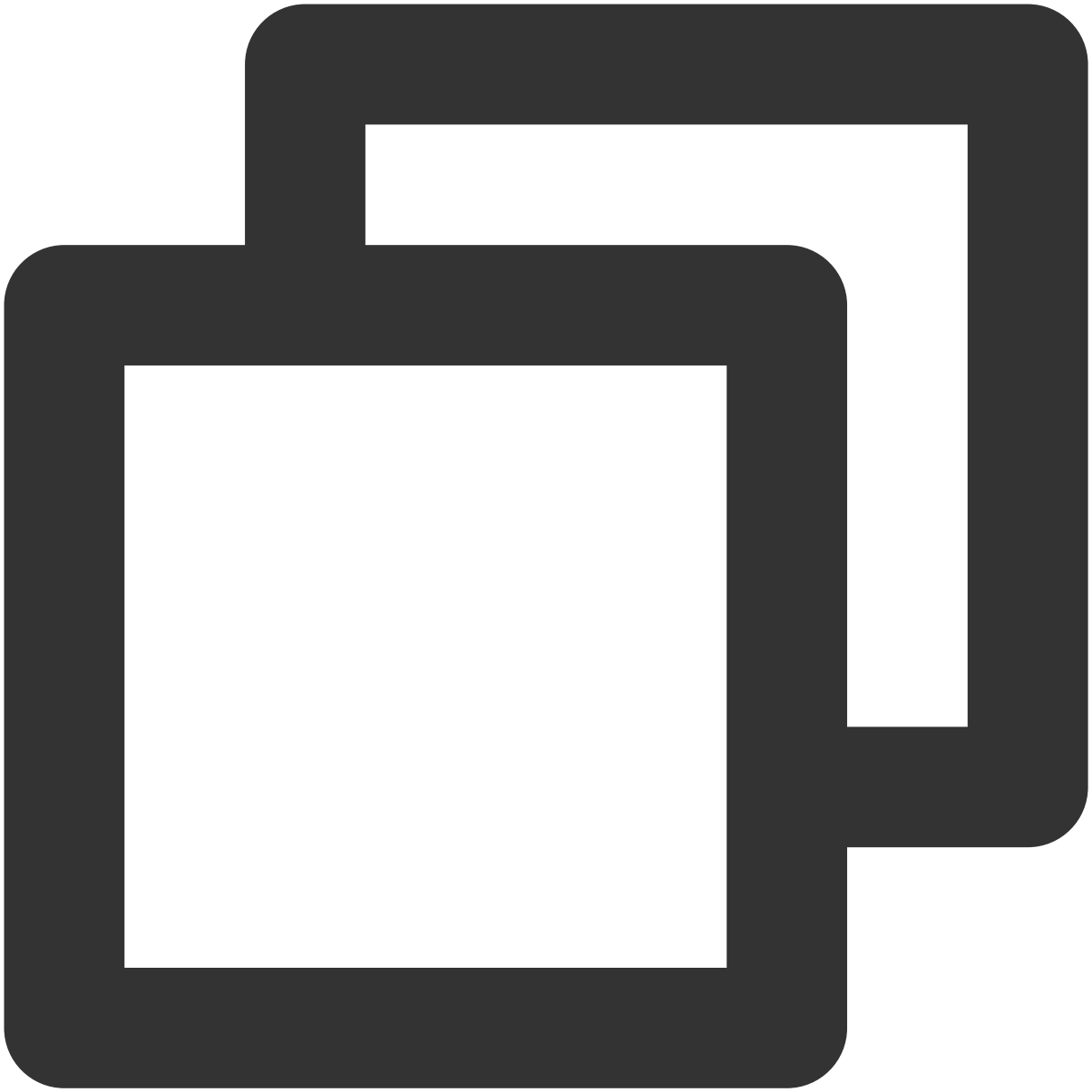
radius 是内切圆的半径，取值范围为大于0且小于原图最小边一半的整数。内切圆的圆心为图片的中心。图片格式为 gif 时，不支持该参数。

Objective-C



```
// 指定半径100为例
[transform setCutWithIRadius:100];
```

swift

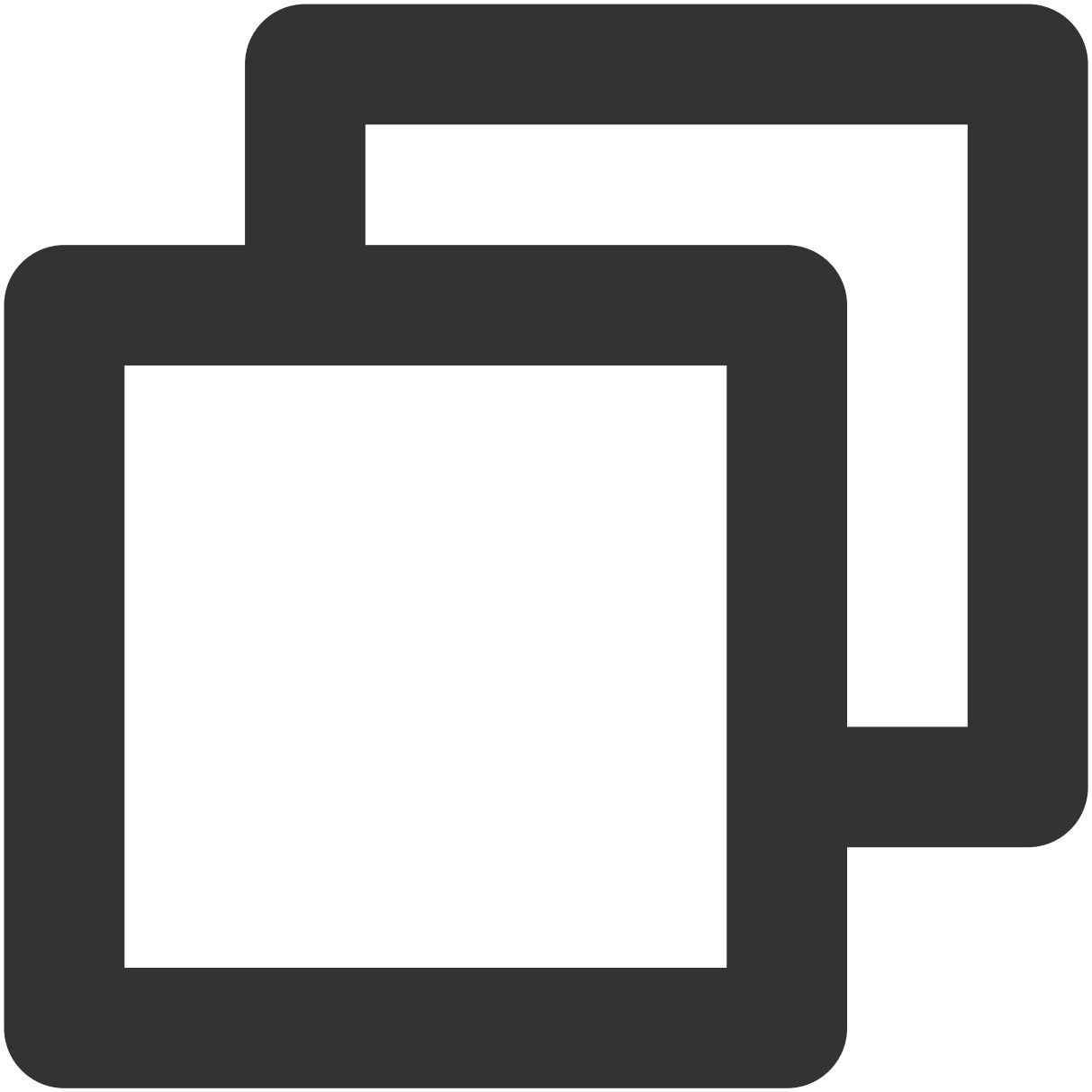


```
transform.setCutWithIRadius(100)
```

圆角裁剪

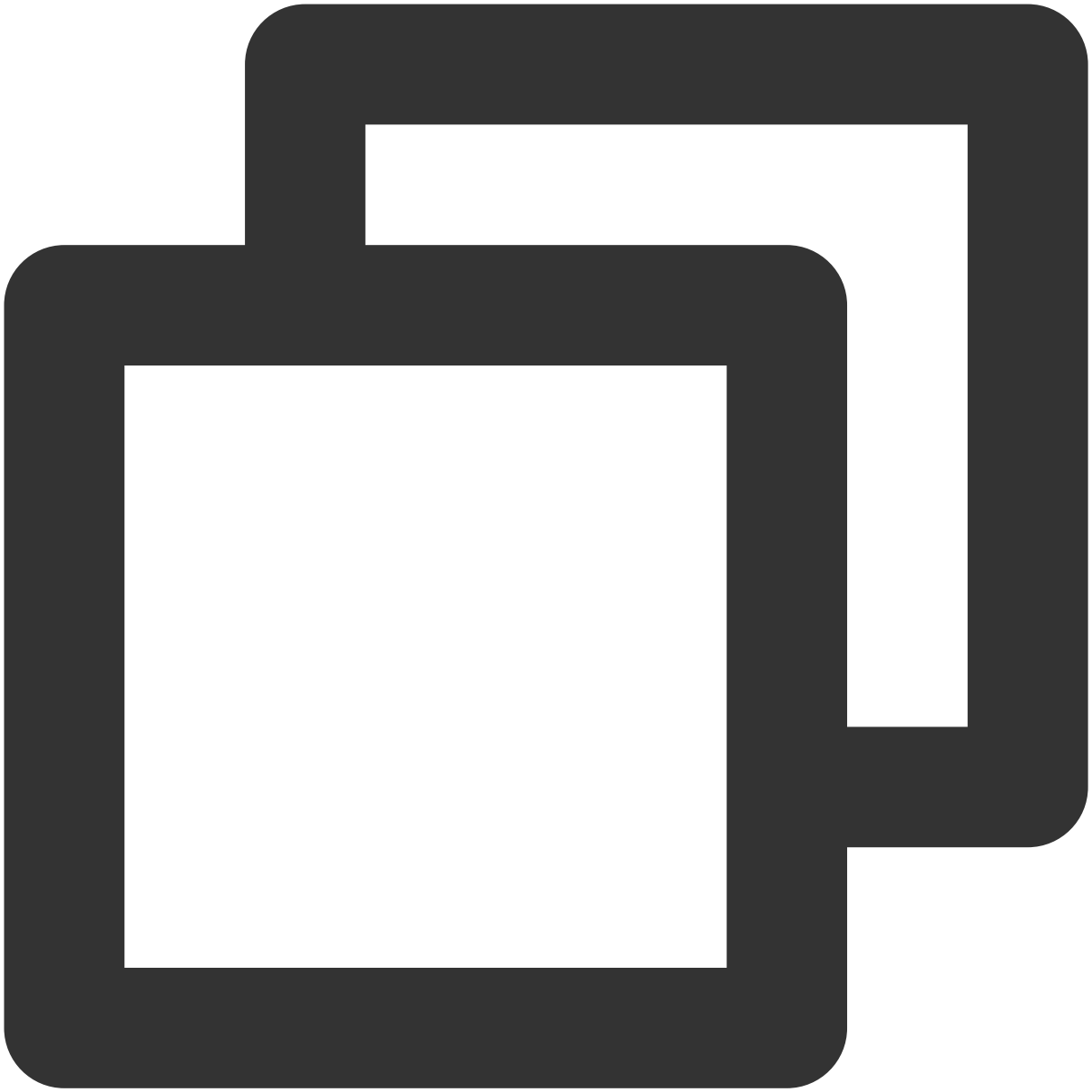
radius 为图片圆角边缘的半径，取值范围为大于0且小于原图最小边一半的整数。圆角与原图边缘相切。图片格式为 gif 时，不支持该参数。

Objective-C



```
// 指定圆角半径100为例
[transform setCutWithRRadius:100];
```

swift



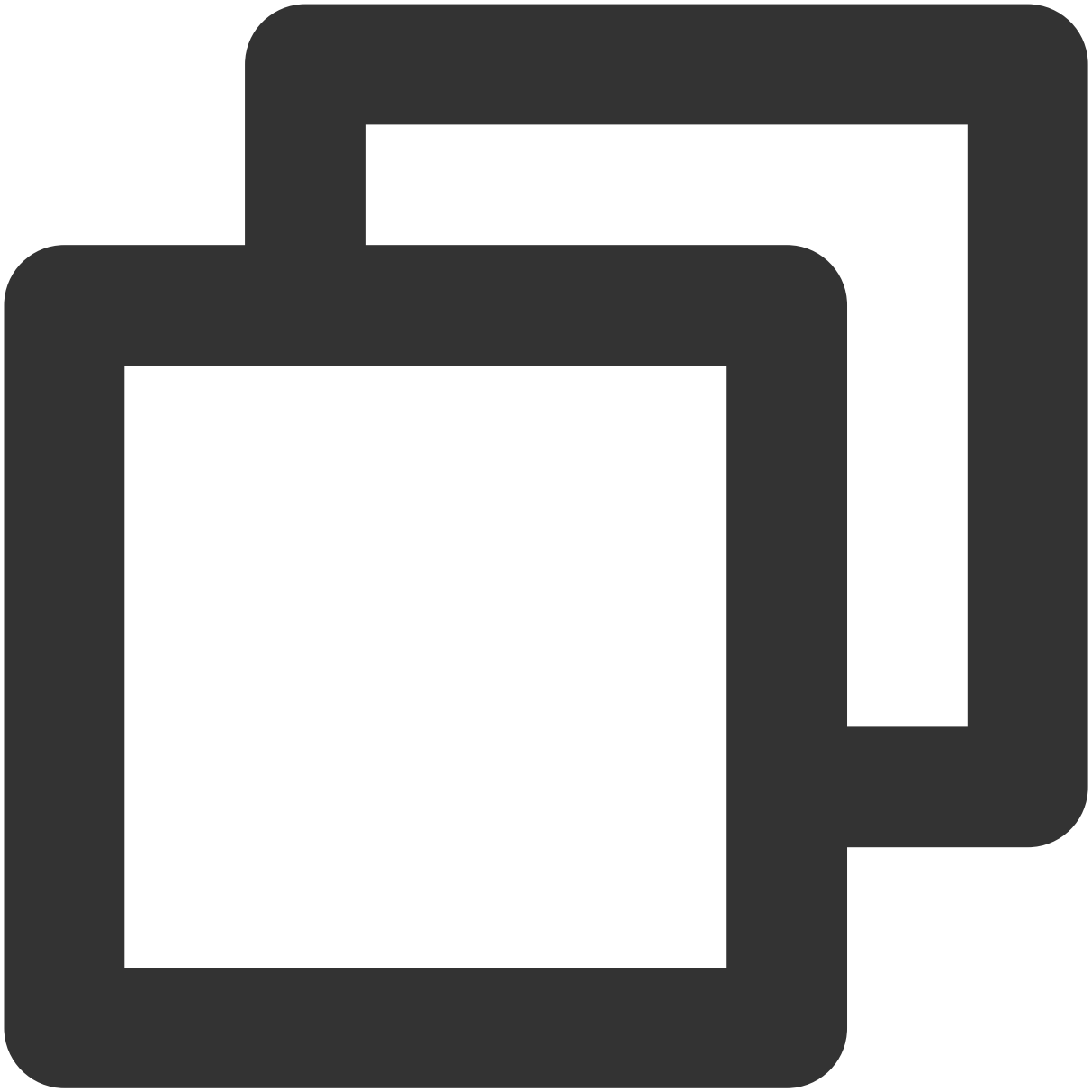
```
transform.setCutWithRRadius(100)
```

缩放裁剪

说明：

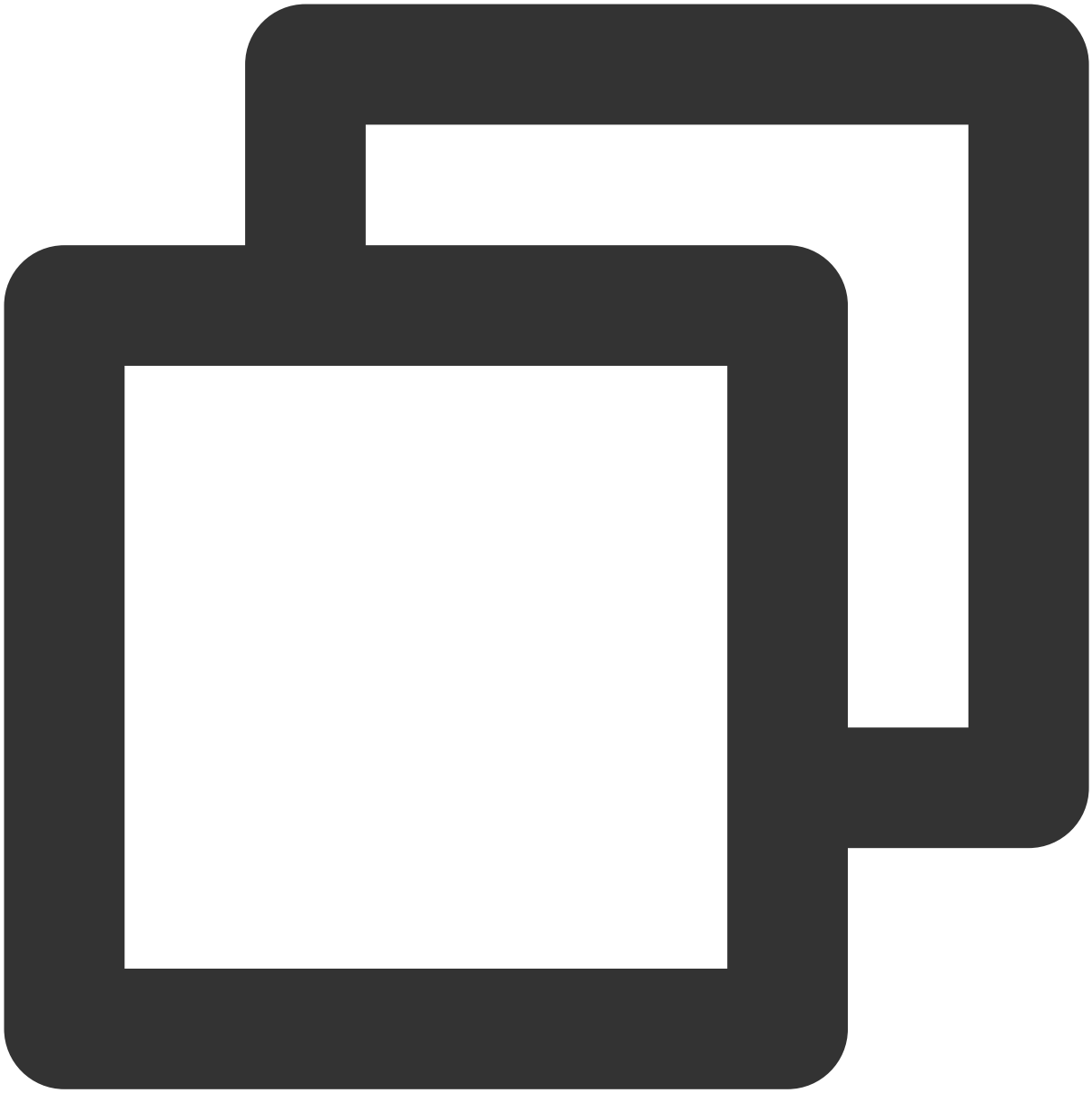
方位图说明，请参见 gravity 介绍。

Objective-C



```
// 指定宽高缩放裁剪，如果为0则不变；
[transform setCutWithCrop:100 height:100];

// 指定宽高以及 gravity 缩放裁剪；
[transform setCutWithCrop:100 height:100 gravity:CIGravityCenter];
```

swift

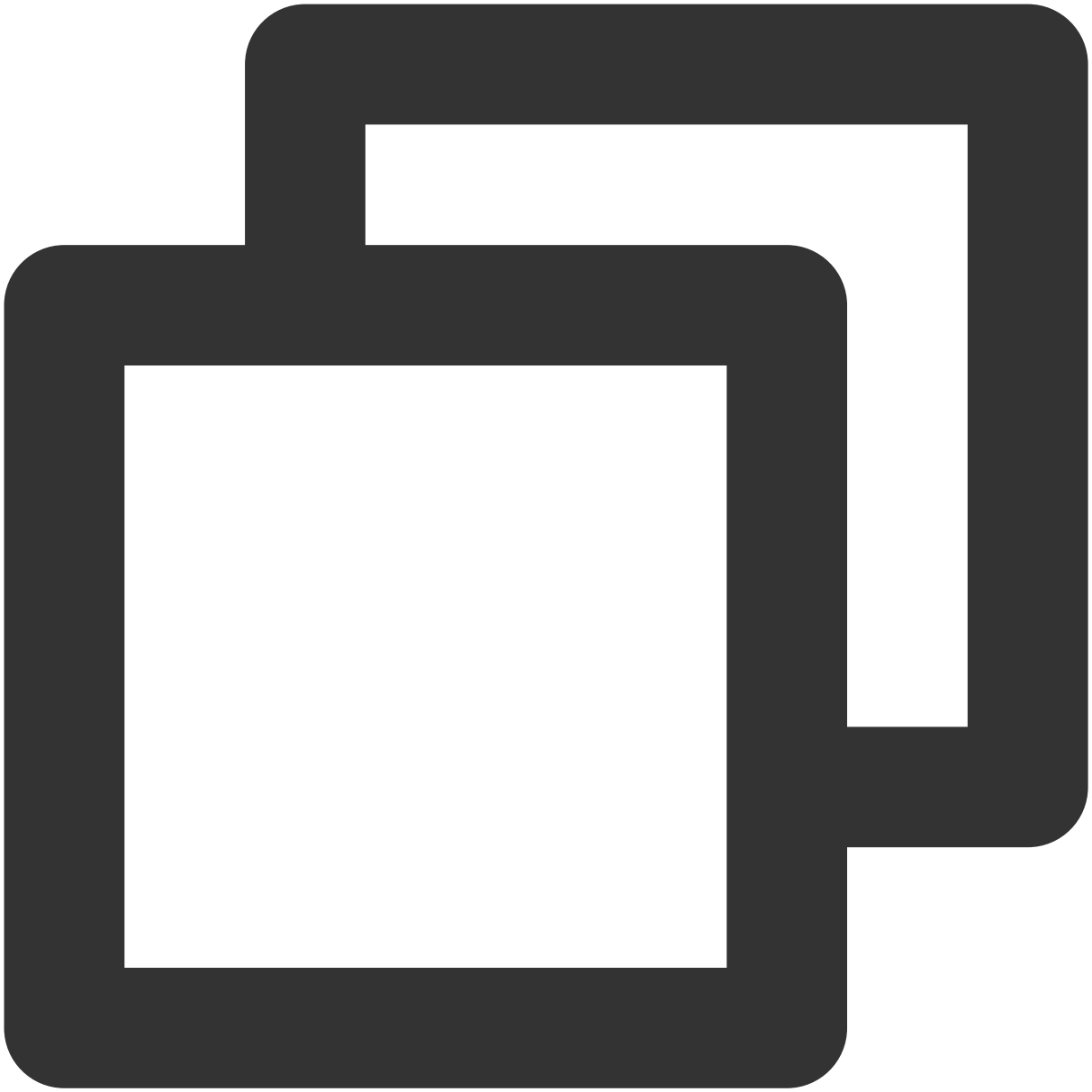


```
transform.setCutWithCrop(100, height: 100)
transform.setCutWithCrop(100, height: 100, gravity: CloudInfiniteGravity.CIGravityC
```

人脸智能裁剪

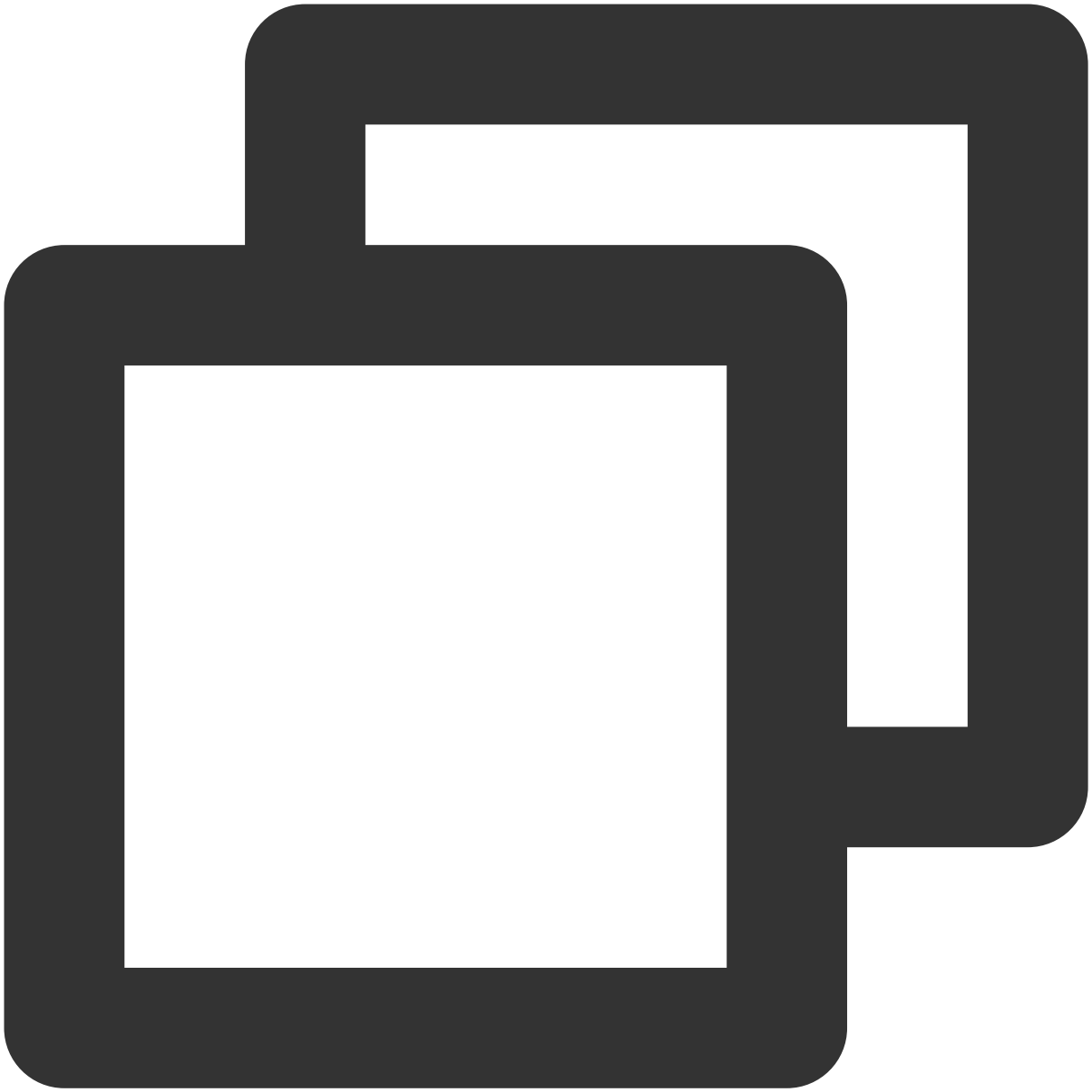
基于图片中的人脸位置进行缩放裁剪。目标图片的宽度为 Width、高度为 Height。

Objective-C



```
// 裁剪人脸并宽高指定100缩放为例；
[transform setCutWithScrop:100 height:100];
```

swift



```
transform.setCutWithScrop(100, height: 100)
```

旋转

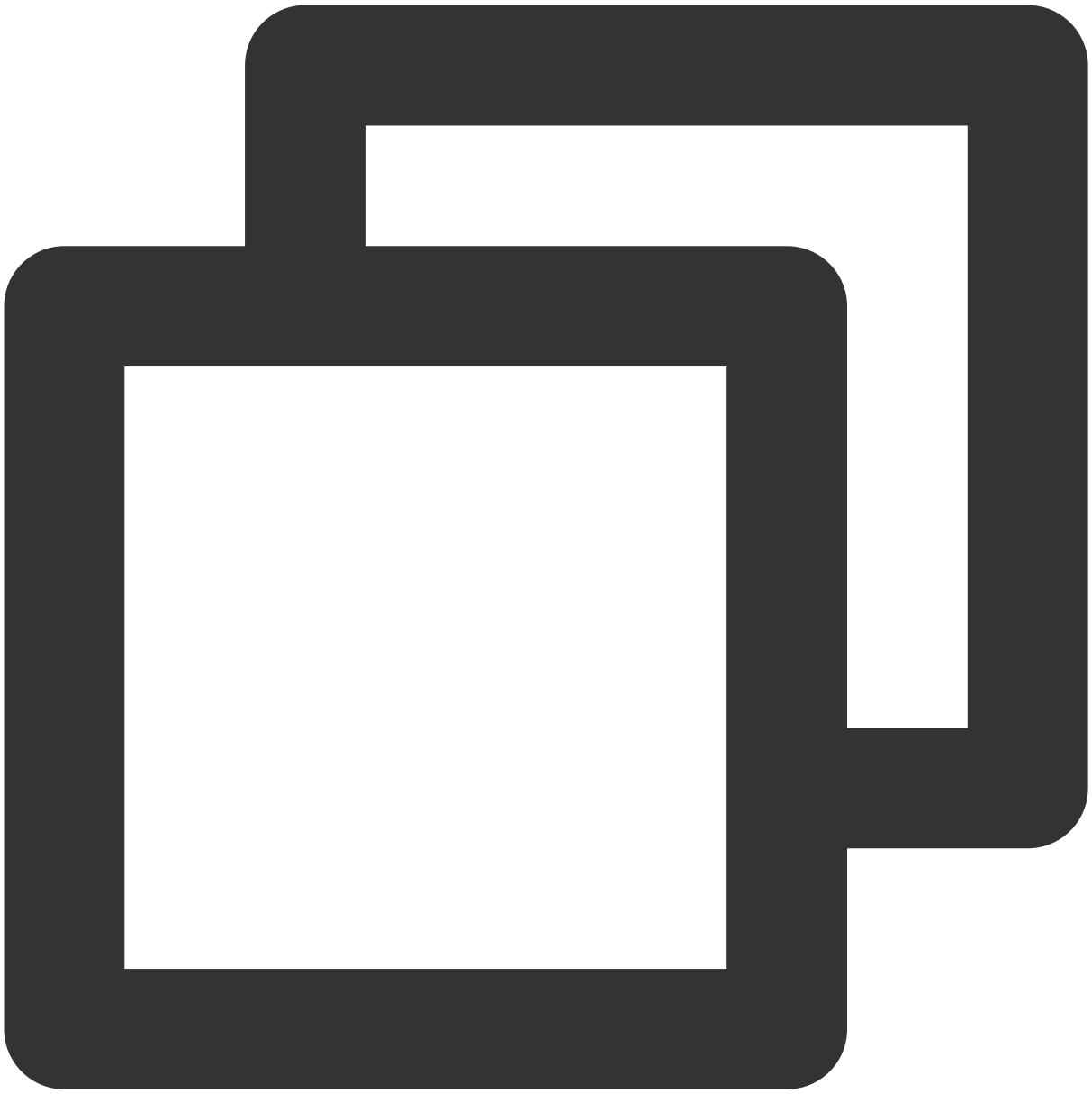
说明：

API 文档请参见 旋转功能接口。

普通旋转

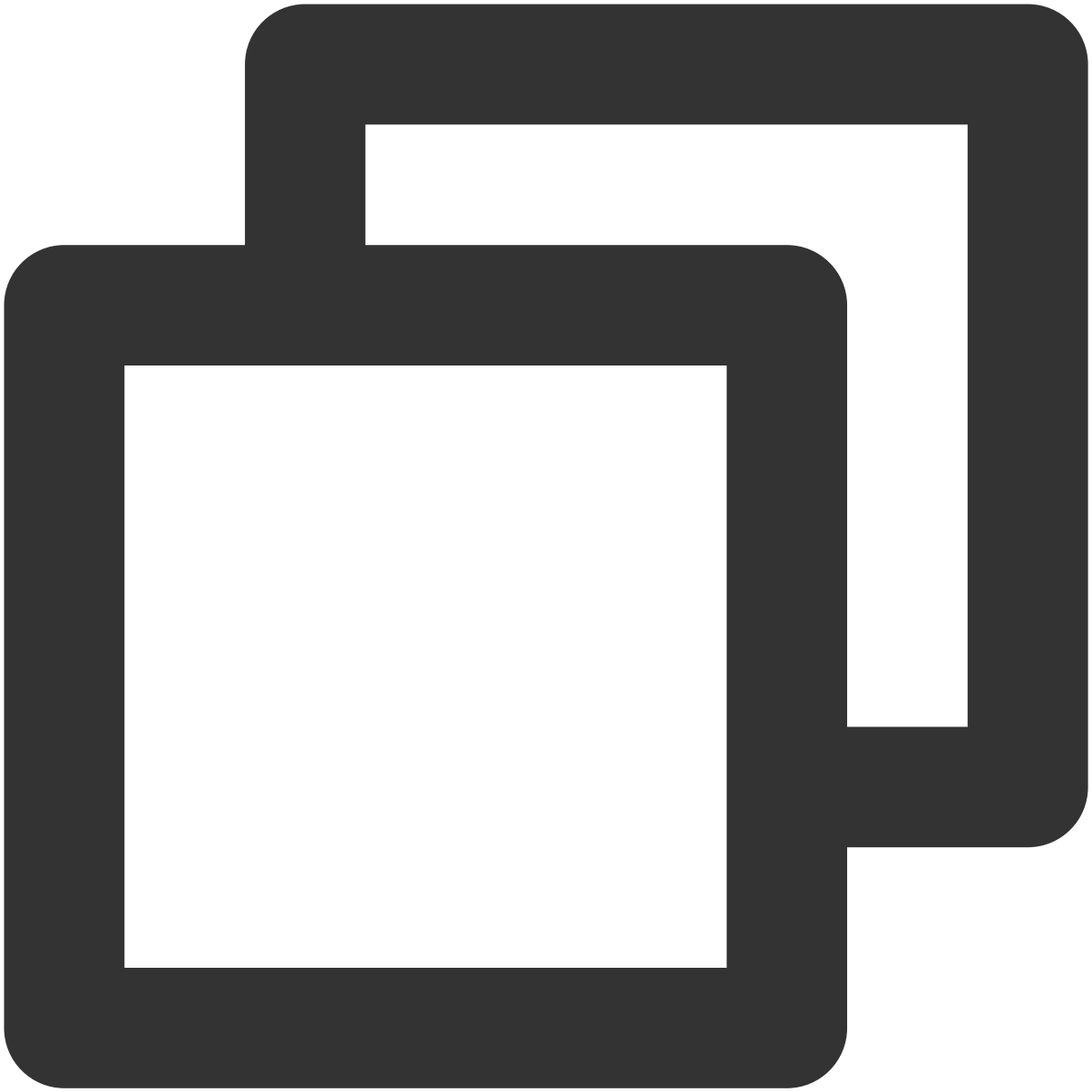
图片顺时针旋转角度，取值范围0 - 360 ，默认不旋转。

Objective-C



```
// 以旋转45度为例；
[transform setRotateWith:45];
```

swift

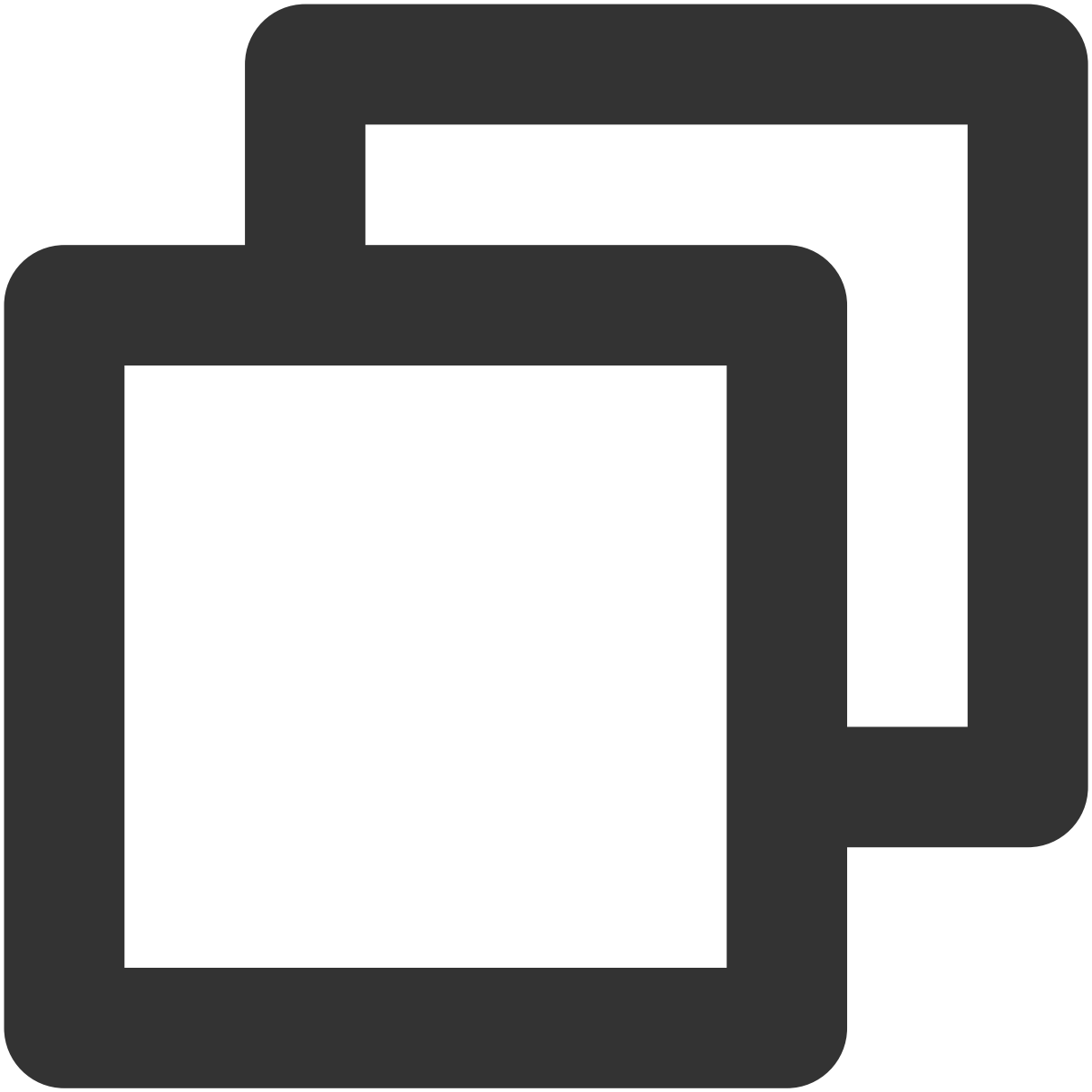


```
transform.setRotateWith(45)
```

自适应旋转

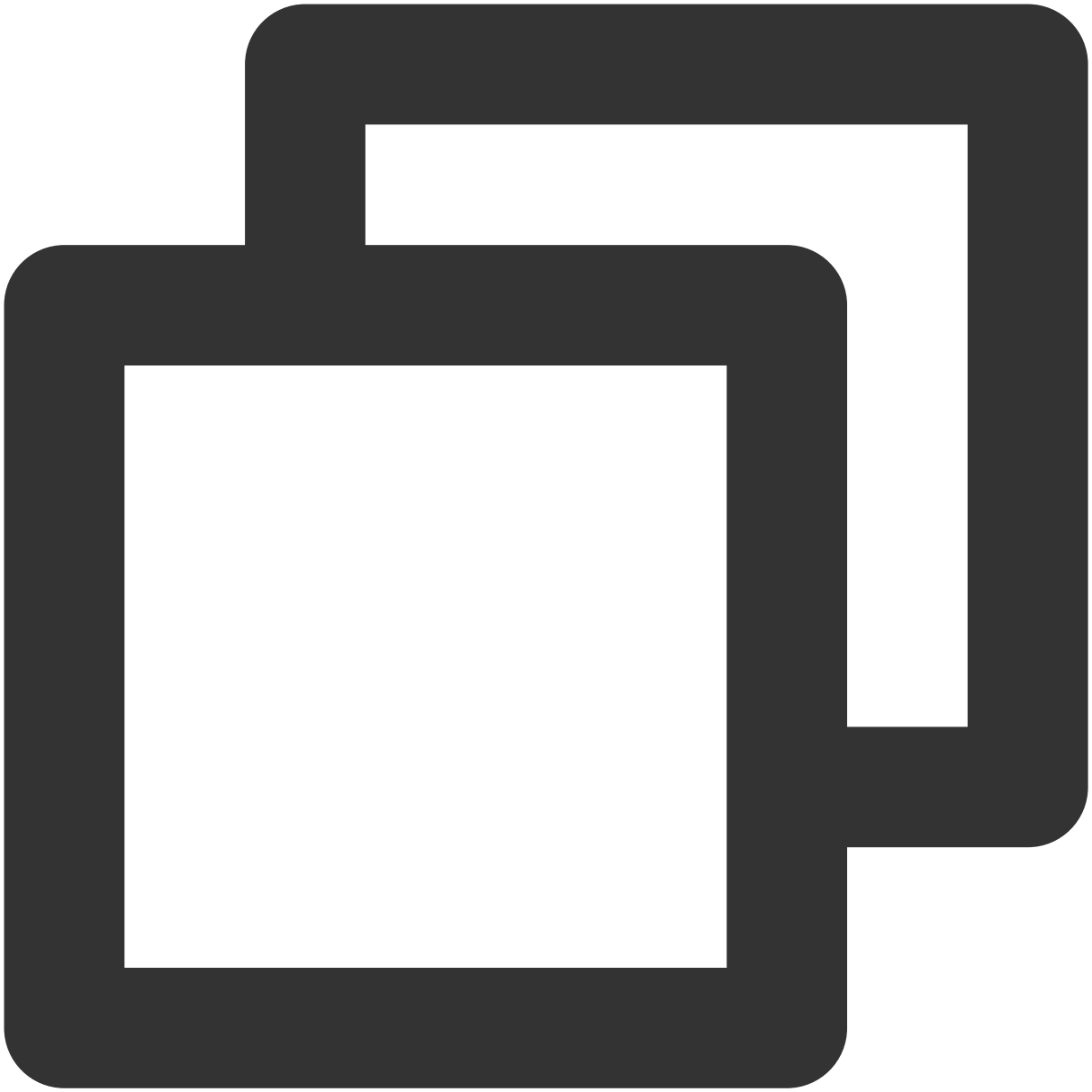
根据原图 EXIF 信息将图片自适应旋转回正。

Objective-C



```
[transform setRotateAutoOrient];
```

swift



```
transform.setRotateAutoOrient()
```

格式转换

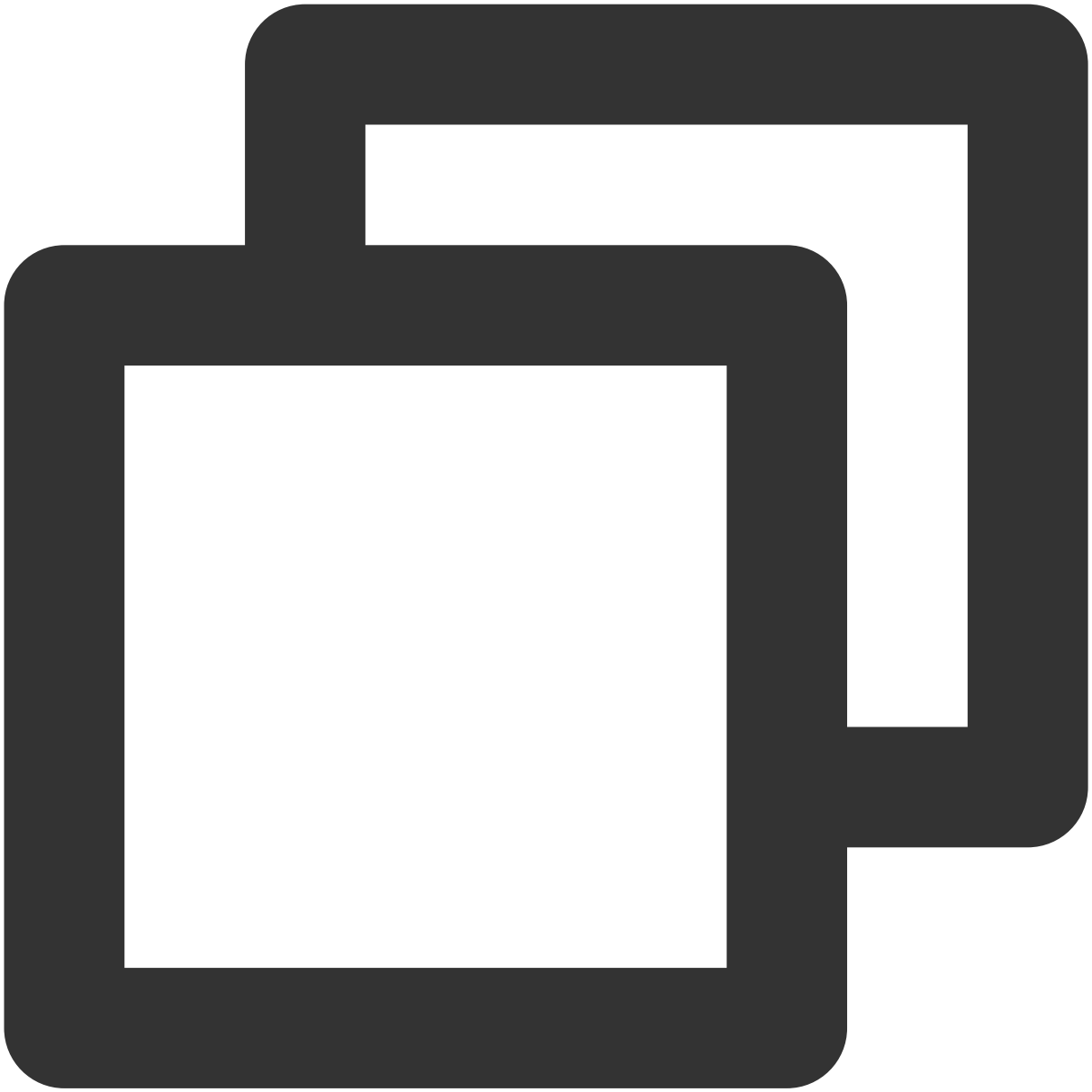
说明：

API 文档请参见 格式转换接口。

格式转换

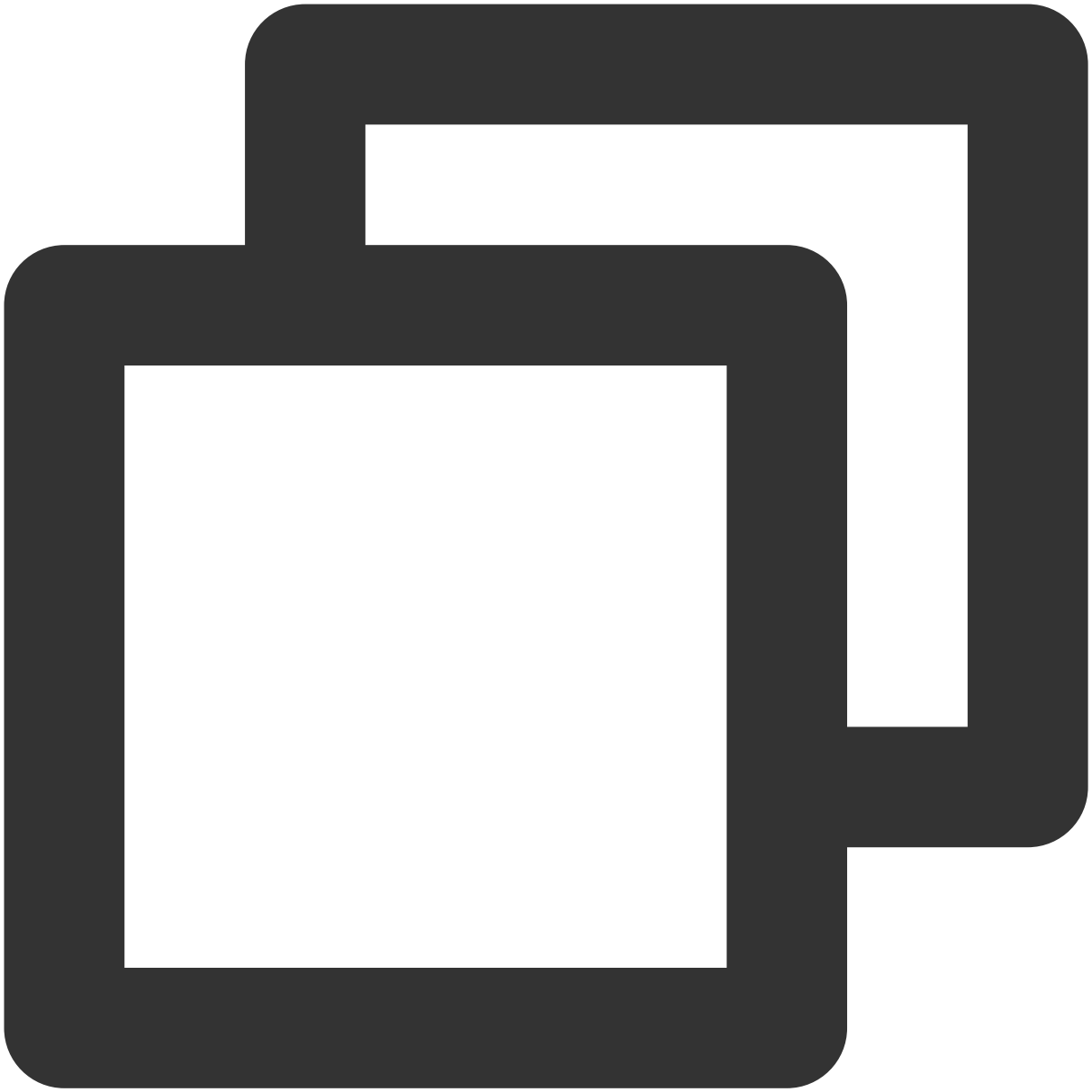
目标缩略图的图片格式可为：TPG，JPG，BMP，GIF，PNG，HEIC，WEBP，YJPEG，AVIF 等，其中 YJPEG 为数据万象针对 JPEG 格式进行的优化，本质为 JPG 格式；缺省为原图格式。

Objective-C



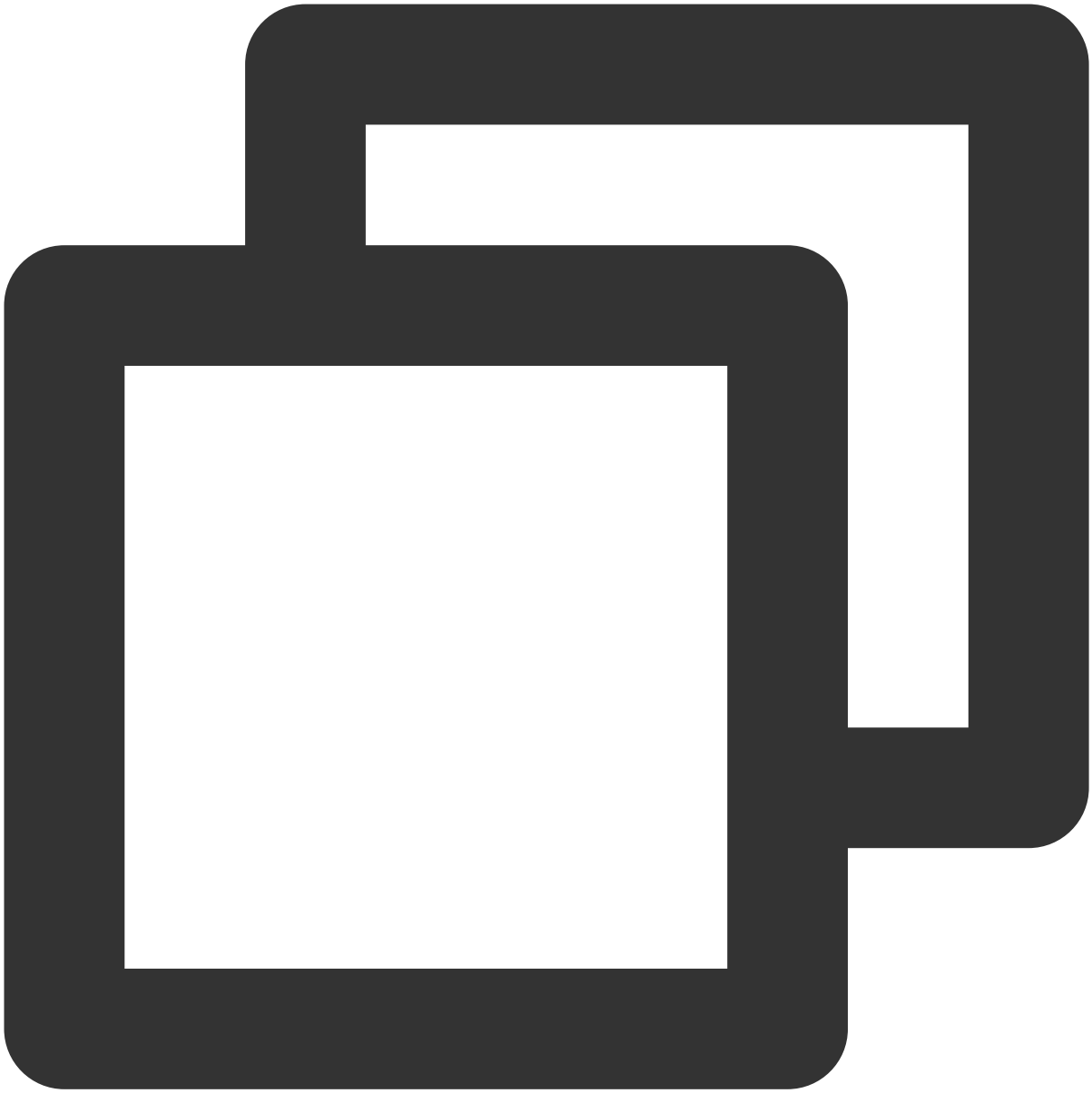
```
// 以转换为 JPG 为例
[transform setFormatWith:CIImageTypeJPG];

// 指定传参方式：
[transform setFormatWith:CIImageTypeTPG options:CILoadTypeUrlFooter];
```

swift



```
transform.setFormatWith(CIImageFormat.typeJPG);
transform.setFormatWith(CIImageFormat.typeJPG, options: CILoadTypeEnum.urlFooter);
```

CILoadTypeEnum



```
// 加载类型 方式一：带 accept 头部 accept:image/ ***
CILoadTypeAcceptHeader = 0,

// 加载类型 方式二：在 url 后面中拼接 imageMogr2/format/ ***
// 如果需要方式二，则使用该值；不传默认为第一种方式
CILoadTypeUrlFooter,
```

说明：

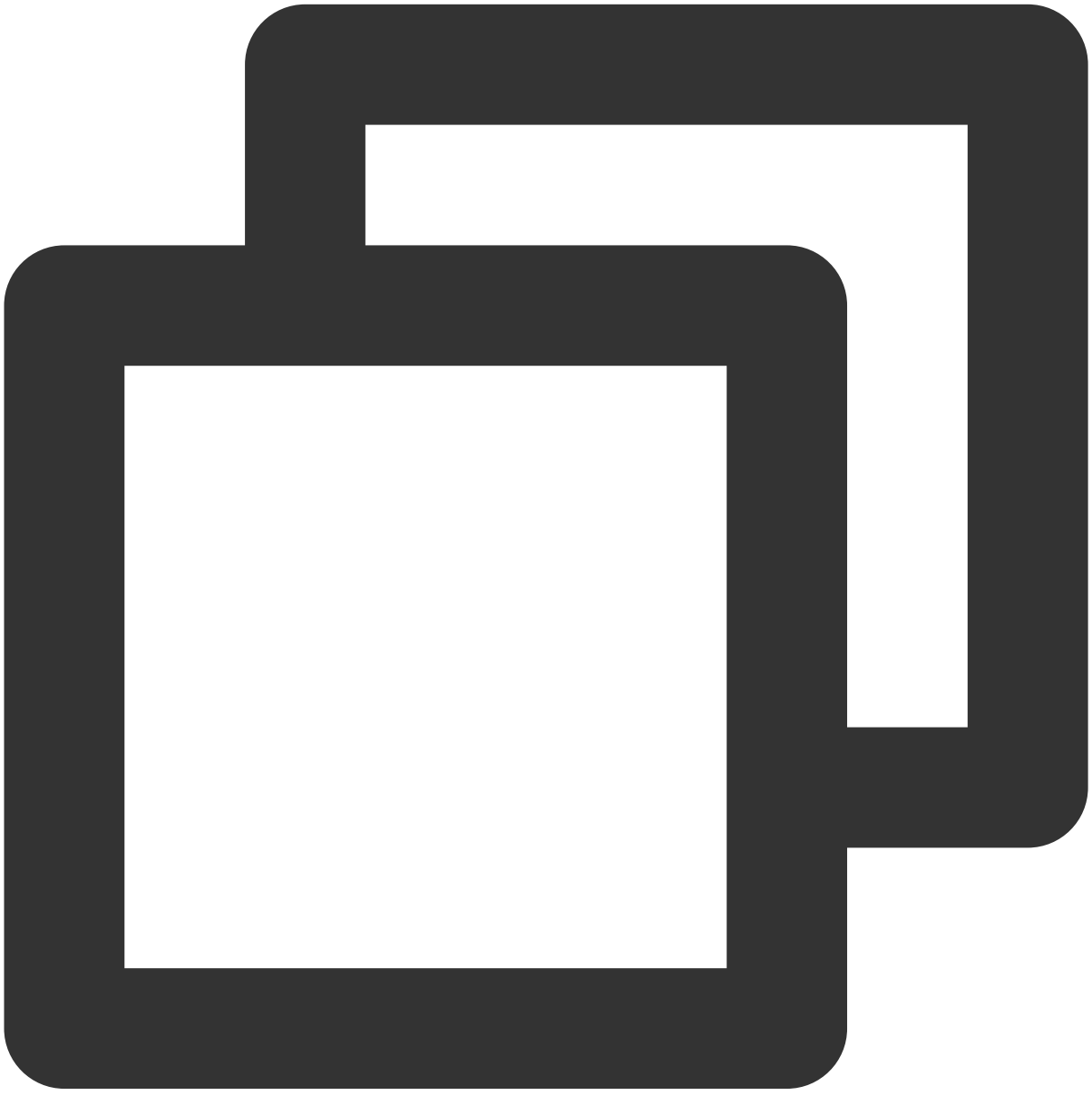
使用 HEIC 格式，需要在 iOS11及以上；不支持 GIF 格式图片转为 HEIC。

当指定为 CILoadTypeAcceptHeader 方式传参时，并且组合了其他的转换则 header 失效，并且在 sdk 内部自动转换为 footer 的方式。

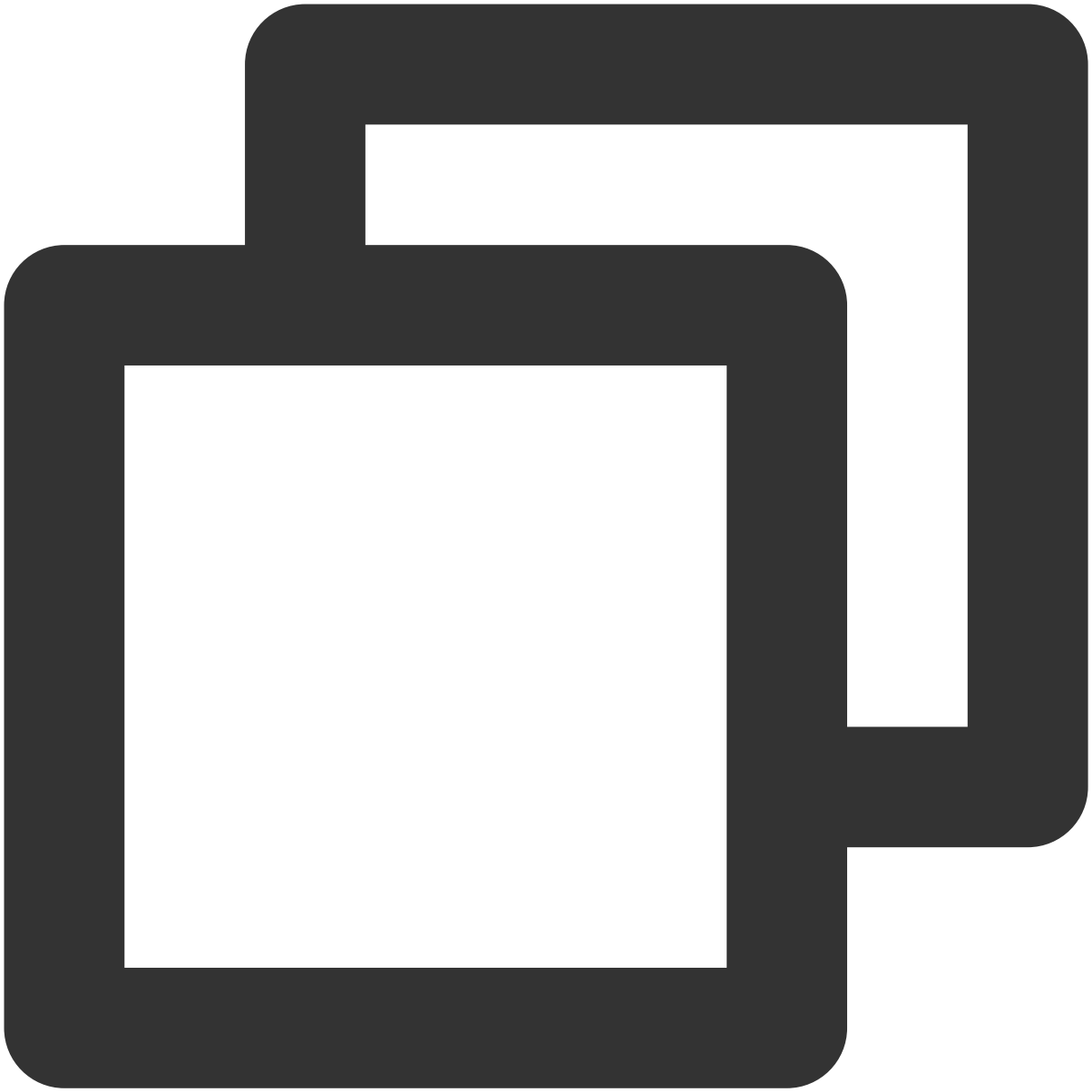
在使用图片格式转换时，如果需要转为 TPG 格式，则需要依赖 `CloudInfinite/TPG` 模块。

在使用图片格式转换时，如果需要转为 AVIF 格式，则需要依赖 `CloudInfinite/AVIF` 模块。

在使用图片格式转换时，如果需要转为WEBP格式，则需要依赖 `SDWebImageWebPCoder` 库。



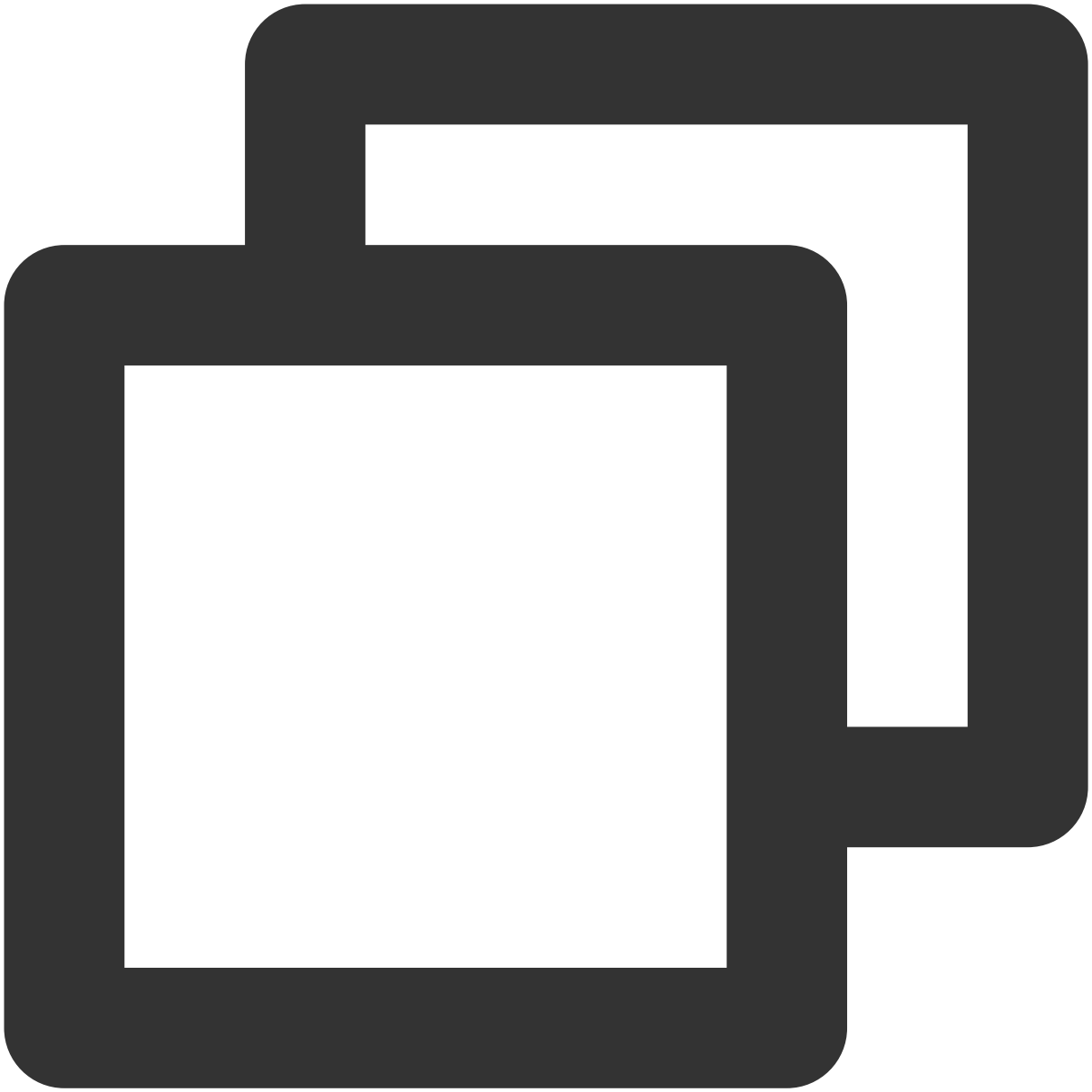
```
pod 'CloudInfinite/TPG'
```



```
pod 'CloudInfinite/AVIF'
```

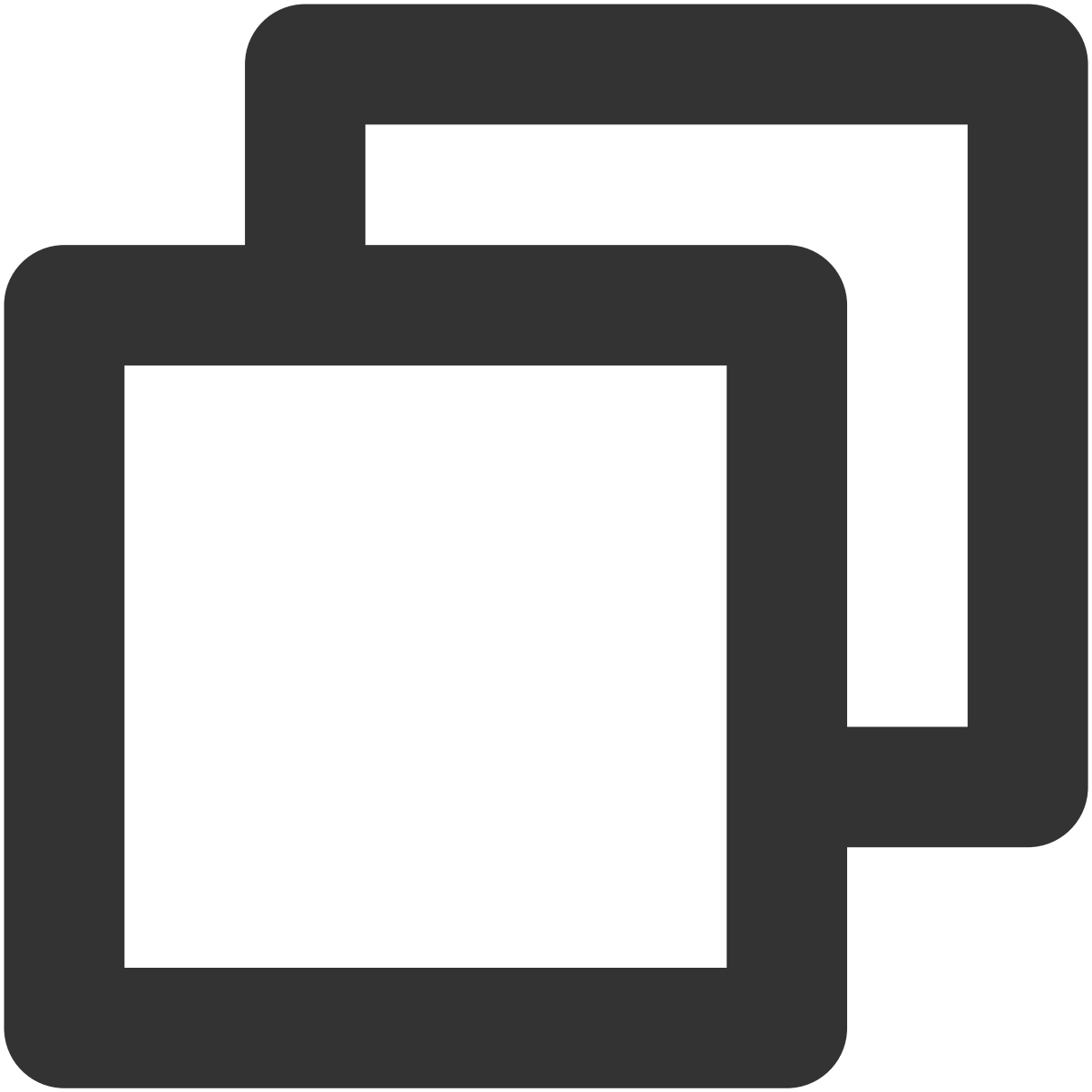
gif 格式优化

只针对原图为 GIF 格式，对 GIF 图片格式进行的优化，降帧降颜色。



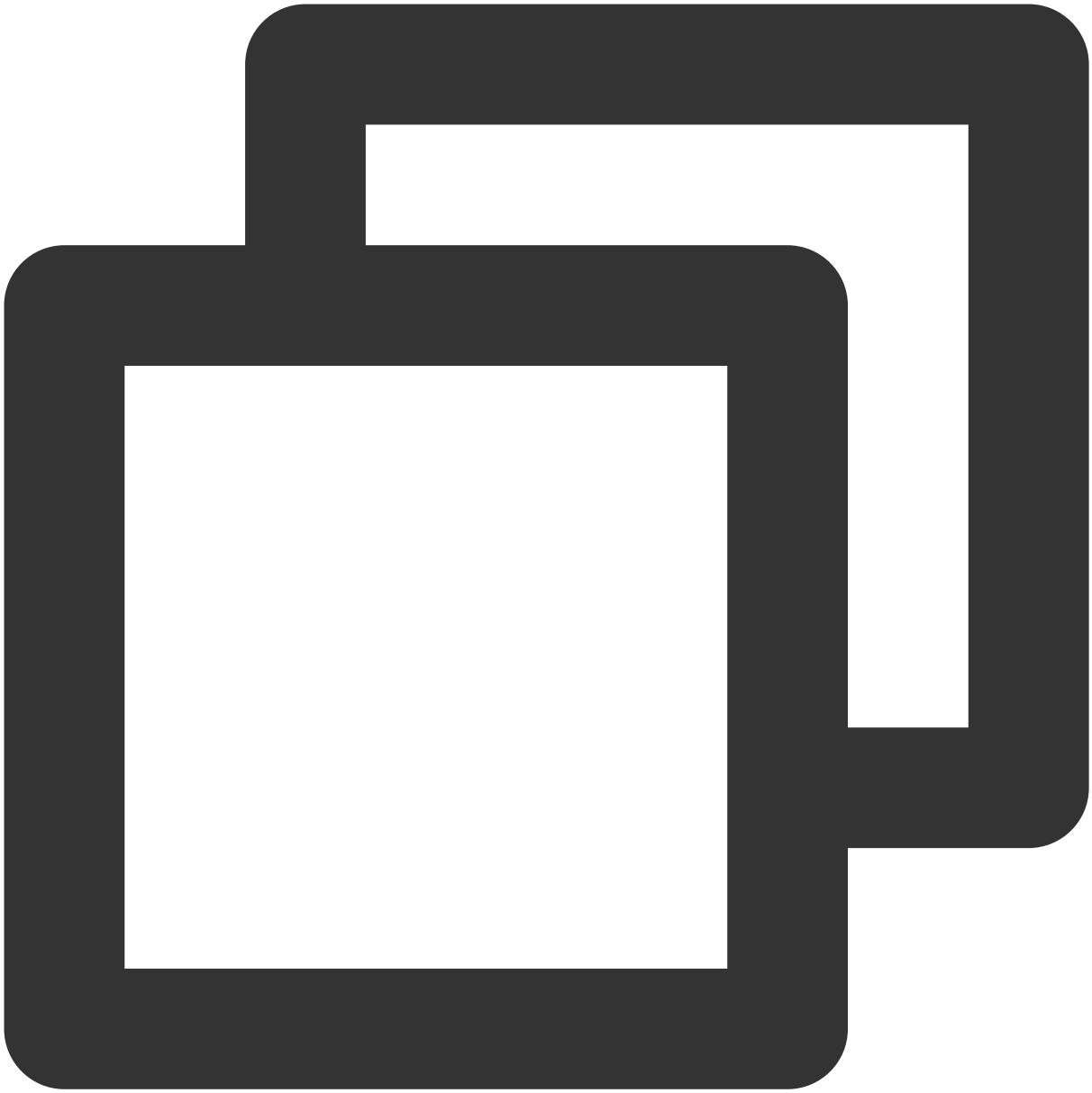
```
FrameNumber=1，则按照默认帧数30处理，如果图片帧数大于该帧数则截取。
FrameNumber 取值( 1,100 ]，则将图片压缩到指定帧数 （FrameNumber）。
```

Objective-C



```
[transform setCgif:50];
```

swift

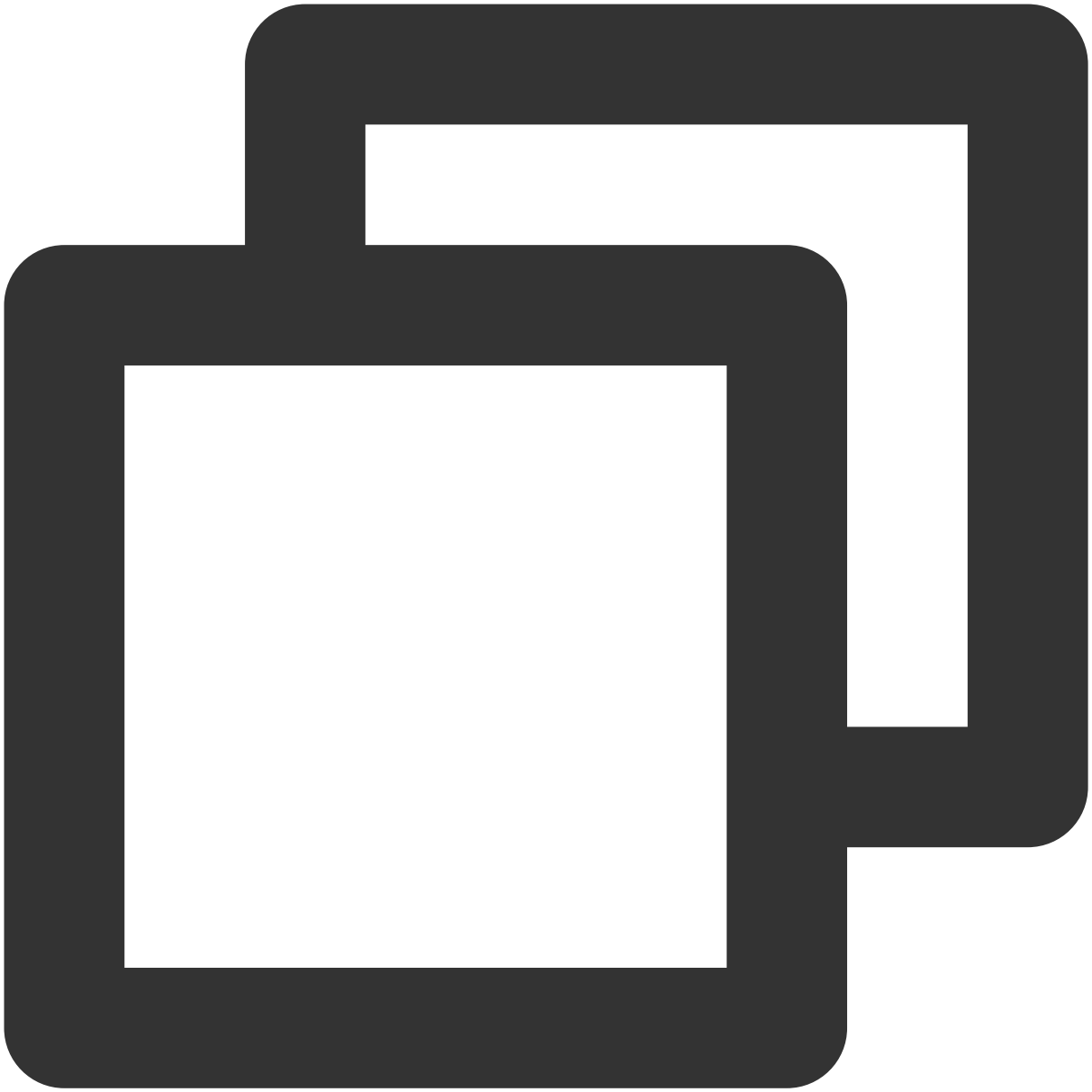


```
transform.setCgif(50)
```

输出渐进式 JPG 格式

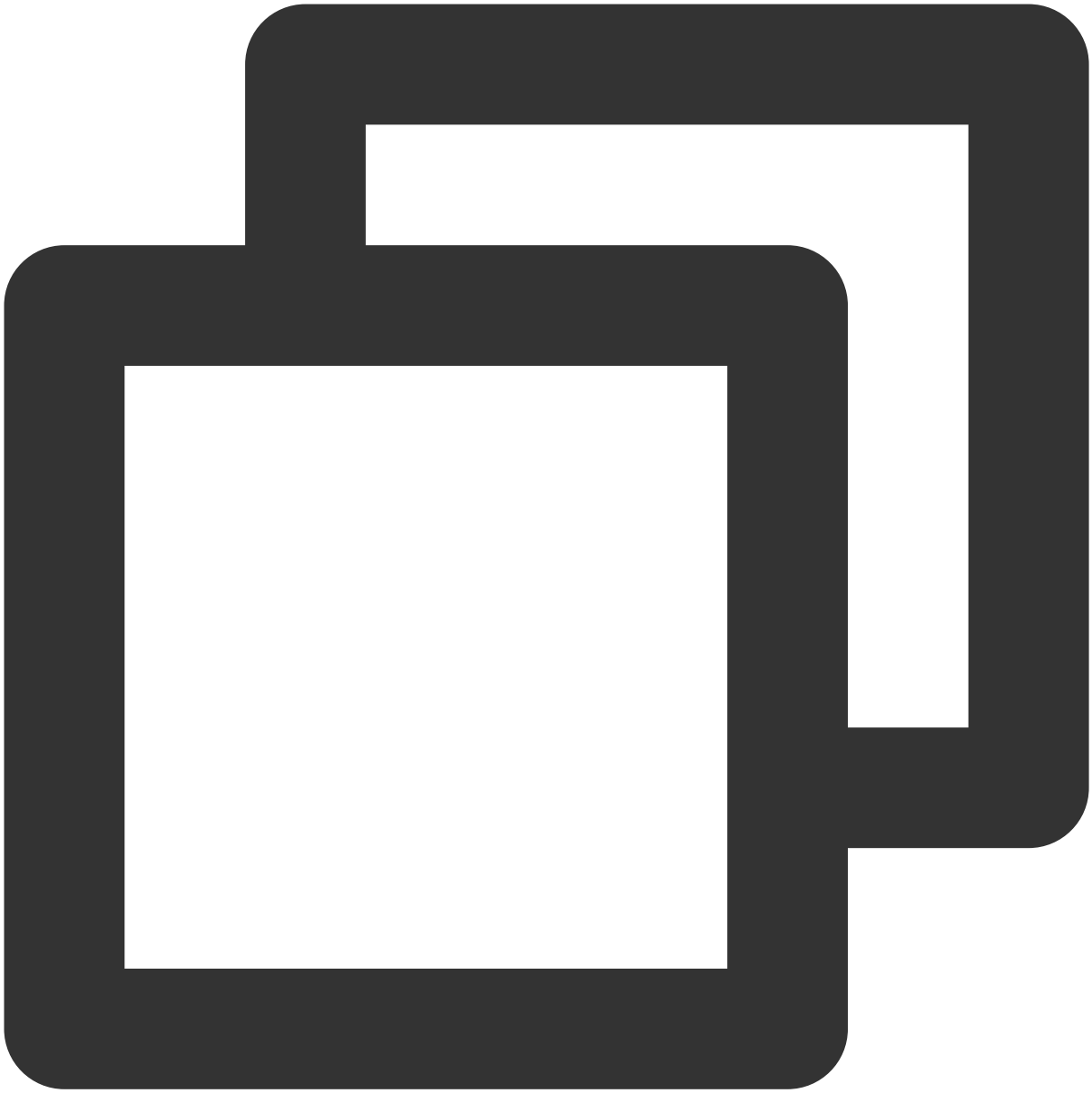
Mode 可为0或1。0表示不开启渐进式；1表示开启渐进式。该参数仅在输出图片格式为 JPG 格式时有效。如果输出非 JPG 图片格式，会忽略该参数，默认值0。

Objective



```
[transform setInterlace:YES];
```

swift

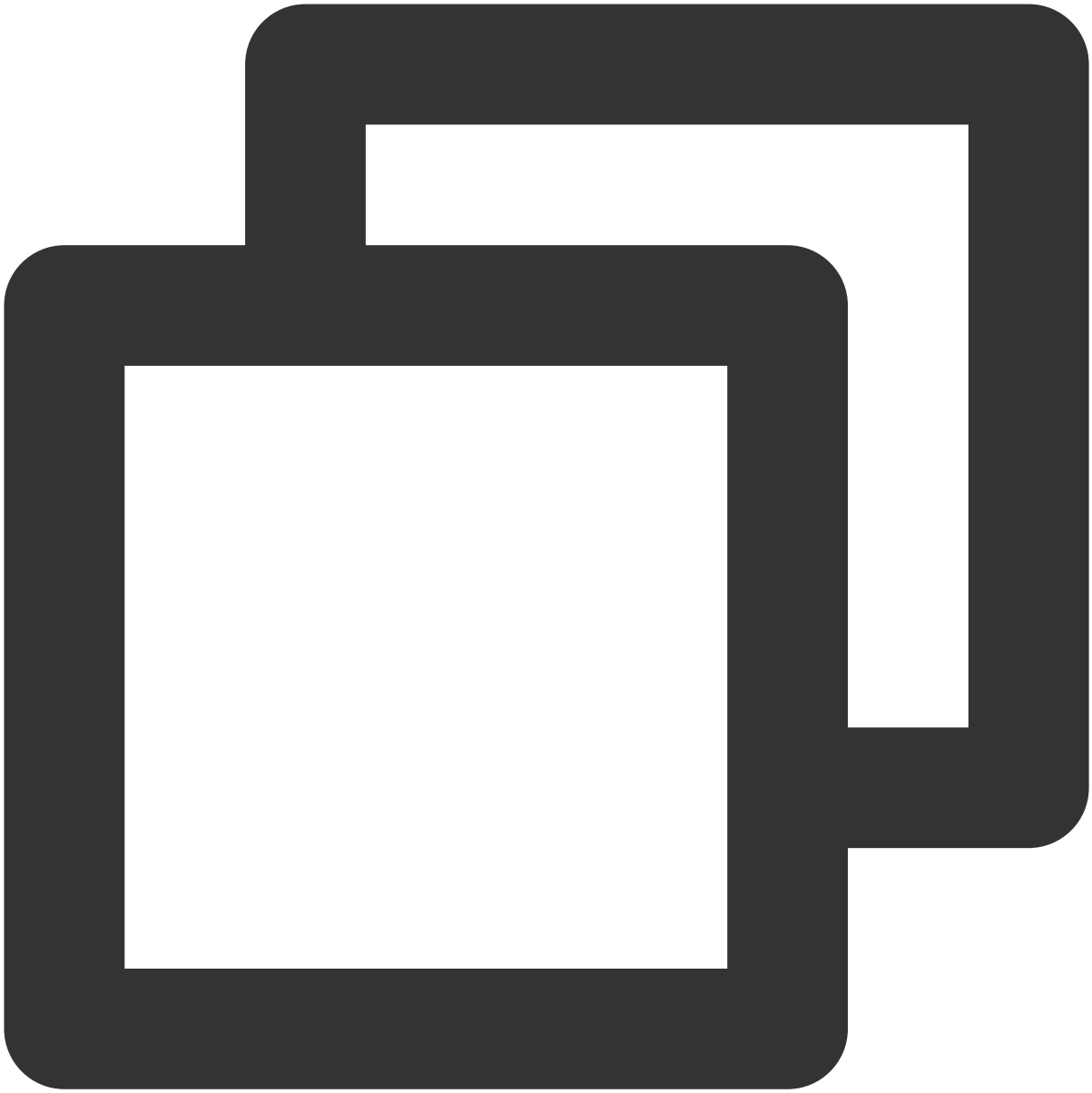


```
transform.setInterlace(true)
```

质量变换

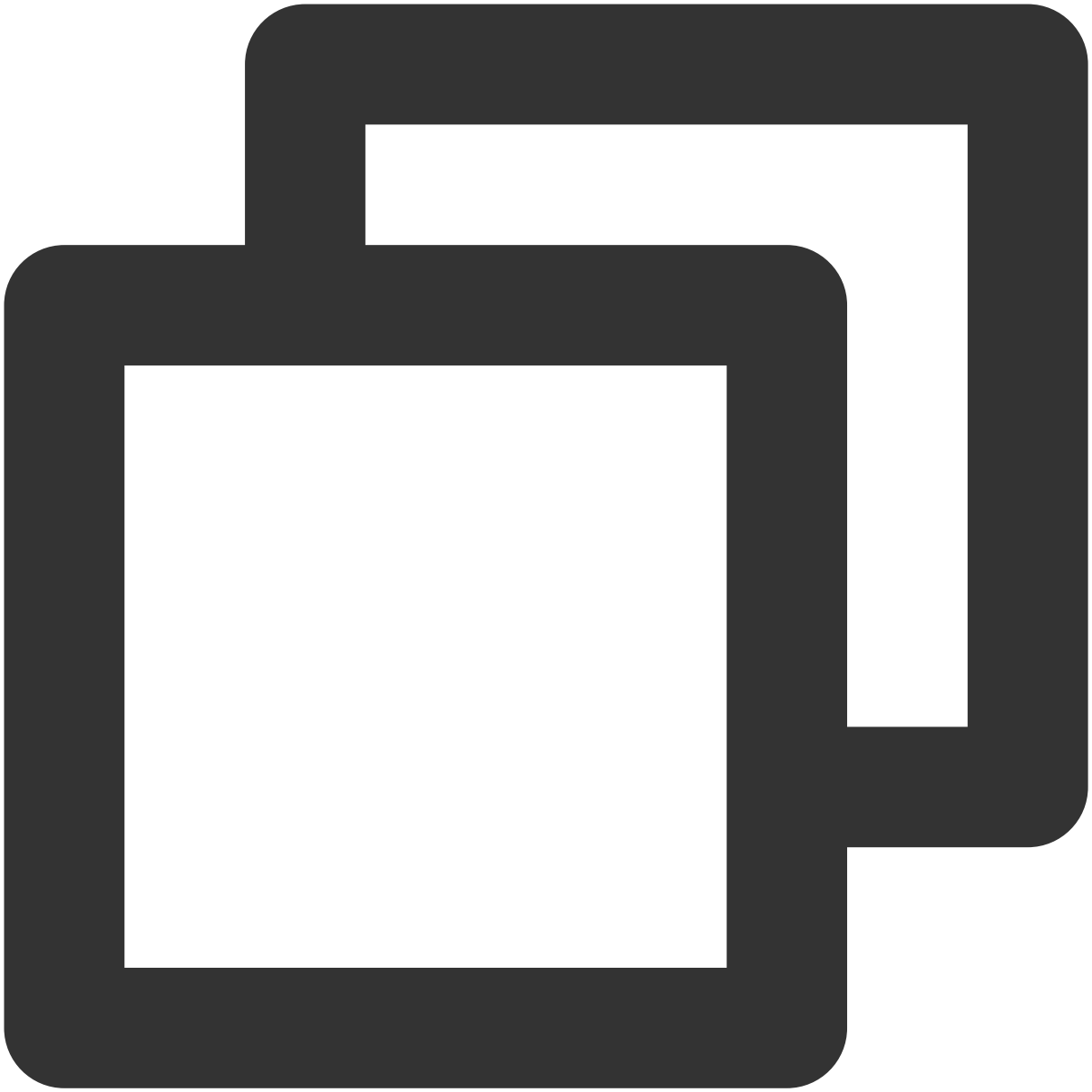
调节图片质量，API 文档请参见 质量变换接口。

Objective-C



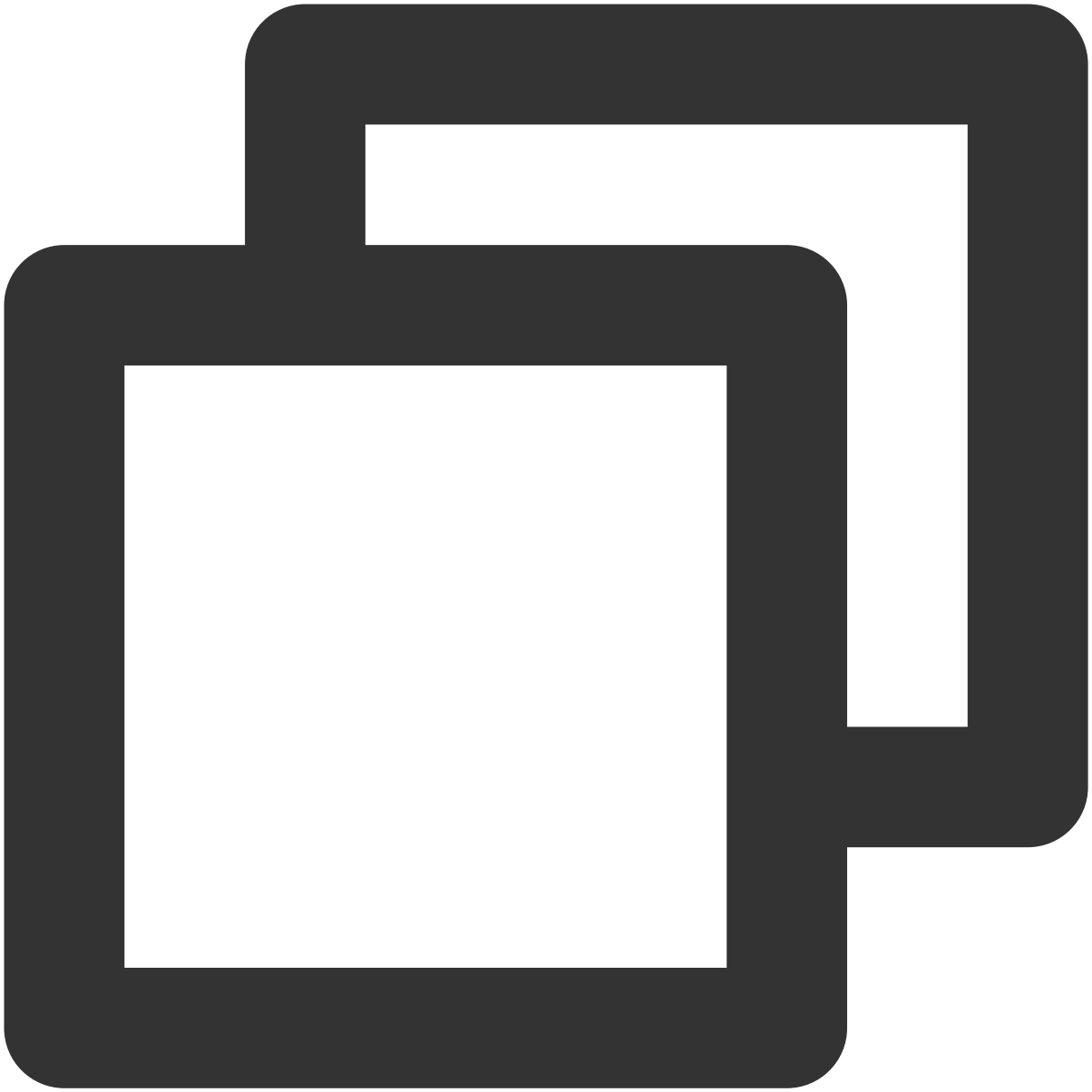
```
// 以图片的绝对质量变换以60为例
// type 为变换类型，下面具体介绍
[transform setQualityWithQuality:60 type:CIQualityChangeAbsolute];
```

swift



```
transform.setQualityWithQuality(60, type: CIQualityChangeEnum.absolute)
```

数据万象提供三种变换类型：绝对变换、相对变换、最低质量变换，其中绝对变换分为强制指定和不指定两种，如下：



```
// 图片的绝对质量，取值范围0 - 100 ，默认值为原图质量；
CIQualityChangeAbsolute = 1,

// 图片的绝对质量，取值范围0 - 100 ，强制使用指定值，例如：90!。
CIQualityChangeAbsoluteFix,

// 图片的相对质量，取值范围0 - 100 ，数值以原图质量为标准。例如原图质量为80，将 rquality 设置为
CIQualityChangeRelative,

// 图片的最低质量，取值范围0 - 100 ，设置结果图的质量参数最小值。
// 例如原图质量为85，将 lquality 设置为80后，处理结果图的图片质量为85。
```

```
// 例如原图质量为60，将 lquality 设置为80后，处理结果图的图片质量会被提升至80。
CIQualityChangeLowest,
```

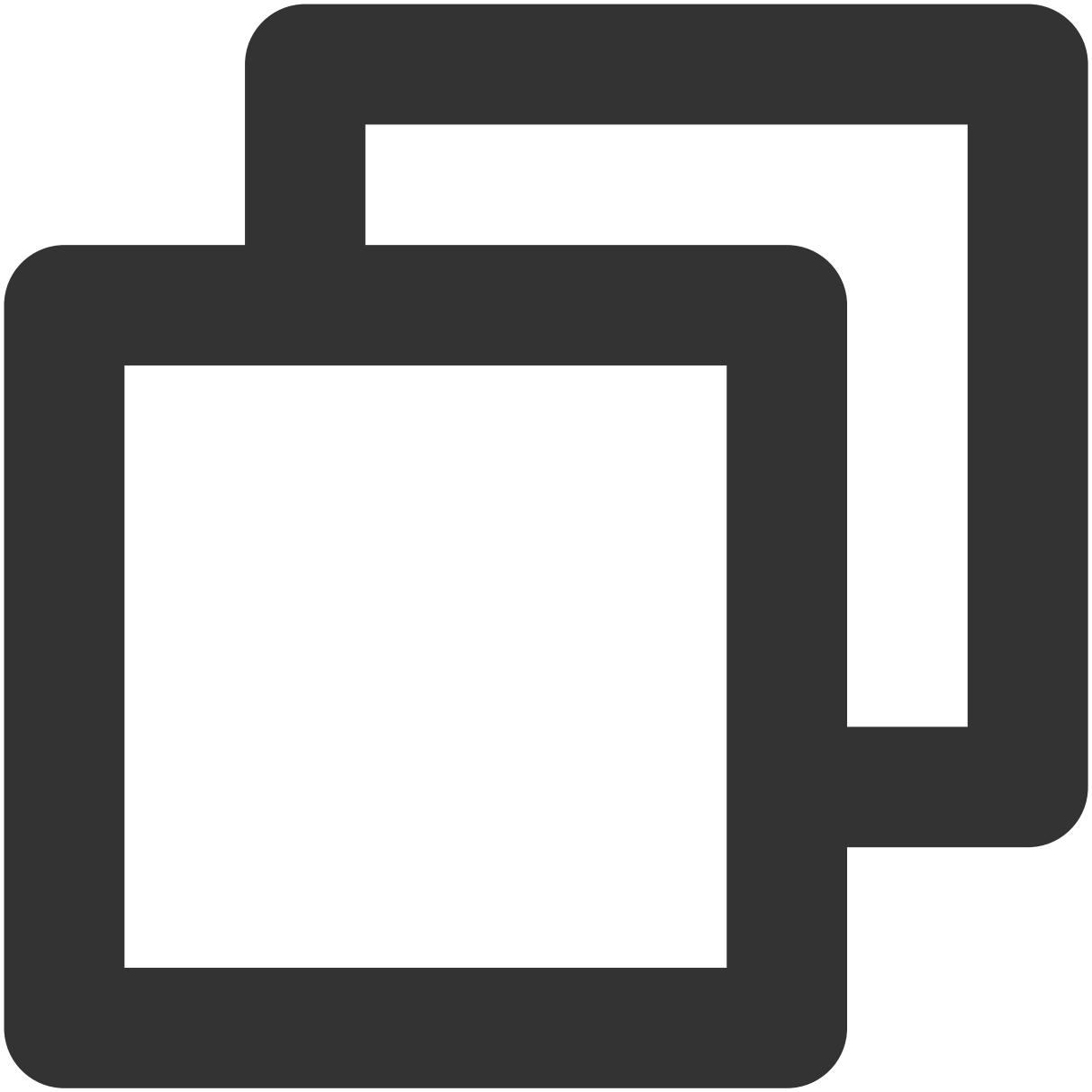
说明：

仅适用于 JPG、WEBP 格式的图片。

高斯模糊

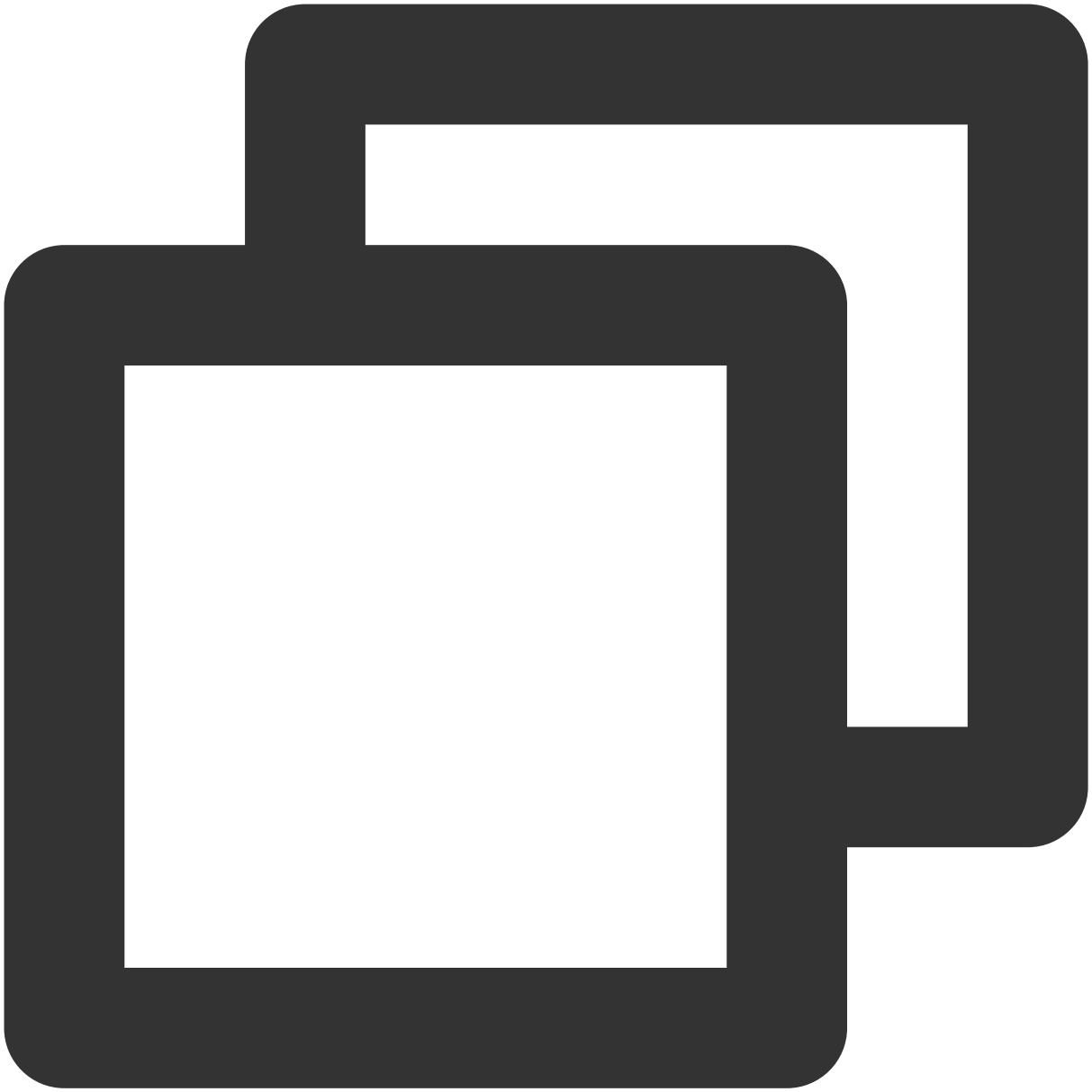
高斯模糊功能，模糊半径，取值范围为1 - 50；正态分布的标准差，必须大于0。API 文档请参见 高斯模糊接口。

Objective-C



```
// 模糊半径20，正态分布的标准差以20为例
[transform setBlurRadius:20 sigma:20];
```

swift



```
transform.setBlurRadius(20, sigma: 20)
```

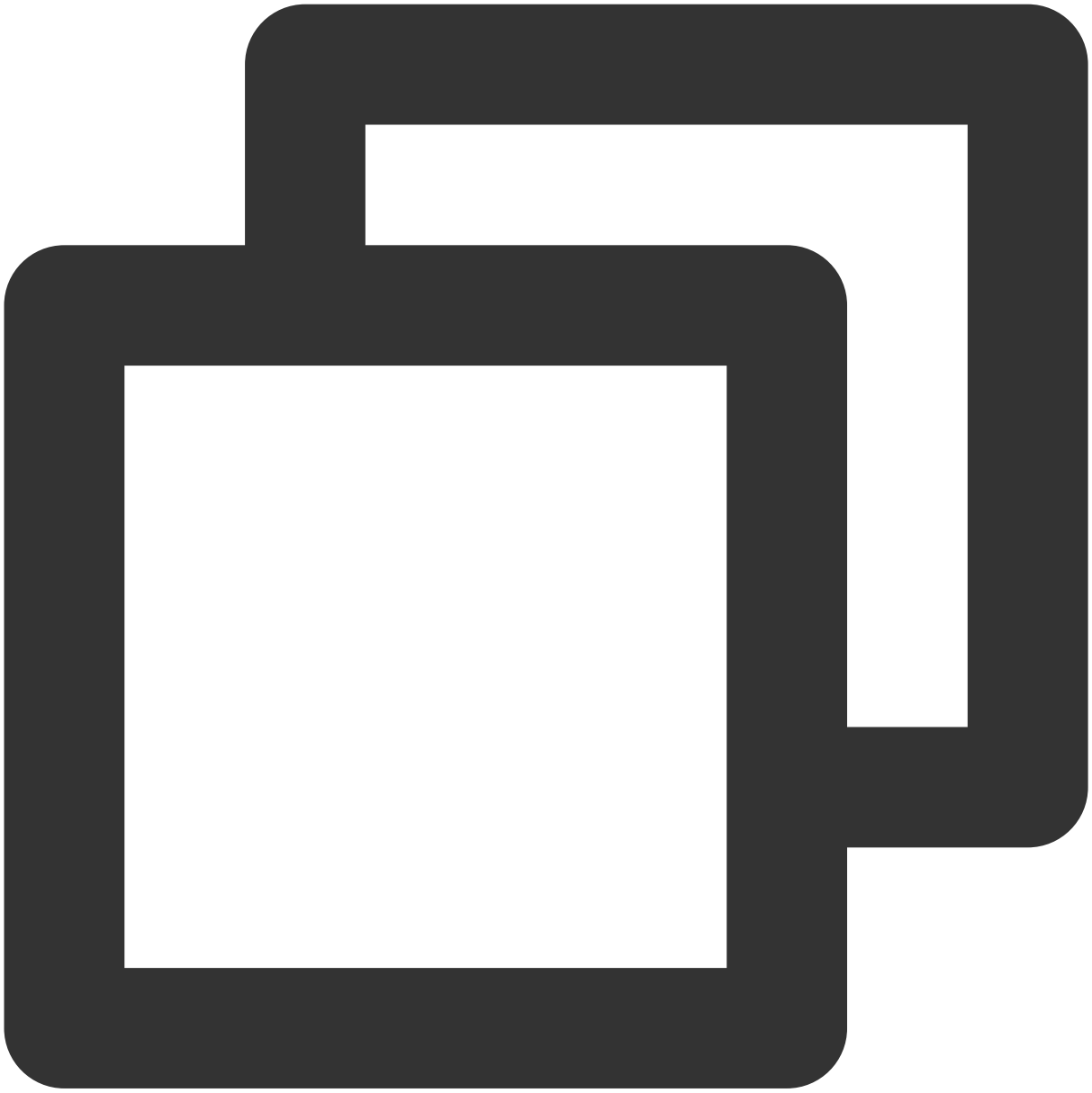
说明：

图片格式为 GIF 时，不支持该操作。

锐化

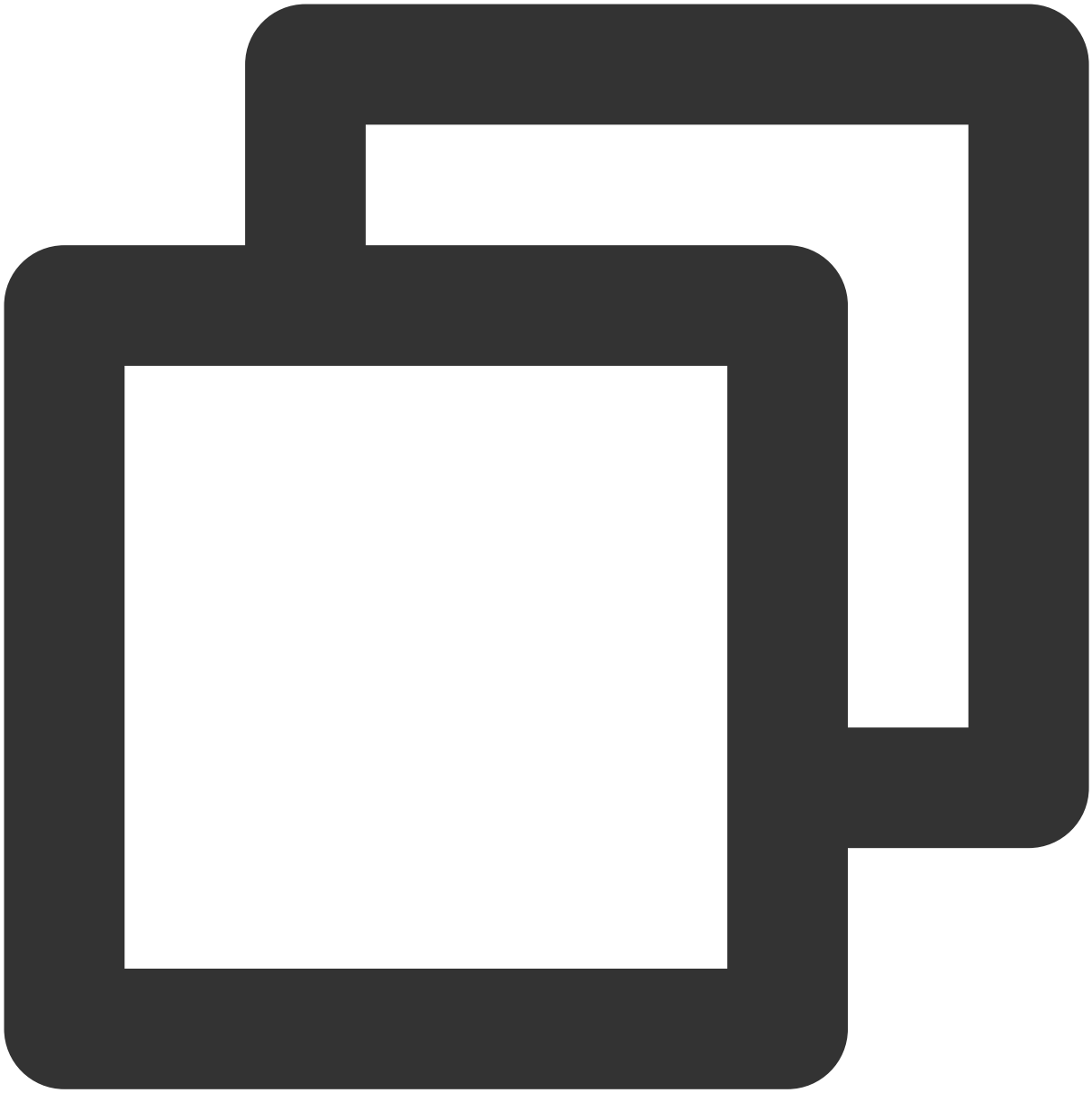
图片锐化功能，value 为锐化参数值，取值范围为10 - 300间的整数，推荐使用70。参数值越大，锐化效果越明显。API 文档请参见 锐化接口。

Objective-C



```
// 以锐化值100为例
[transform setSharpenWith:100];
```

swift



```
transform.setSharpenWith(100)
```

水印

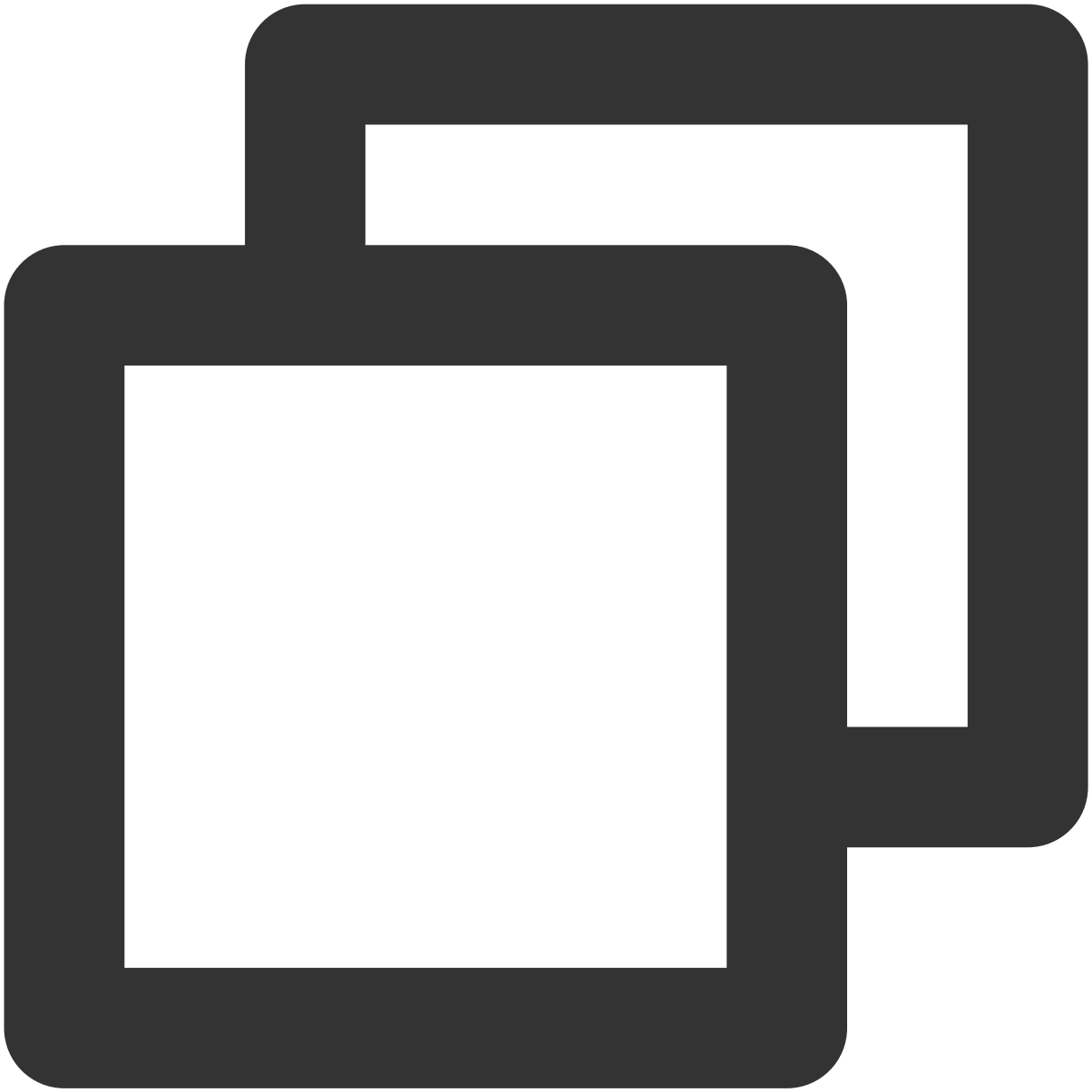
说明：

API 文档请参见 图片水印接口 和 文字水印接口。

方位图说明，请参见 gravity。

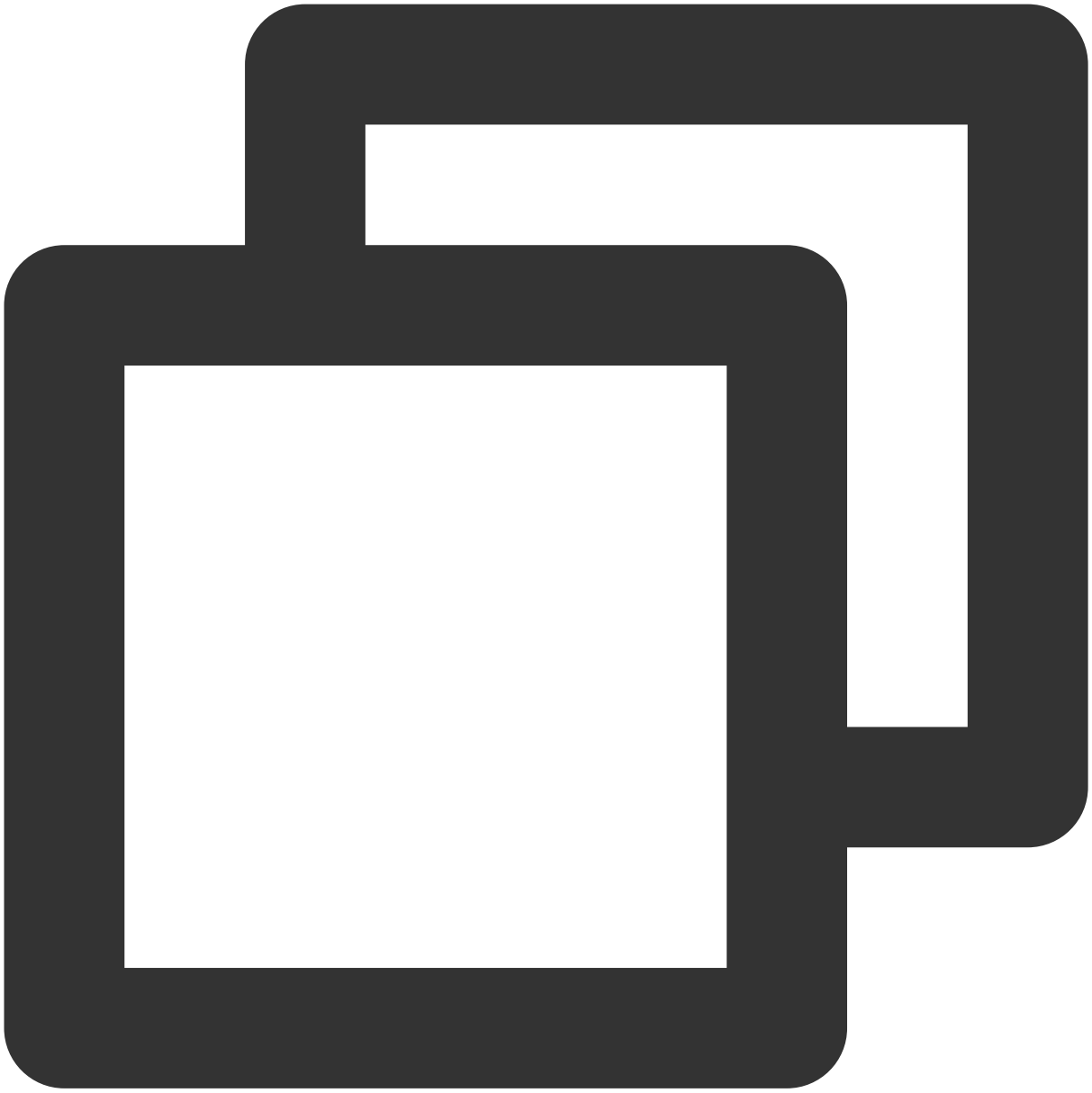
图片水印

Objective-C



```
//imageUrl 水印图片地址
// gravity 文字水印位置，九宫格位置（参考九宫格方位图 ），默认值 SouthEast
// dx 水平（横轴）边距，单位为像素，缺省值为0
// dy 垂直（纵轴）边距，单位为像素，默认值为0
// blogo 水印图适配功能，适用于水印图尺寸过大的场景（如水印墙）。共有两种类型：
///     当 blogo 设置为1时，水印图会被缩放至与原图相似大小后添加
///     当 blogo 设置为2时，水印图会被直接裁剪至与原图相似大小后添加
[tran setWaterMarkWithImageUrl:@"http://tpg-1253653367.cos.ap-guangzhou.myqcloud.co
```

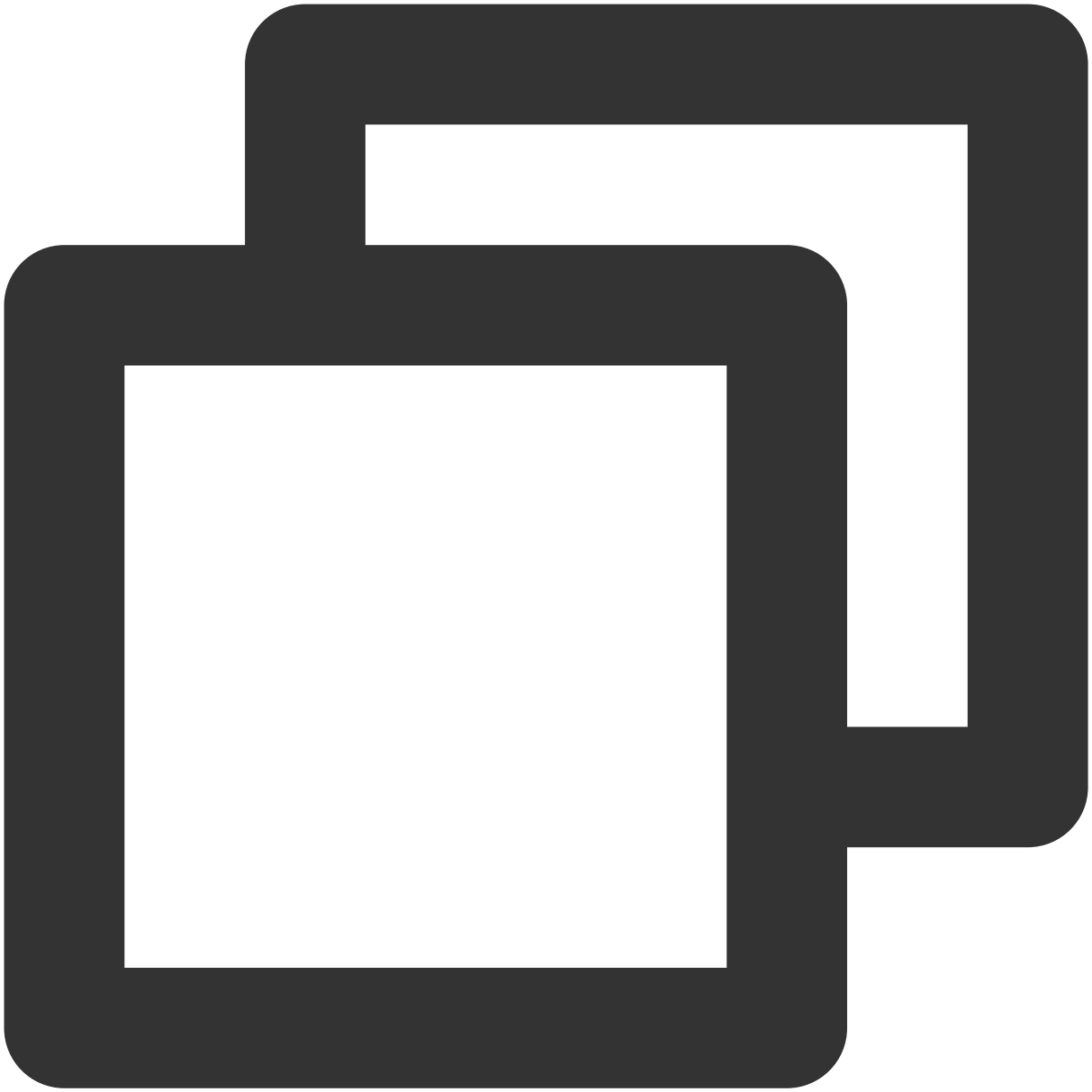
swift



```
transform.setWaterMarkWithImageUrl("", gravity: CloudInfiniteGravity.CIGravityCente
```

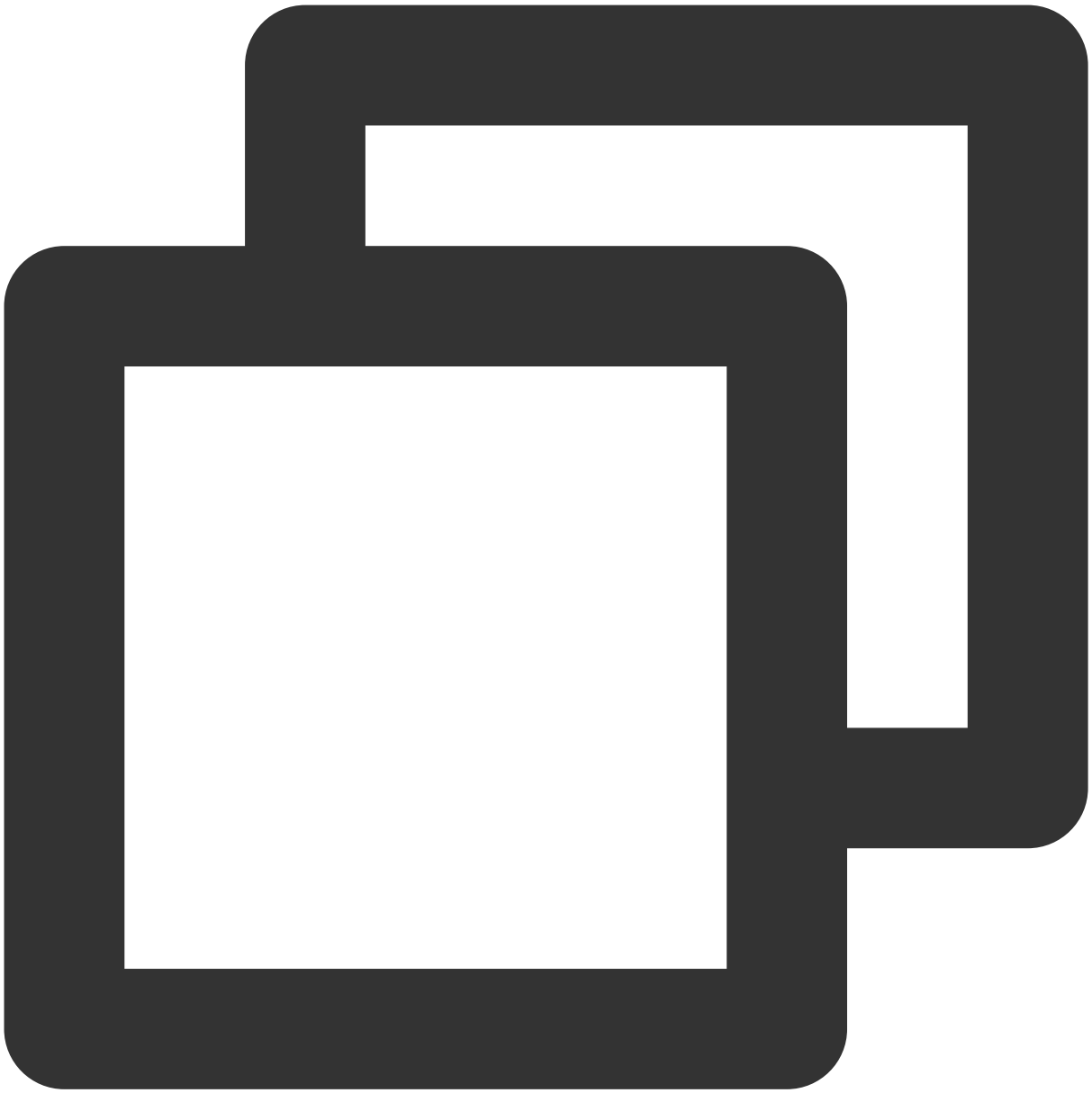
文字水印

Objective-C



```
// text 水印内容
// font 水印字体
// color 字体颜色，默认值为 #3D3D3D
// dissolve 文字透明度，取值1 - 100 ，默认90
// gravity 文字水印位置，默认值 SouthEast
// dx 水平（横轴）边距，单位为像素，缺省值为0
// dy 垂直（纵轴）边距，单位为像素，默认值为0
// batch 平铺水印功能，可将文字水印平铺至整张图片。当 batch 设置为1时，开启平铺水印功能
// degree 文字水印的旋转角度设置，取值范围为0 - 360，默认0
[transform setWaterMarkText:@"腾讯云数据万象" font:nil textColor:nil dissolve:90 gravi
```

swift

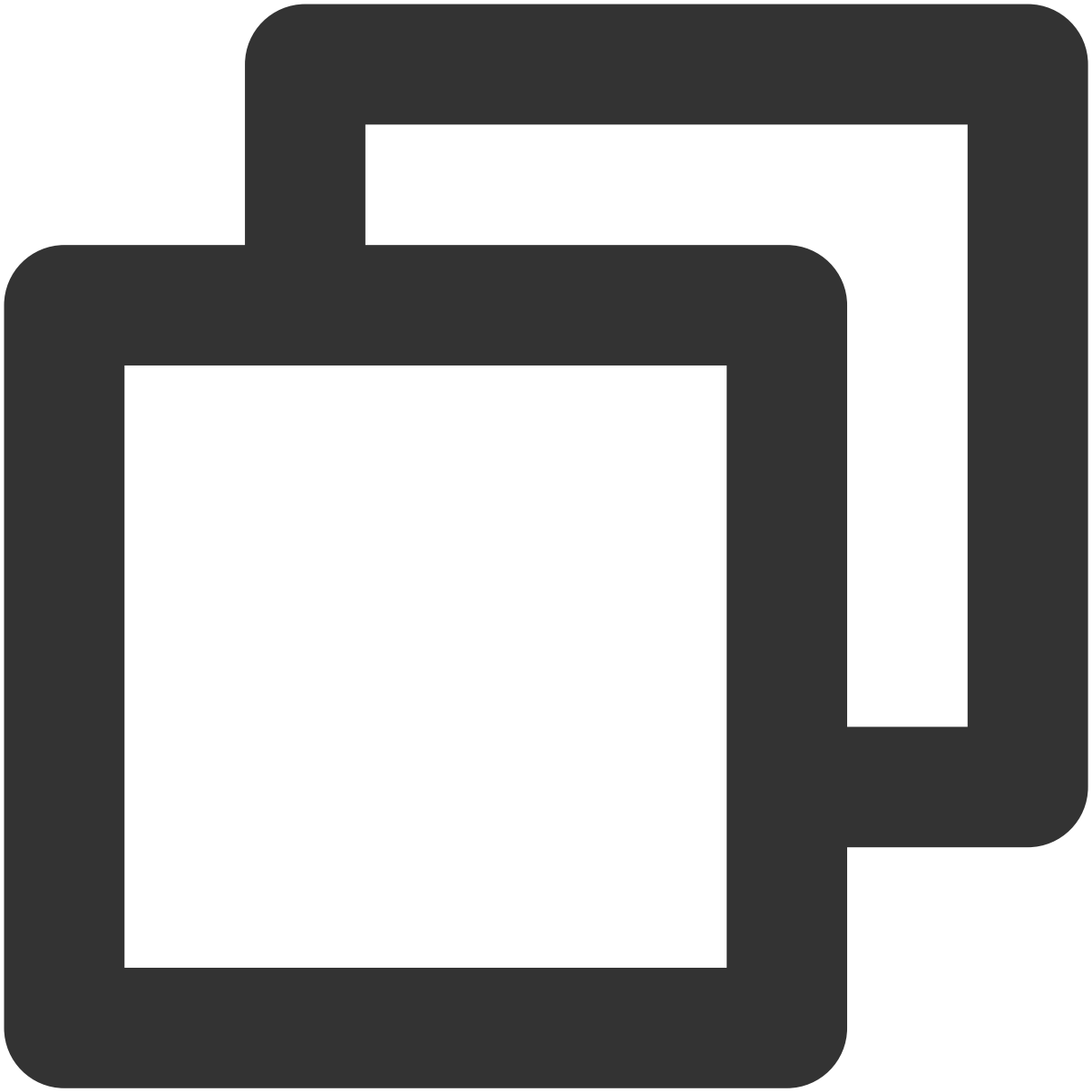


```
transform.setWaterMarkText("腾讯云数据万象", font: nil, textColor: nil, dissolve: 90,
```

获取图片主题色

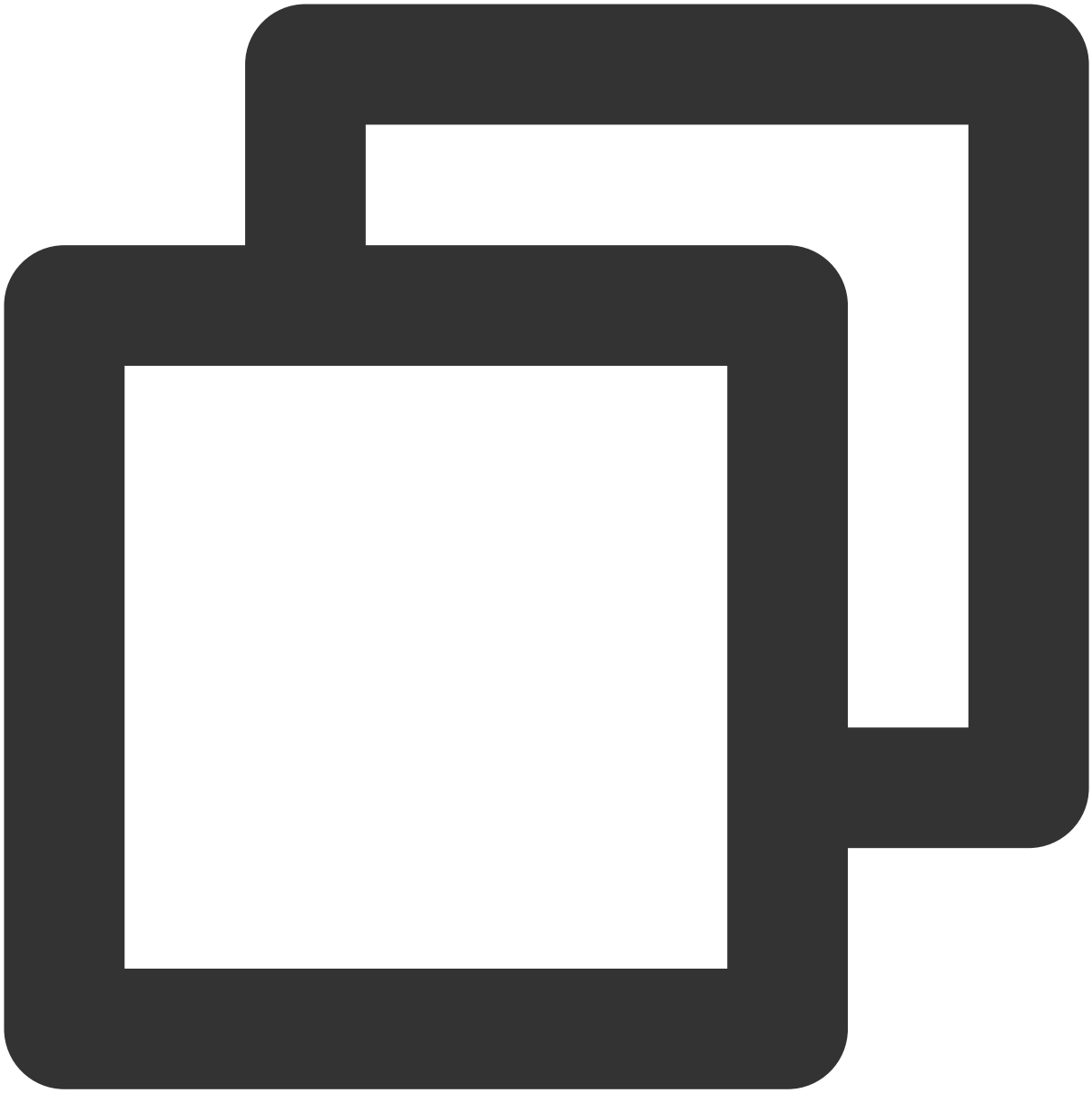
腾讯云数据万象通过 imageAve 接口获取图片主色调信息。API 文档请参见 获取主题色接口。

Objective-C



```
[transform setViewBackgroudColorWithImageAveColor:YES]
```

swift

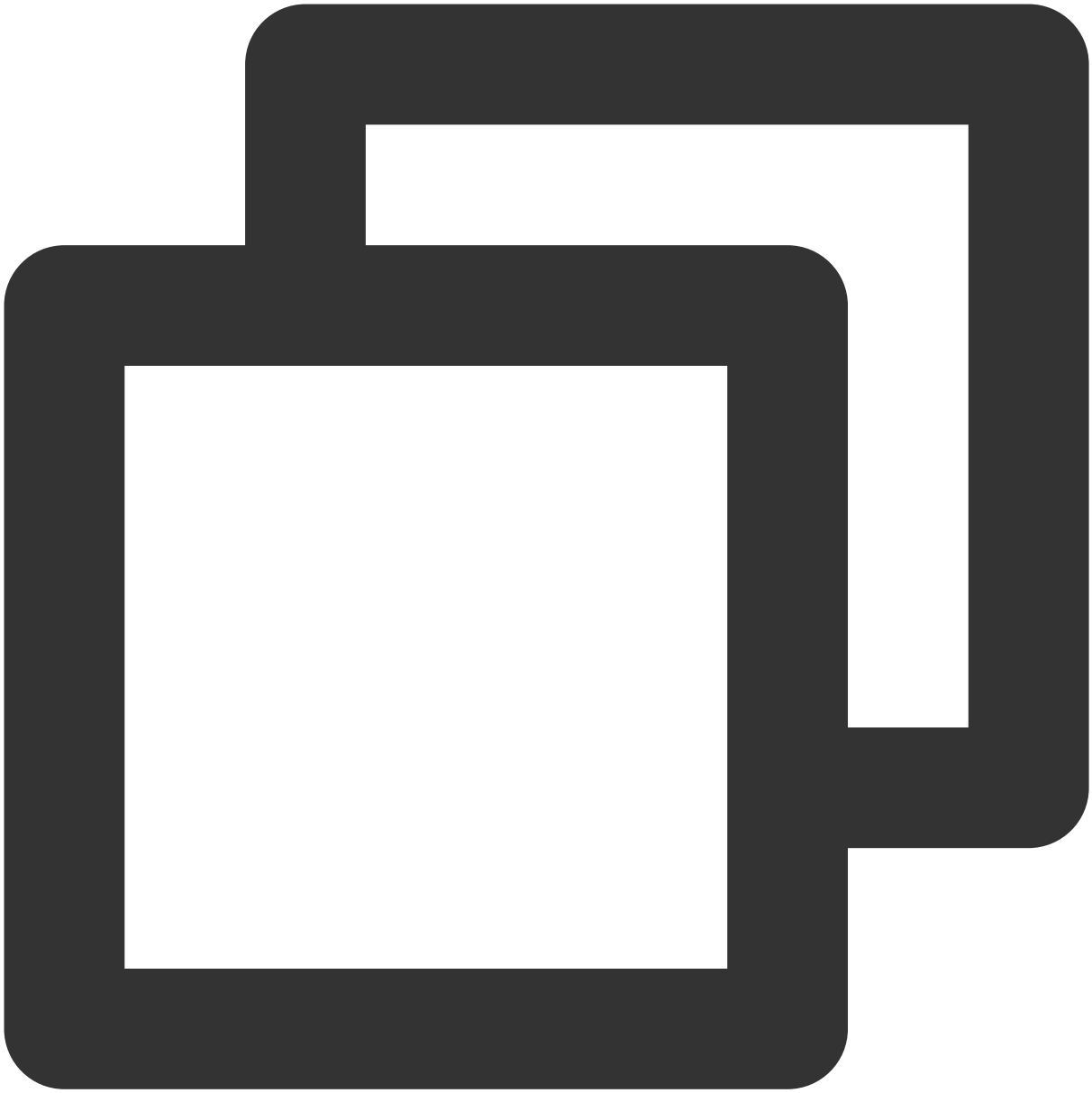


```
transform.setViewBackgroudColorWithImageAveColor(true)
```

去除图片元信息

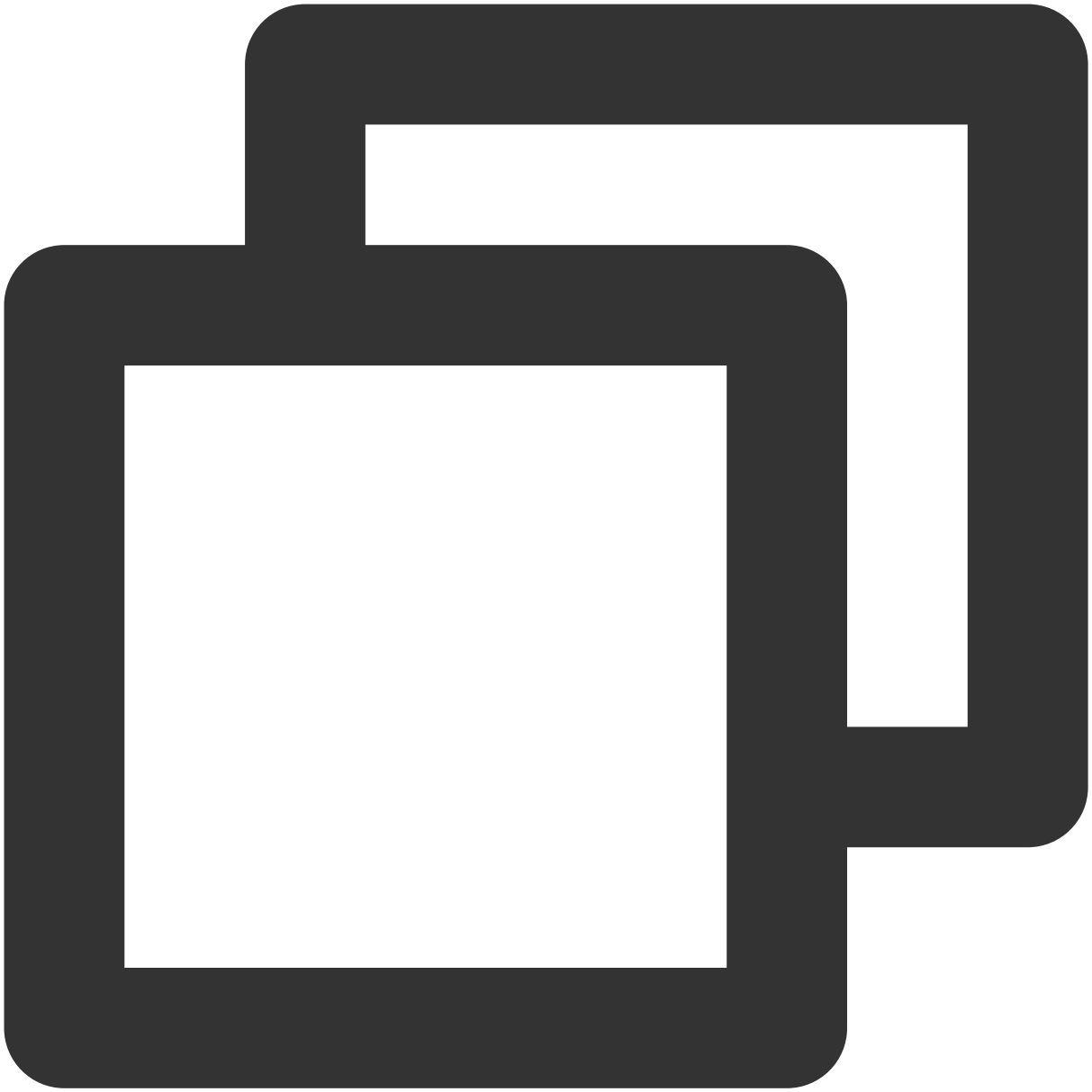
腾讯云数据万象通过 imageMogr2 接口可去除图片元信息，包括 exif 信息。API 文档请参见 去除图片元信息接口。

Objective-C



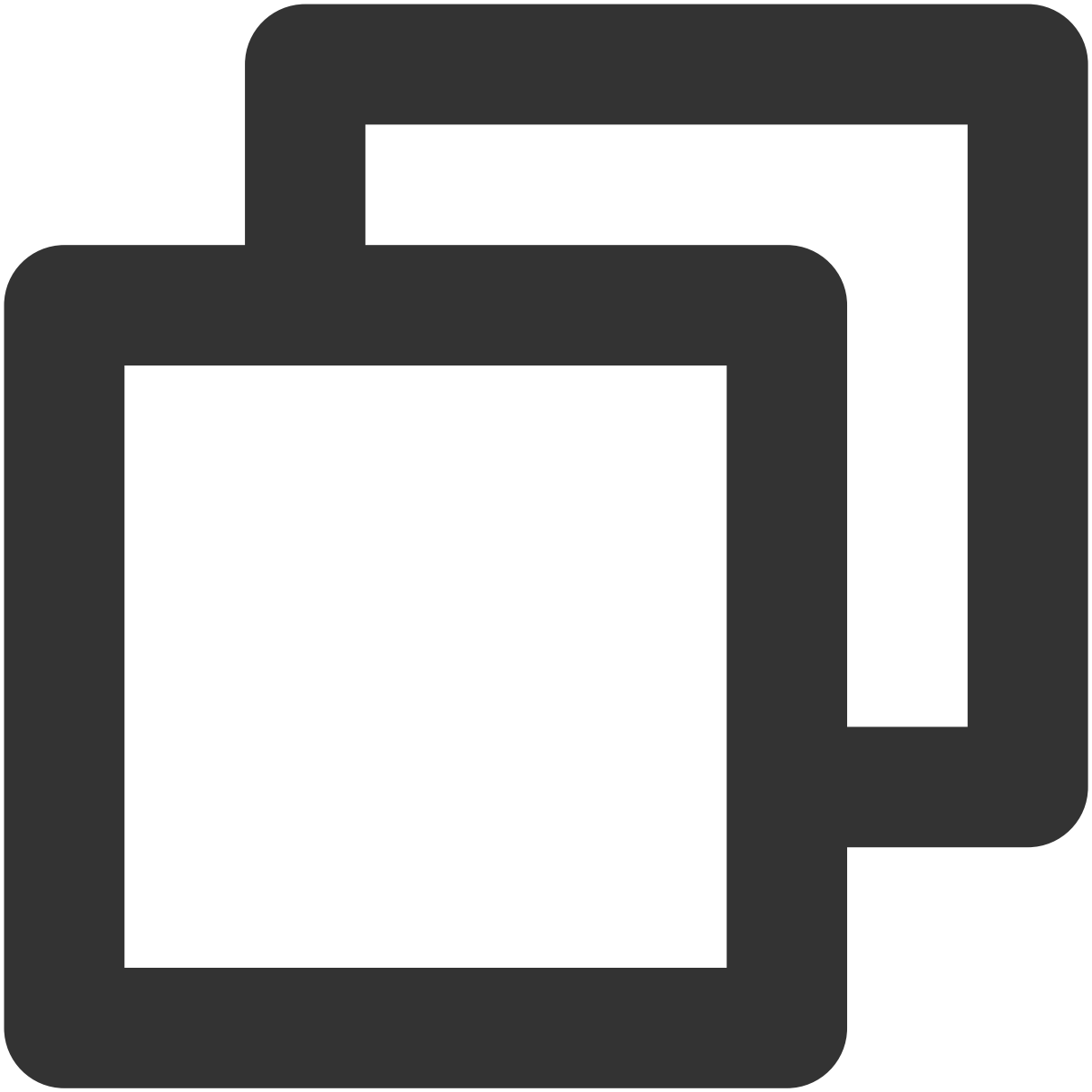
```
[transform setImageStrip];
```

swift



```
transform.setImageStrip()
```

组合操作



```
CITransformation * tran = [CITransformation new];
//    ***************缩放*************
// 缩放百分之50
[transform setZoomWithPercent:50 scaleType:ScalePercentTypeALL];


//    **格式转换
// 转为TPG格式
[tran setFormatWith:CIImageTypeTPG];
//    **裁剪
```

```
// 裁剪图片 宽高100 左上角左边 30 30
[tran setCutWithWidth:100 height:100 dx:30 dy:30];
```

COS Android SDK
内容审核
图片审核

最近更新时间：2024-03-13 15:17:16

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
SensitiveContentRecognition	图片同步审核	用于图片同步审核
PostImagesAudit	图片批量审核	用于图片批量审核
GetImageAudit	查询图片审核任务结果	用于查询图片审核任务结果

SDK API 参考

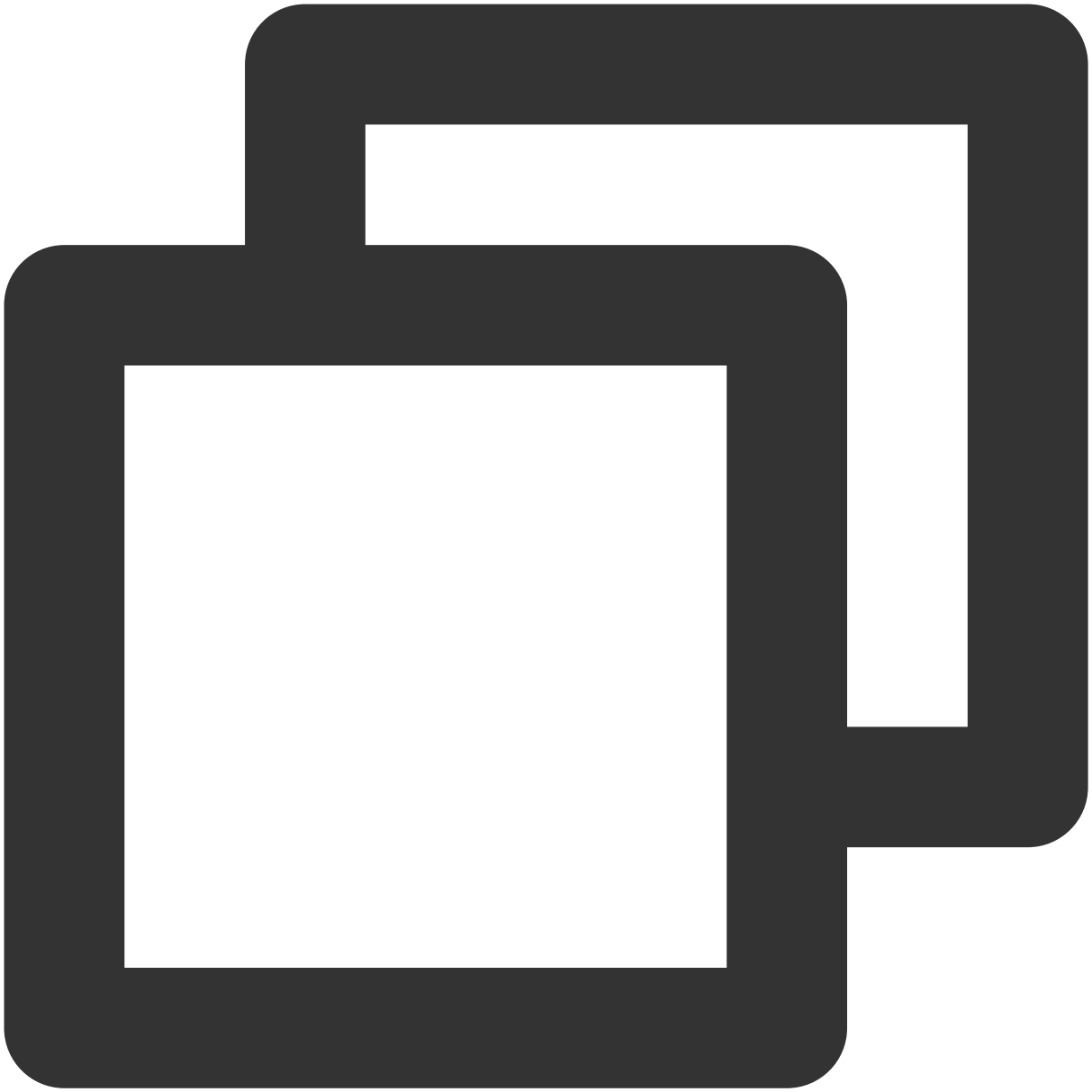
SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

图片同步审核

功能说明

用于图片同步审核。

示例代码



```
// 存储桶名称，由bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https:
String bucket = "examplebucket-1250000000";
String key = "exampleobject"; //对象键
SensitiveContentRecognitionRequest sensitiveContentRecognitionRequest = new Sensiti
sensitiveContentRecognitionRequest.addType("ads");
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.sensitiveContentRecognitionAsync(sensitiveContentRecognitionRequest, new
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        SensitiveContentRecognitionResult sensitiveContentRecognitionResult = (Sens
    }
```

```
    @Override
    public void onFail(CosXmlRequest request, CosXmlClientException clientException
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

说明：

更多完整示例，请前往 GitHub 查看。

图片批量审核

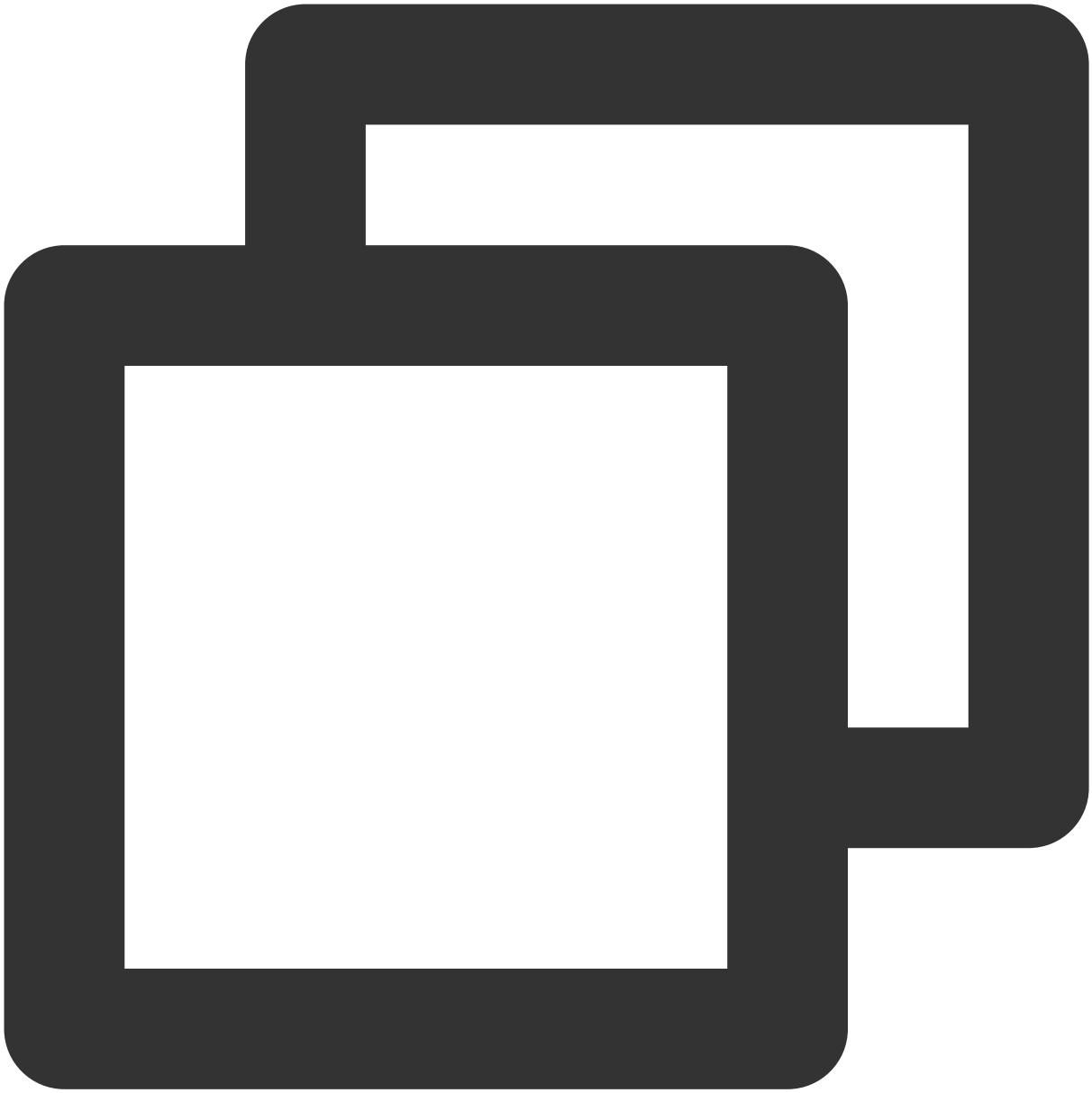
功能说明

用于图片批量审核。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
String cosPath1 = "dir1/exampleobject1.jpg";
String cosPath2 = "dir1/exampleobject2.jpg";
//图片的链接地址,Object 和 Url 只能选择其中一种
String imageUrl = "https://myqcloud.com/%205image.jpg";
PostImagesAuditRequest request = new PostImagesAuditRequest(bucket);
PostImagesAudit.ImagesAuditInput image1 = new PostImagesAudit.ImagesAuditIn
image1.object = cosPath1;
//设置原始内容，长度限制为512字节，该字段会在响应中原样返回
```

```
        image1.dataId = "DataId1";
        //截帧频率，GIF 图检测专用，默认值为5，表示从第一帧（包含）开始每隔5帧截取一帧
        image1.interval = 2;
        //最大截帧数量，GIF 图检测专用，默认值为5，表示只截取 GIF 的5帧图片进行审核，必须大于0
        image1.maxFrames = 5;
        PostImagesAudit.ImagesAuditInput image2 = new PostImagesAudit.ImagesAuditIn
        image2.object = cosPath2;
        image2.dataId = "DataId2";
        image2.interval = 2;
        image2.maxFrames = 5;
        PostImagesAudit.ImagesAuditInput image3 = new PostImagesAudit.ImagesAuditIn
        image3.url = imageUrl;
        image3.dataId = "DataId3";
        image3.interval = 2;
        image3.maxFrames = 5;
        request.addImage(image1);
        request.addImage(image2);
        request.addImage(image3);
        //审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以,分隔,
        request.setDetectType("Porn,Ads");

        ciService.postImagesAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 图片批量审核的结果
                // 详细字段请查看api文档或者SDK源码
                PostImagesAuditResult result = (PostImagesAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

查询图片审核任务结果

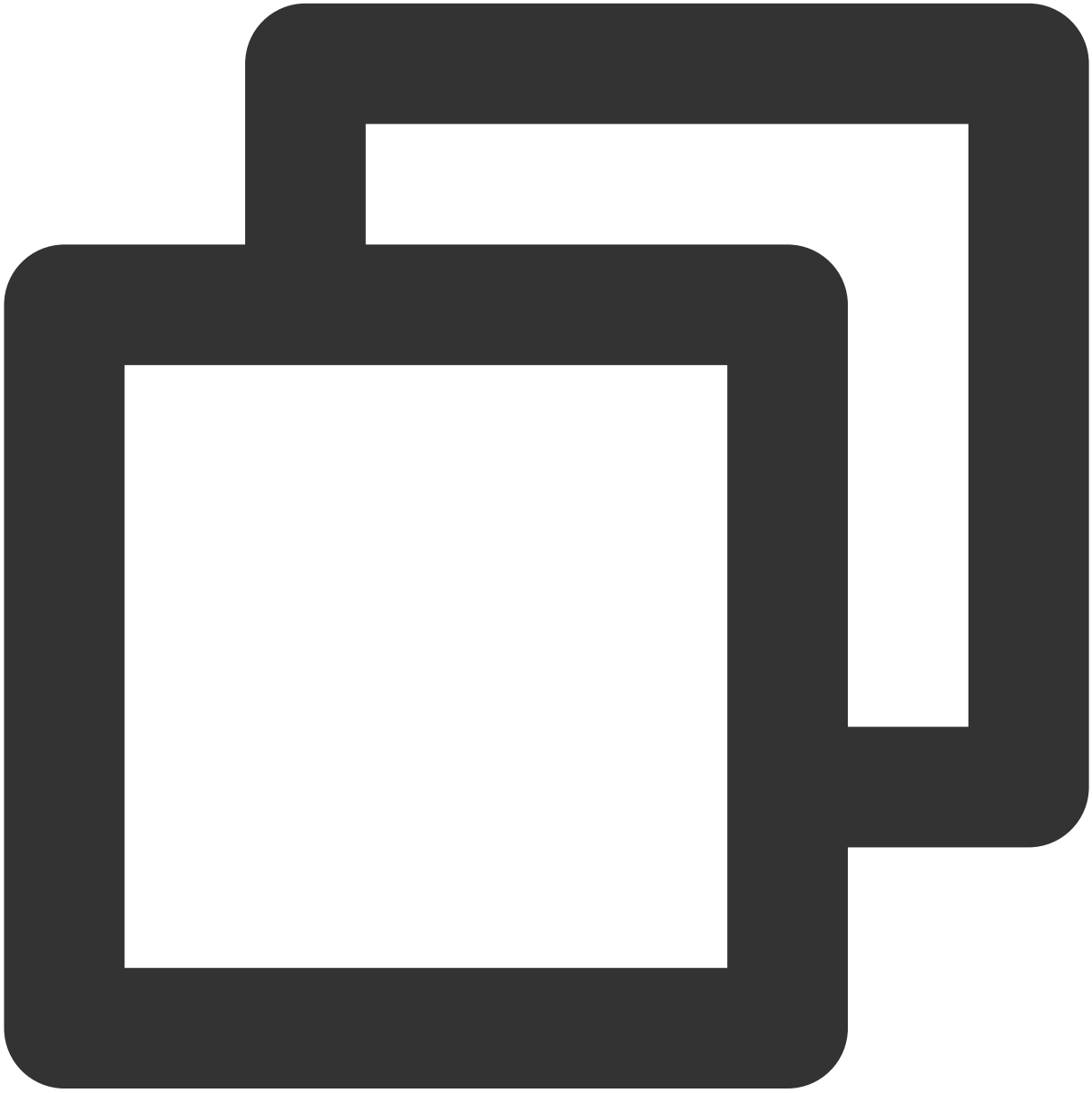
功能说明

用于查询图片审核任务结果。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 审核任务的 ID
String jobId = "iab1ca9fc8a3ed11ea834c525400863904";
GetImageAuditRequest request = new GetImageAuditRequest(bucket, jobId);
```

```
        ciService.getImageAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 查询图片审核任务的结果
                // 详细字段请查看api文档或者SDK源码
                GetImageAuditResult result = (GetImageAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

视频审核

最近更新时间：2024-03-13 15:17:14

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PostVideoAudit	提交视频审核任务	用于提交视频审核任务
GetVideoAudit	查询视频审核任务结果	用于查询视频审核任务结果

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

提交视频审核任务

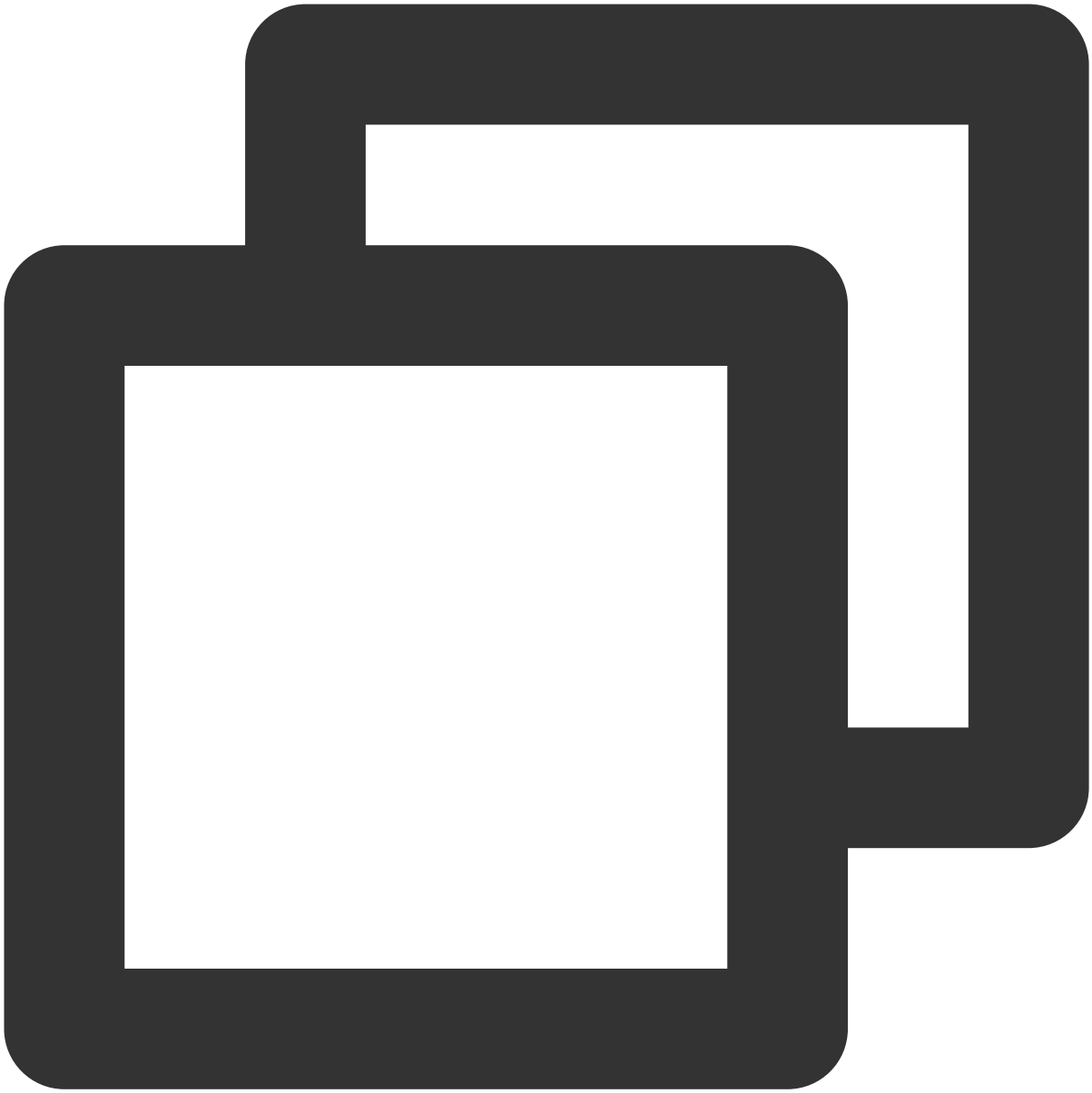
功能说明

用于提交视频审核任务。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
String cosPath = "dir1/exampleobject.mp4";
//视频的链接地址,Object 和 Url 只能选择其中一种
String url = "https://myqcloud.com/%205video.mp4";
PostVideoAuditRequest request = new PostVideoAuditRequest(bucket);
request.setObject(cosPath);
request.setUrl(url);
//设置原始内容，长度限制为512字节，该字段会在响应中原样返回
request.setDataId("DataId");
```

```
        //回调地址，以http://或者https://开头的地址。
        request.setCallback("https://github.com");
        //回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）
        request.setCallbackVersion("Detail");
        //视频截帧数量，范围为(0, 10000]。
        request.setCount(3);
        //视频截帧频率，范围为(0, 60]，单位为秒，支持 float 格式，执行精度精确到毫秒。
        request.setTimeInterval(10);
        //审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔
        request.setDetectType("Porn,Ads");
        //用于指定是否审核视频声音，当值为0时：表示只审核视频画面截图；值为1时：表示同时审核视频画
        request.setDetectContent(1);

        // CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
        ciService.postVideoAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 提交视频审核任务的结果
                // 详细字段请查看api文档或者SDK源码
                PostAuditResult result = (PostAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

查询视频审核任务结果

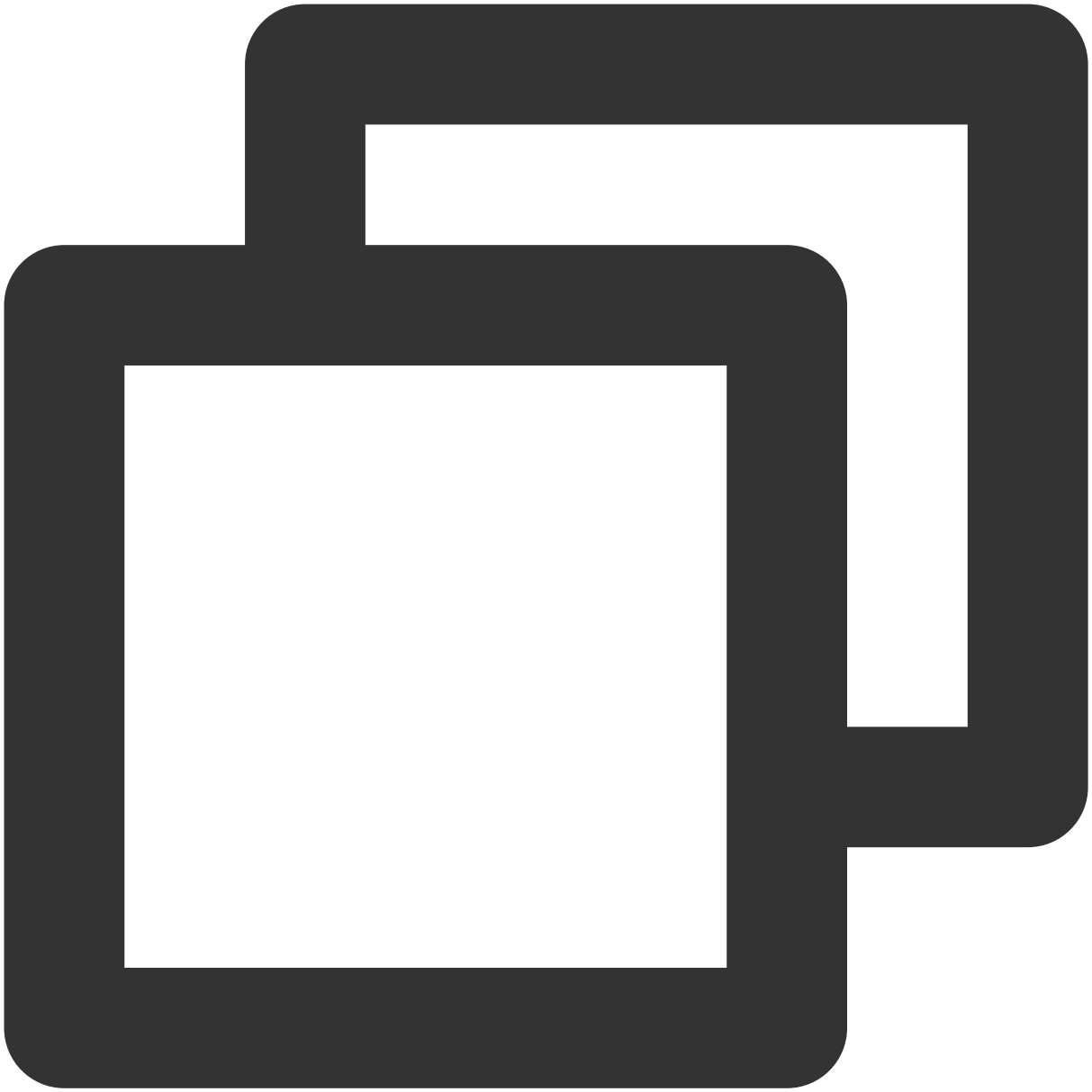
功能说明

用于查询视频审核任务结果。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 审核任务的 ID
String jobId = "iab1ca9fc8a3ed11ea834c525400863904";
GetVideoAuditRequest request = new GetVideoAuditRequest(bucket, jobId);
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.getVideoAuditAsync(request, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
        // result 查询视频审核任务的结果
        // 详细字段请查看api文档或者SDK源码
```

```
                GetVideoAuditResult result = (GetVideoAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

音频审核

最近更新时间：2024-03-13 15:17:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于音频审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PostAudioAudit	提交音频审核任务	用于提交音频审核任务
GetAudioAudit	查询音频审核任务结果	用于查询音频审核任务结果

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

提交音频审核任务

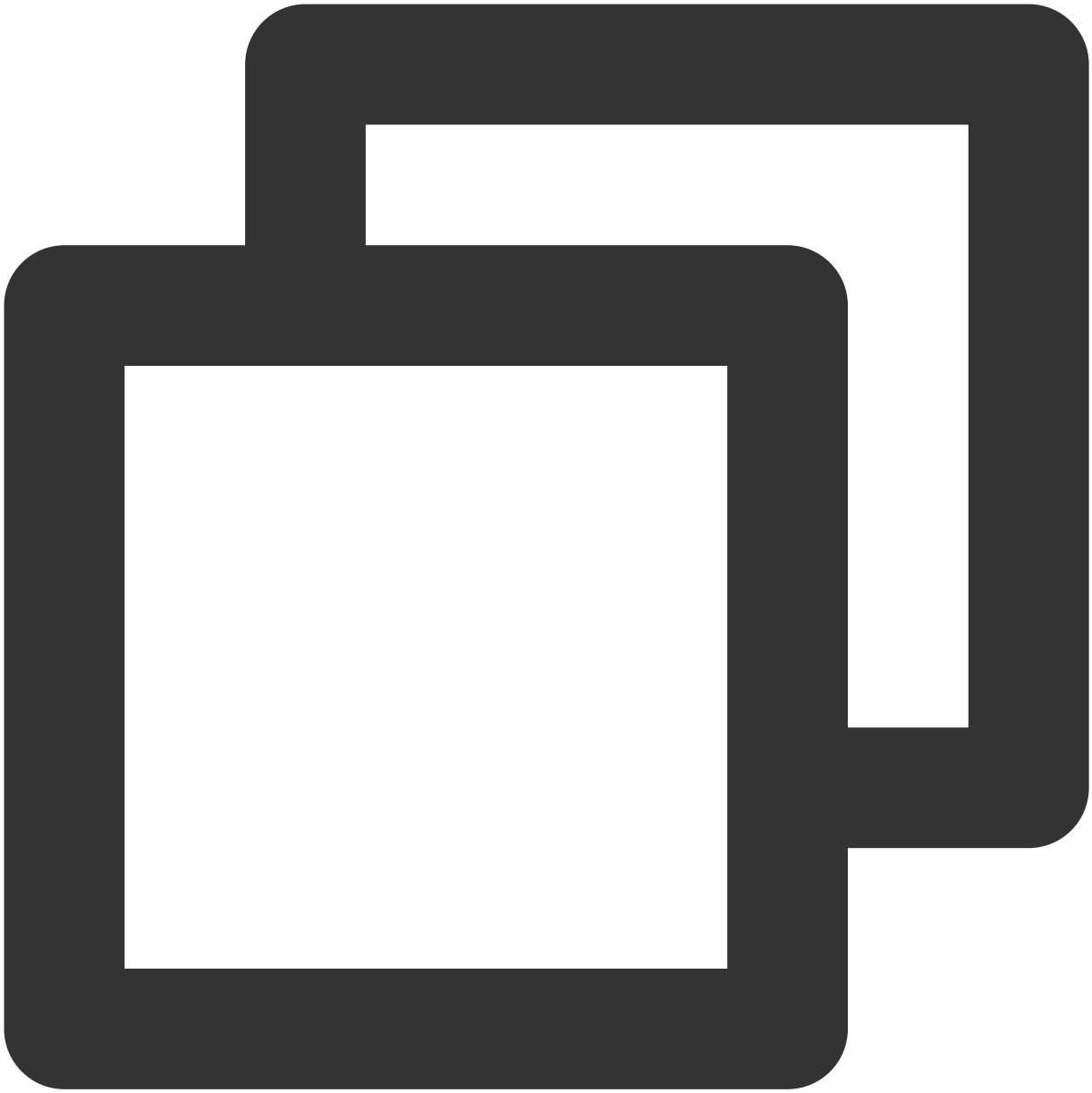
功能说明

用于提交音频审核任务。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
String cosPath = "dir1/exampleobject.mp3";
//音频的链接地址,Object 和 Url 只能选择其中一种
String url = "https://myqcloud.com/%205Audio.mp3";
PostAudioAuditRequest request = new PostAudioAuditRequest(bucket);
request.setObject(cosPath);
request.setUrl(url);
//设置原始内容，长度限制为512字节，该字段会在响应中原样返回
request.setDataId("DataId");
```

```
        //回调地址，以http://或者https://开头的地址。
        request.setCallback("https://github.com");
        //回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）
        request.setCallbackVersion("Detail");
        //审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔
        request.setDetectType("Porn,Ads");

        // CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
        ciService.postAudioAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 提交音频审核任务的结果
                // 详细字段请查看api文档或者SDK源码
                PostAuditResult result = (PostAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

查询音频审核任务结果

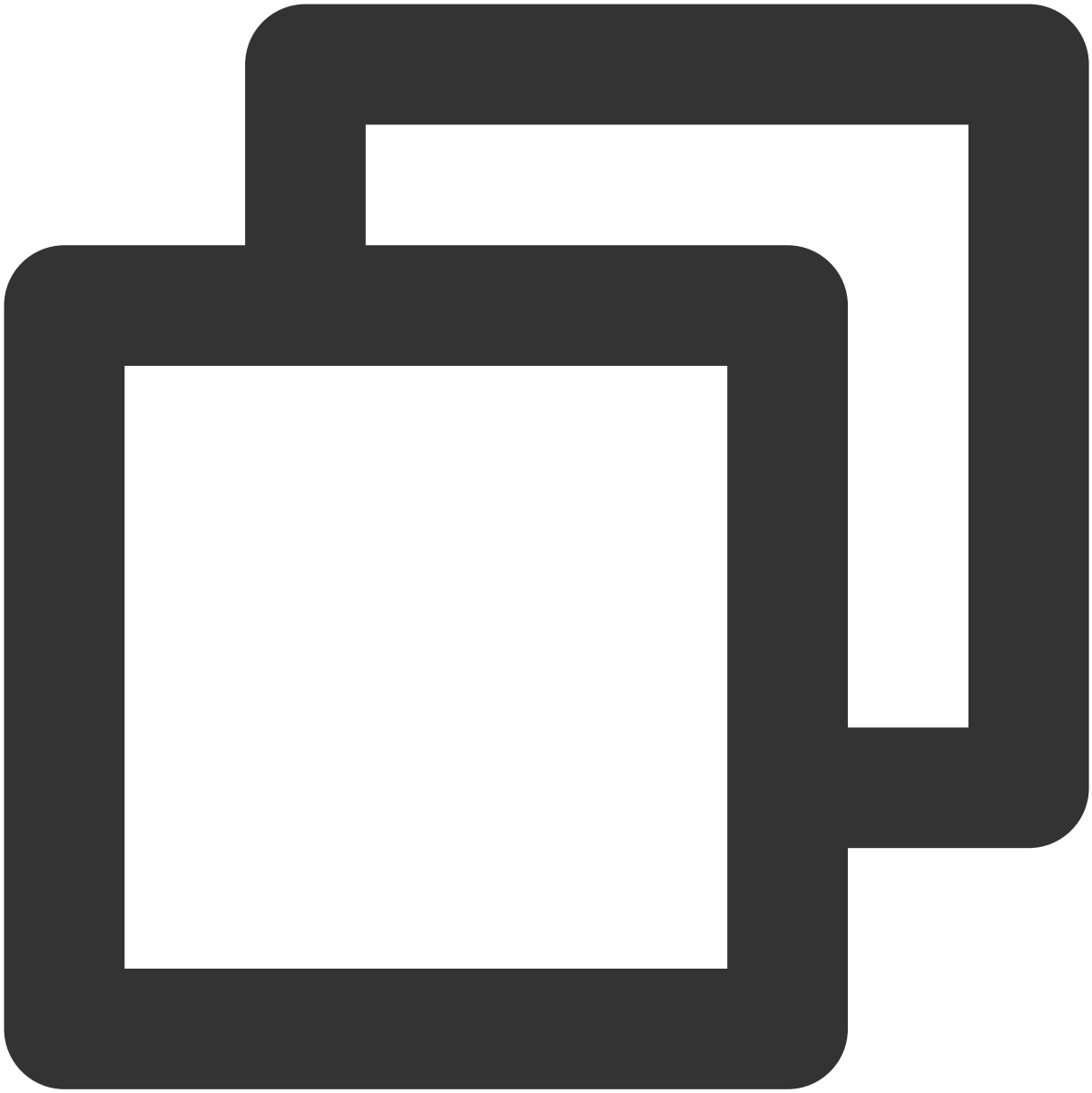
功能说明

用于查询音频审核任务结果。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 审核任务的 ID
String jobId = "iab1ca9fc8a3ed11ea834c525400863904";
GetAudioAuditRequest request = new GetAudioAuditRequest(bucket, jobId);
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.getAudioAuditAsync(request, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
        // result 查询音频审核任务的结果
        // 详细字段请查看api文档或者SDK源码
```

```
                GetAudioAuditResult result = (GetAudioAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

文本审核

最近更新时间：2024-03-13 15:17:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文本审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PostTextAudit	提交文本审核任务	用于提交文本审核任务
GetTextAudit	查询文本审核任务结果	用于查询文本审核任务结果

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

提交文本审核任务

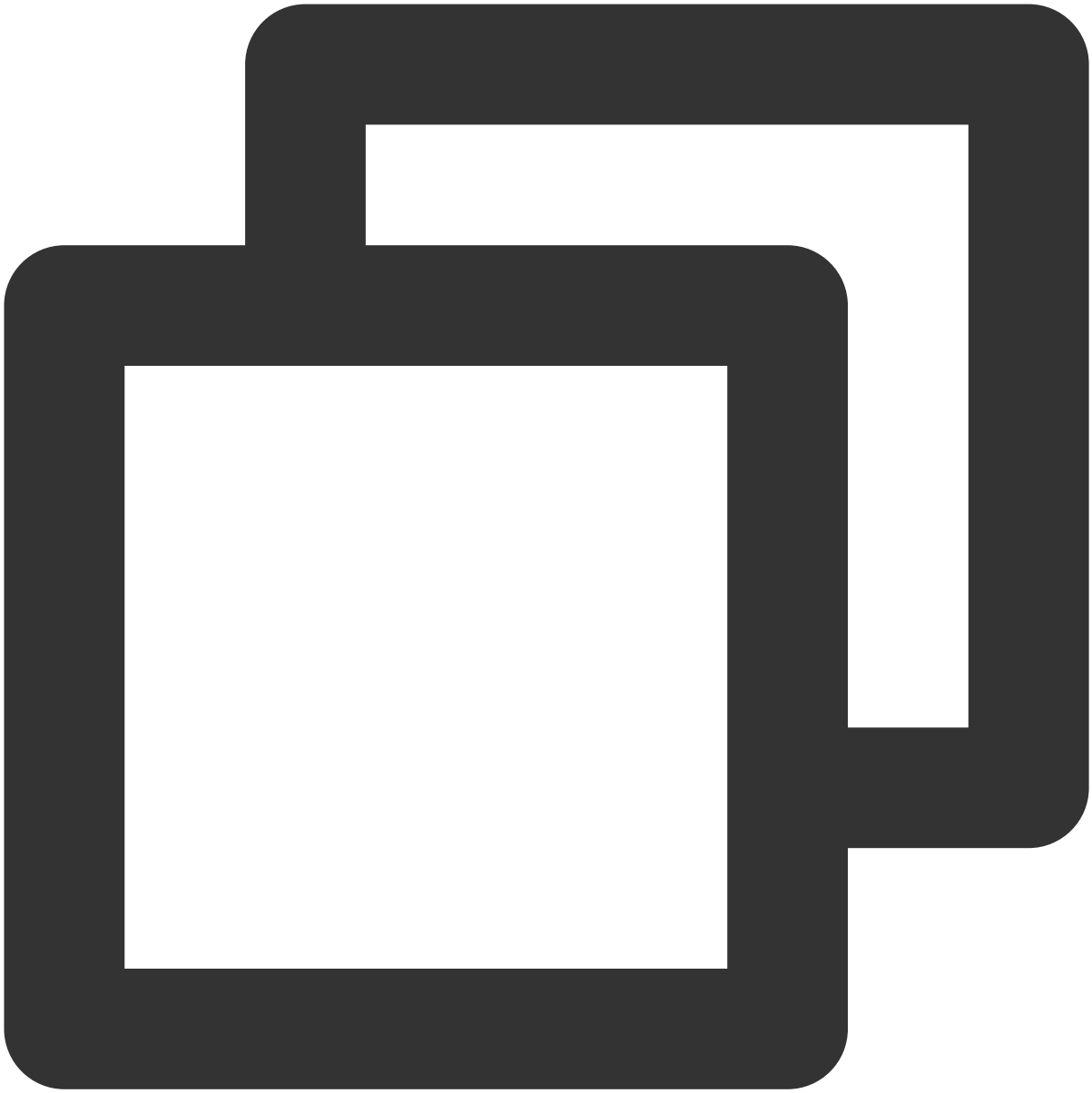
功能说明

用于提交文本审核任务。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
String cosPath = "dir1/exampleobject.txt";
//文本的链接地址,Object 和 Url 只能选择其中一种
String url = "https://myqcloud.com/%205text.txt";
//当传入的内容为纯文本信息，需要先经过 base64 编码，文本编码前的原文长度不能超过10000个
String content = Base64.encodeToString("测试文本".getBytes(Charset.forName("U
PostTextAuditRequest request = new PostTextAuditRequest(bucket);
request.setObject(cosPath);
request.setUrl(url);
```

```
        request.setContent(content);
        //设置原始内容，长度限制为512字节，该字段会在响应中原样返回
        request.setDataId("DataId");
        //回调地址，以http://或者https://开头的地址。
        request.setCallback("https://github.com");
        //回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）
        request.setCallbackVersion("Detail");
        //审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔
        request.setDetectType("Porn,Ads");

        // CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
        ciService.postTextAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 提交文本审核任务的结果
                // 详细字段请查看api文档或者SDK源码
                TextAuditResult result = (TextAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

查询文本审核任务结果

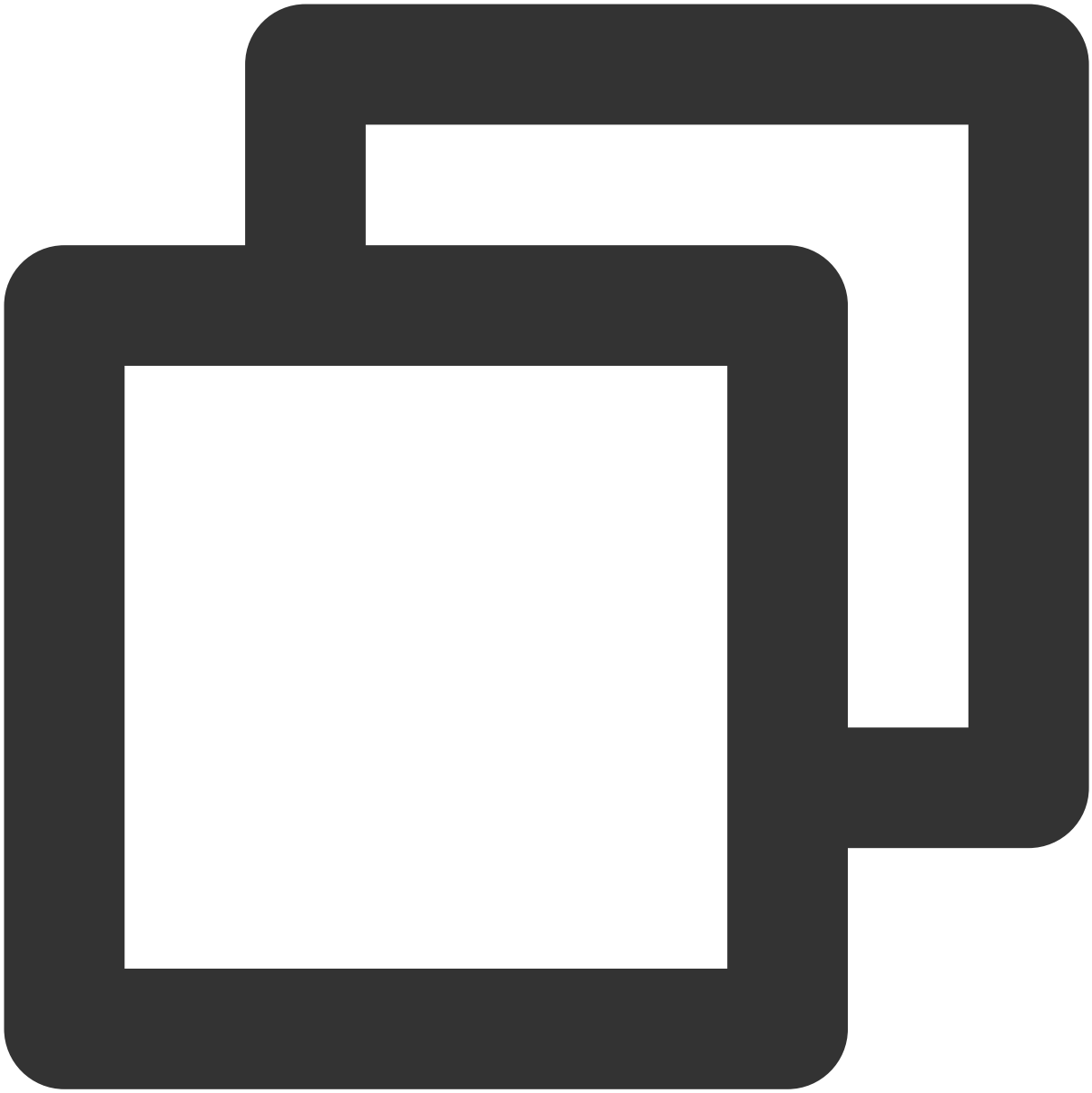
功能说明

用于查询文本审核任务结果。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 审核任务的 ID
String jobId = "iab1ca9fc8a3ed11ea834c525400863904";
GetTextAuditRequest request = new GetTextAuditRequest(bucket, jobId);
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.getTextAuditAsync(request, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
        // result 查询文本审核任务的结果
        // 详细字段请查看api文档或者SDK源码
```

```
                TextAuditResult result = (TextAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

文档审核

最近更新时间：2024-03-13 15:17:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文档审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PostDocumentAudit	提交文档审核任务	用于提交文档审核任务
GetDocumentAudit	查询文档审核任务结果	用于查询文档审核任务结果

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

提交文档审核任务

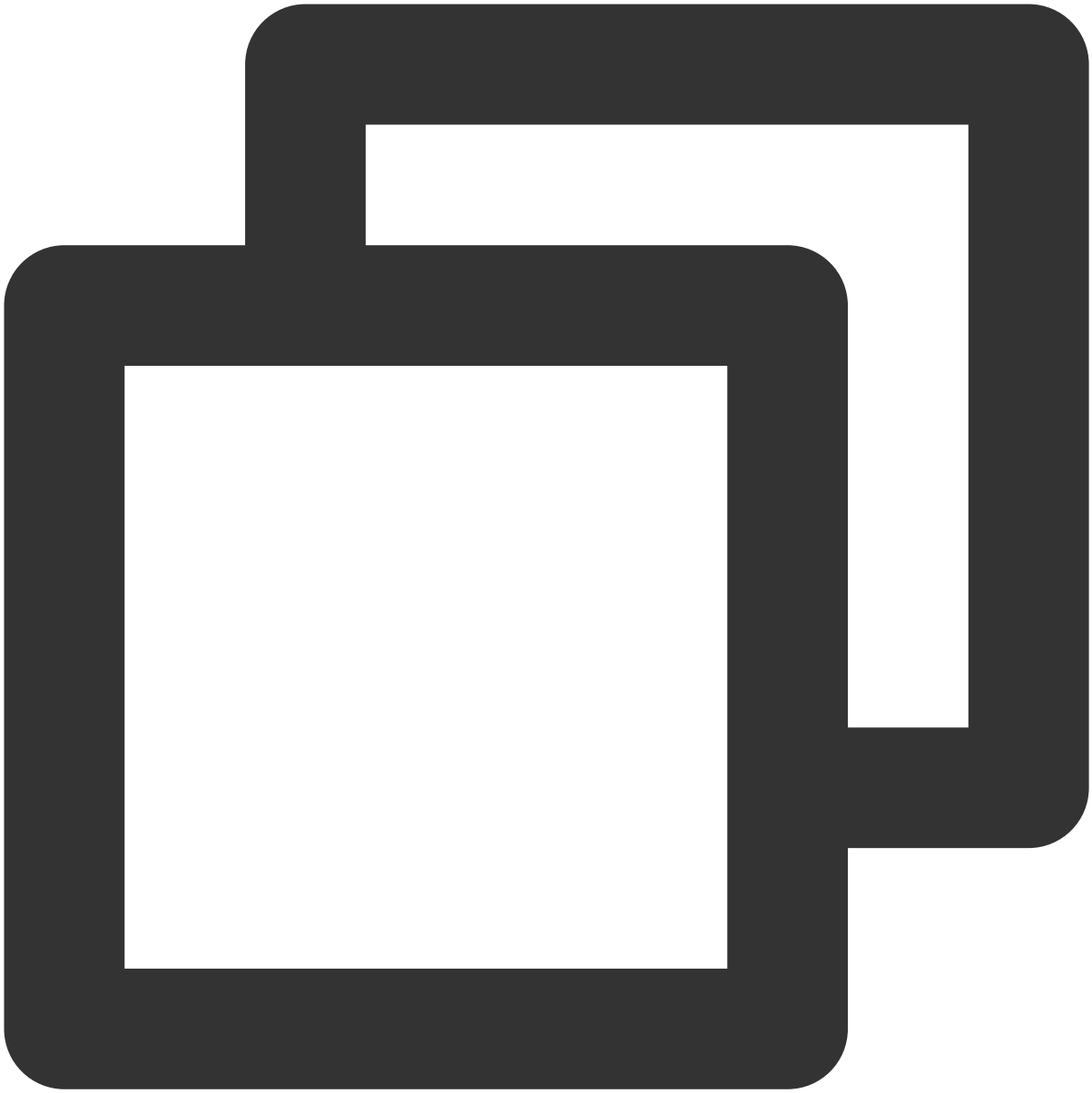
功能说明

用于提交文档审核任务。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
String cosPath = "dir1/exampleobject.pdf";
//文档的链接地址,Object 和 Url 只能选择其中一种
String url = "https://myqcloud.com/%205Document.pdf";
PostDocumentAuditRequest request = new PostDocumentAuditRequest(bucket);
request.setObject(cosPath);
request.setUrl(url);
//设置原始内容，长度限制为512字节，该字段会在响应中原样返回
request.setDataId("DataId");
```

```
        //回调地址，以http://或者https://开头的地址。
        request.setCallback("https://github.com");
        //审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔
        request.setDetectType("Porn,Ads");

        // CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
        ciService.postDocumentAuditAsync(request, new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
                // result 提交文档审核任务的结果
                // 详细字段请查看api文档或者SDK源码
                PostAuditResult result = (PostAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

查询文档审核任务结果

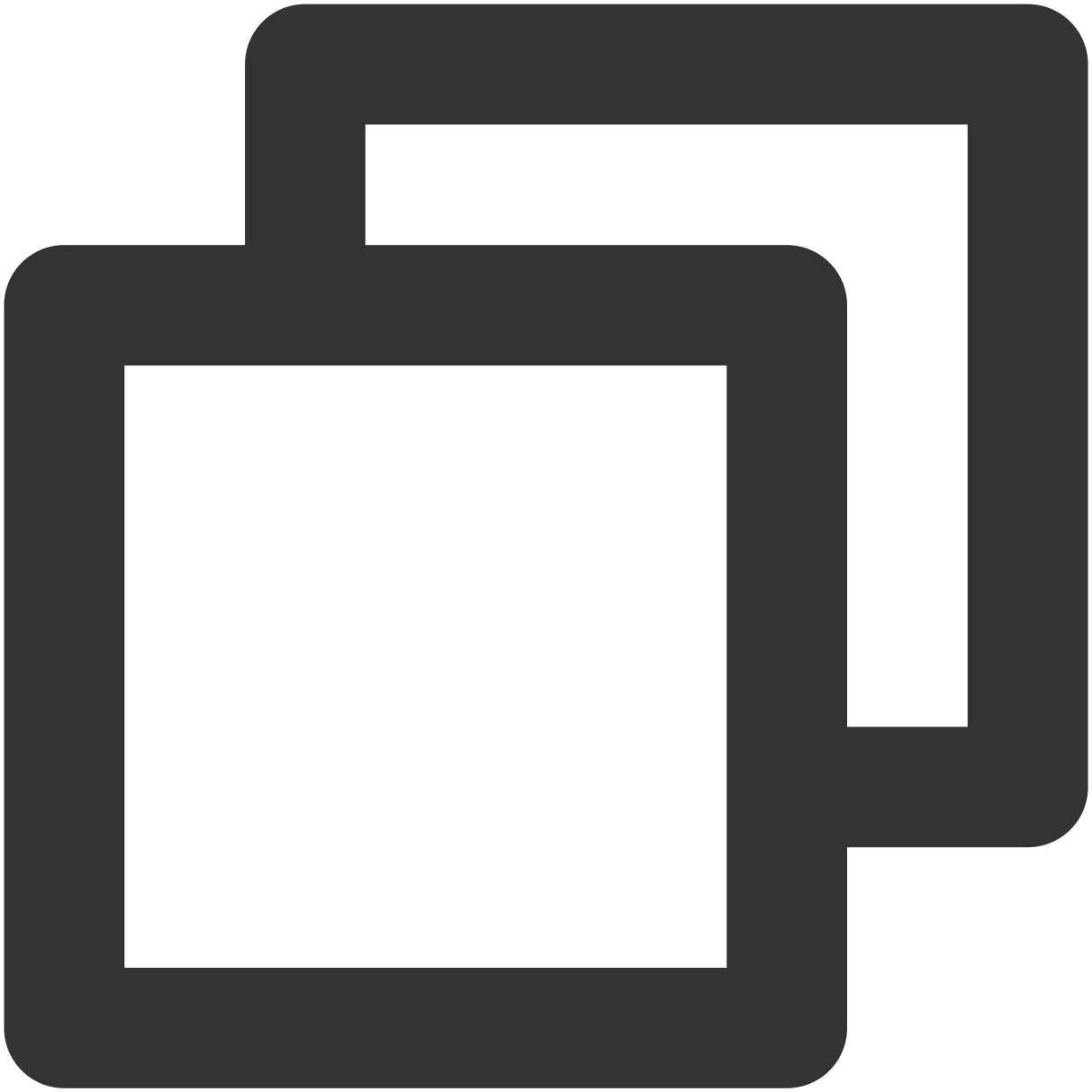
功能说明

用于查询文档审核任务结果。

注意：

COS Android SDK 版本需要大于等于 v5.8.7。

示例代码



```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
// 审核任务的 ID
String jobId = "iab1ca9fc8a3ed11ea834c525400863904";
GetDocumentAuditRequest request = new GetDocumentAuditRequest(bucket, jobId
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.getDocumentAuditAsync(request, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
        // result 查询文档审核任务的结果
        // 详细字段请查看api文档或者SDK源码
```

```
                GetDocumentAuditResult result = (GetDocumentAuditResult) cosResult;
            }

            @Override
            public void onFail(CosXmlRequest request, CosXmlClientException clientE
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

说明：

更多完整示例，请前往 GitHub 查看。

C SDK
快速入门

最近更新时间：2024-03-13 15:17:11

下载与安装

相关资源

对象存储的 XML C SDK 源码下载地址：XML C SDK。

演示示例 Demo 下载地址：XML C SDK Demo。

SDK 更新日志请参见 ChangeLog。

SDK 常见问题请参见：C SDK 常见问题。

说明

如果您在使用 SDK 时遇到函数或方法不存在等错误，请先将 SDK 升级到最新版再重试。

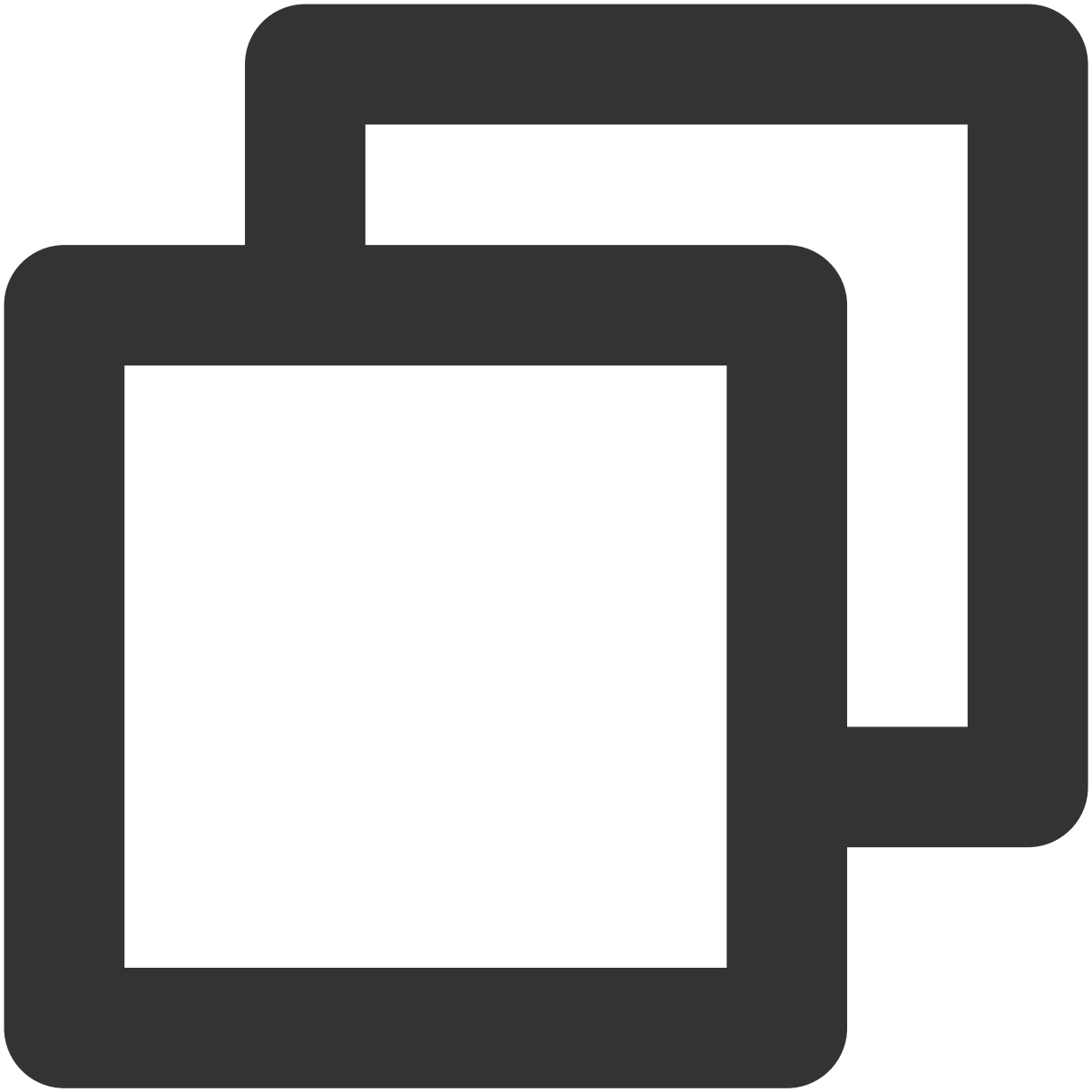
环境依赖

依赖库：libcurl apr apr-util minixml。

安装 SDK

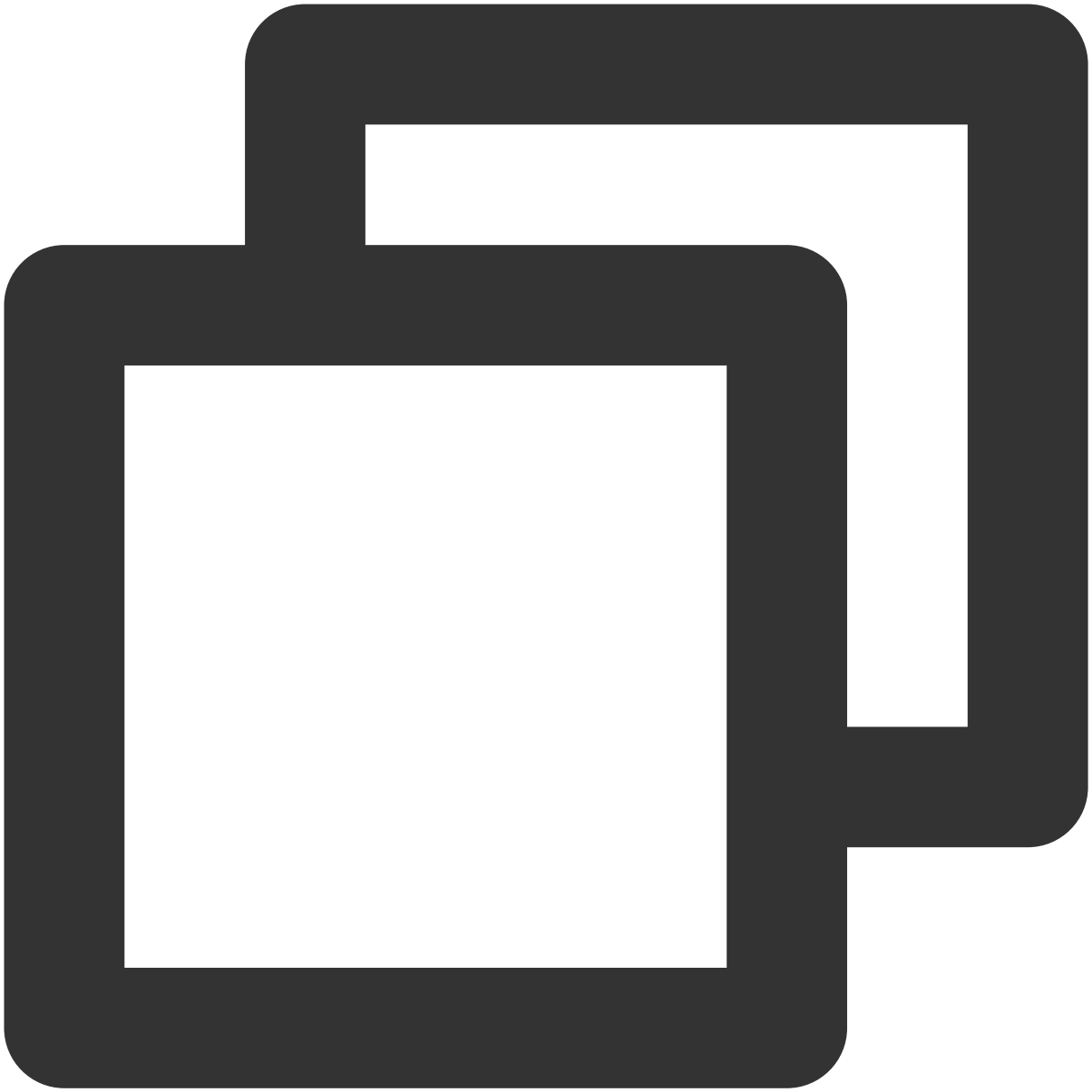
1. 安装 CMake 工具（建议 2.6.0 及以上版本），单击 这里 下载，安装方式如下：

版权所有：腾讯云计算（北京）有限责任公司



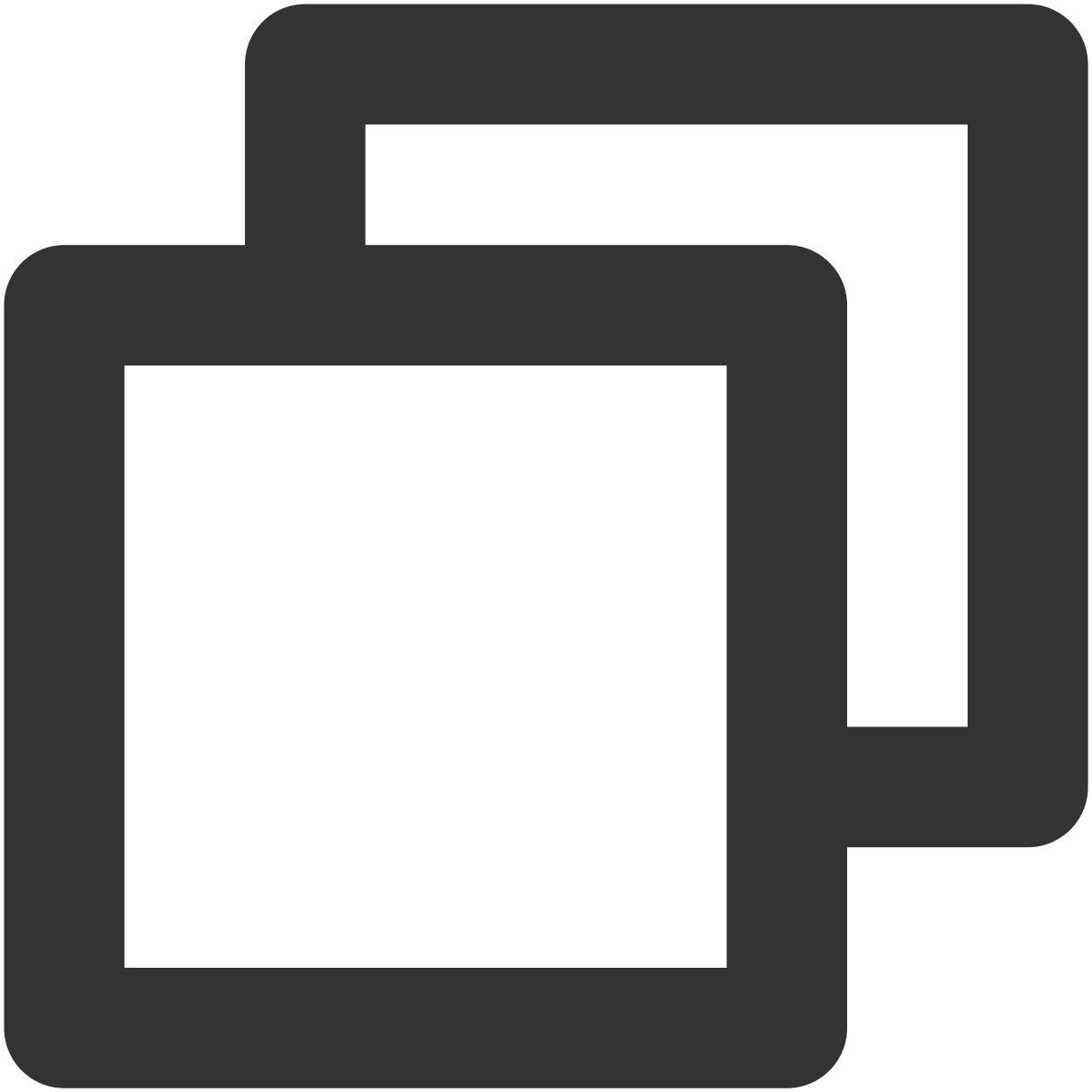
```
./configure
make
make install
```

2. 安装 libcurl（建议 7.32.0 及以上版本），单击 这里 下载，安装方式如下：



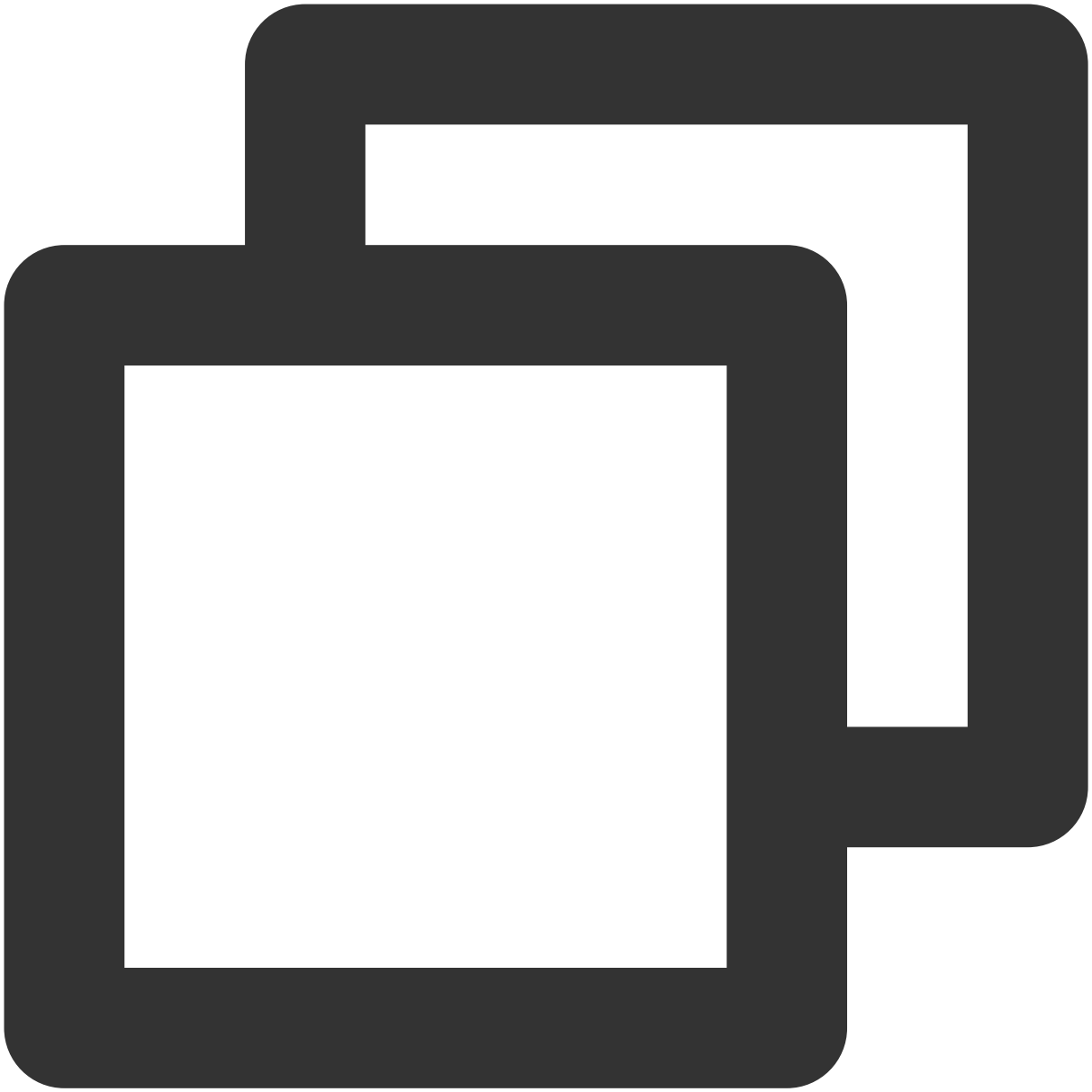
```
./configure
make
make install
```

3. 安装 apr（建议 1.5.2 及以上版本），单击 这里 下载，安装方式如下：



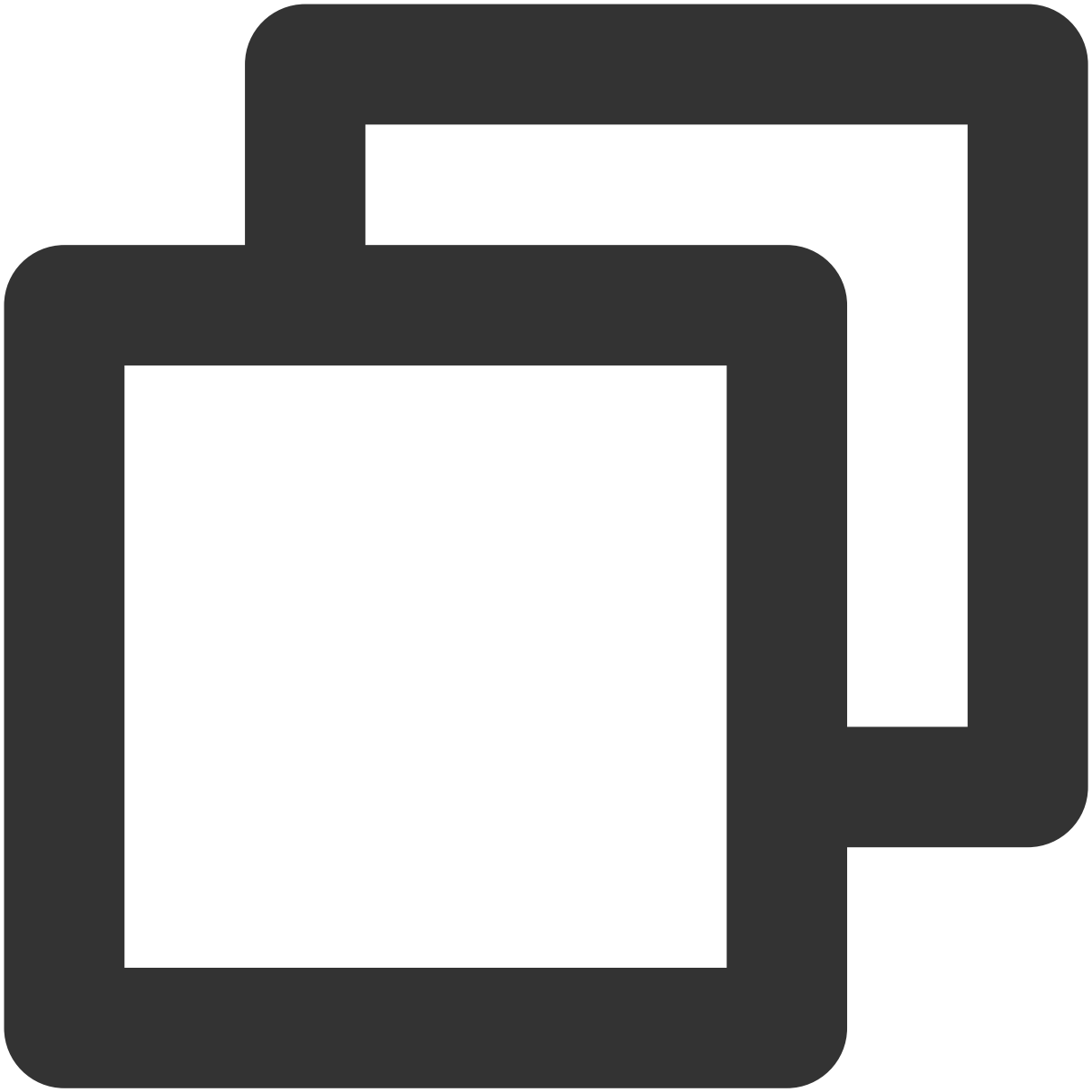
```
./configure
make
make install
```

4. 安装 apr-util（建议 1.5.4 及以上版本），单击 这里 下载，安装时需要指定 `--with-apr` 选项，安装方式如下：



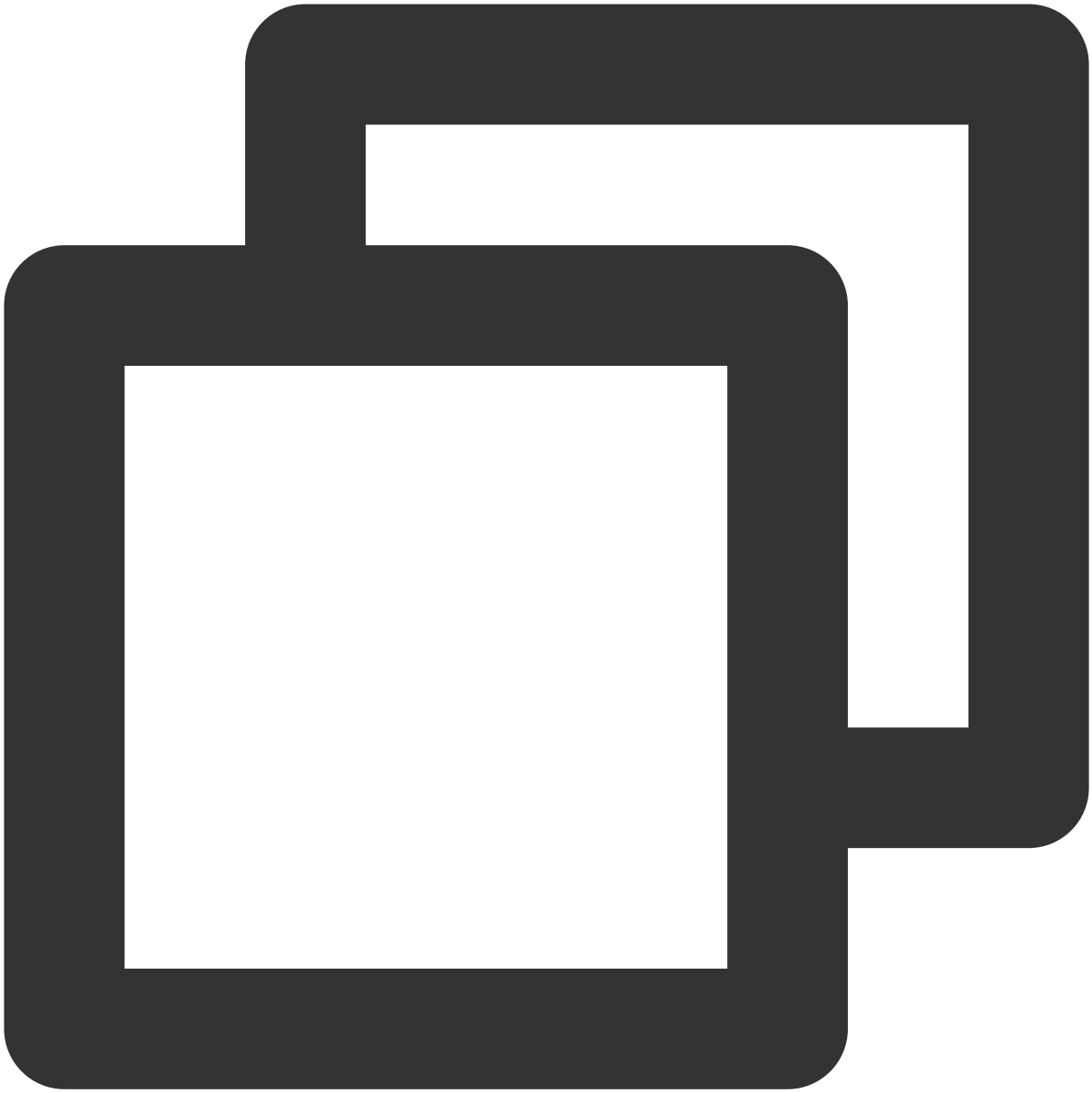
```
./configure --with-apr=/your/apr/install/path
make
make install
```

5. 安装 minixml（建议 2.8 - 2.12 版本），单击 这里 下载，安装方式如下：



```
./configure
make
make install
```

6. 编译 COS C SDK。下载 XML C SDK 源码，执行如下编译命令：



```
cmake .
make
make install
```

开始使用

下面为您介绍使用 XML C SDK 的一般流程。

1. 初始化 SDK。

2. 设置请求选项参数。关于 APPID、SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。

APPID 是申请腾讯云账号后，系统分配的账户标识之一。

access_key_id 与 access_key_secret 是账号 API 密钥。

endpoint 是 COS 访问域名信息，详情请参见 地域和访问域名 文档。例如，广州地域 endpoint 为 `cos.ap-guangzhou.myqcloud.com`，全球加速域名的 endpoint 为 `cos.accelerate.myqcloud.com`。endpoint可以加上http或者https，SDK 默认通过 http 访问 COS，如通过 https 访问广州地域的 endpoint 为 `https://cos.ap-guangzhou.myqcloud.com`。

is_cname 表示 endpoint 是否为自定义域名，如果is_cname设置成1，则 endpoint 表示自定义域名。

3. 设置 API 接口必需的参数。

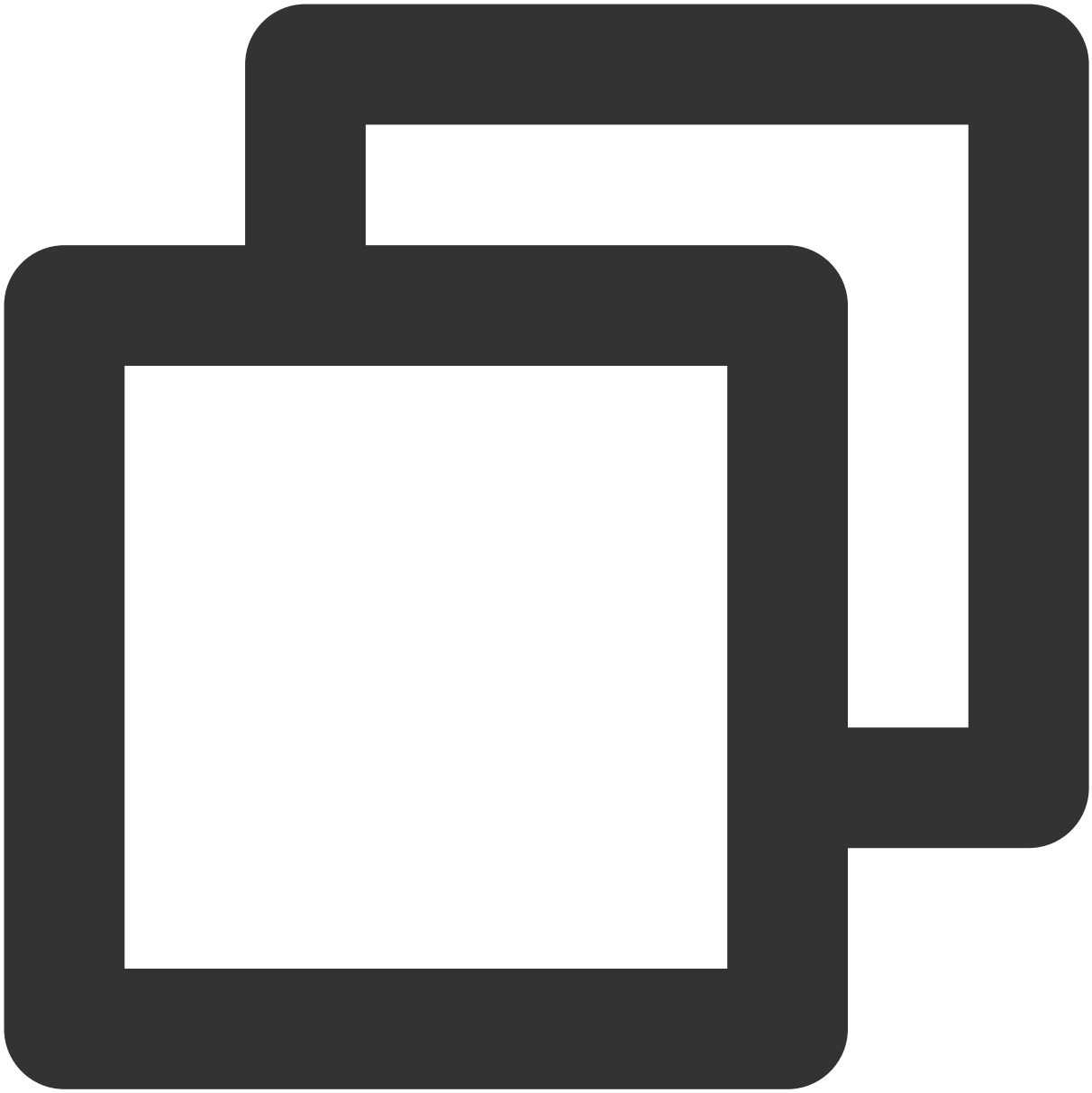
4. 调用 SDK API 发起请求并获得请求响应结果。

初始化

注意

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

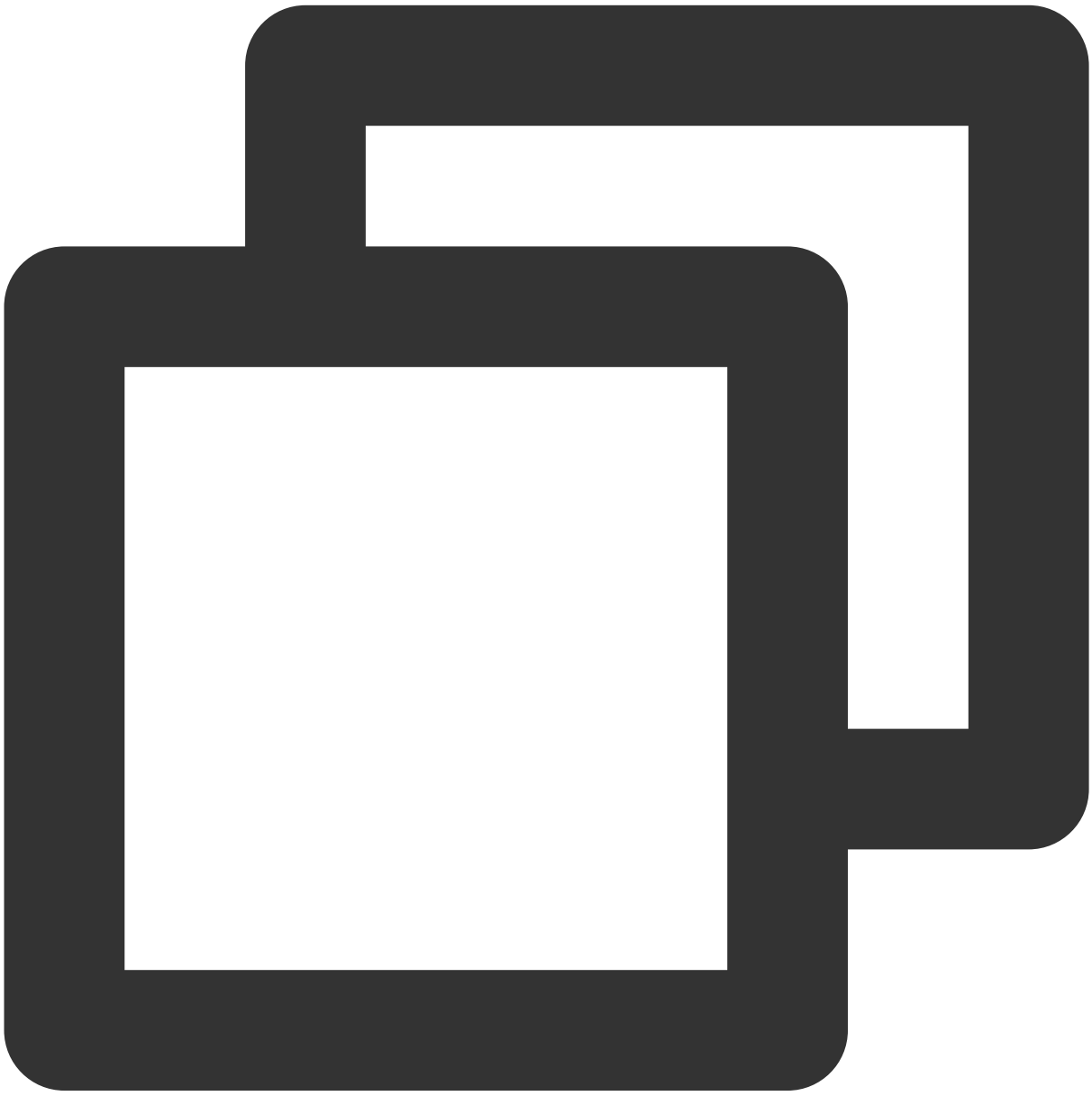
如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。



```
int main(int argc, char *argv[])
{
    /* 程序入口处调用 cos_http_io_initialize 方法，这个方法内部会做一些全局资源的初始化，涉及
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
        exit(1);
    }

    /* 调用 COS SDK 的接口上传或下载文件 */
    /* ... 用户逻辑代码，这里省略 */

    /* 程序结束前，调用 cos_http_io_deinitialize 方法释放之前分配的全局资源 */
```

```
    cos_http_io_deinitialize();
    return 0;
}
```

初始化请求选项



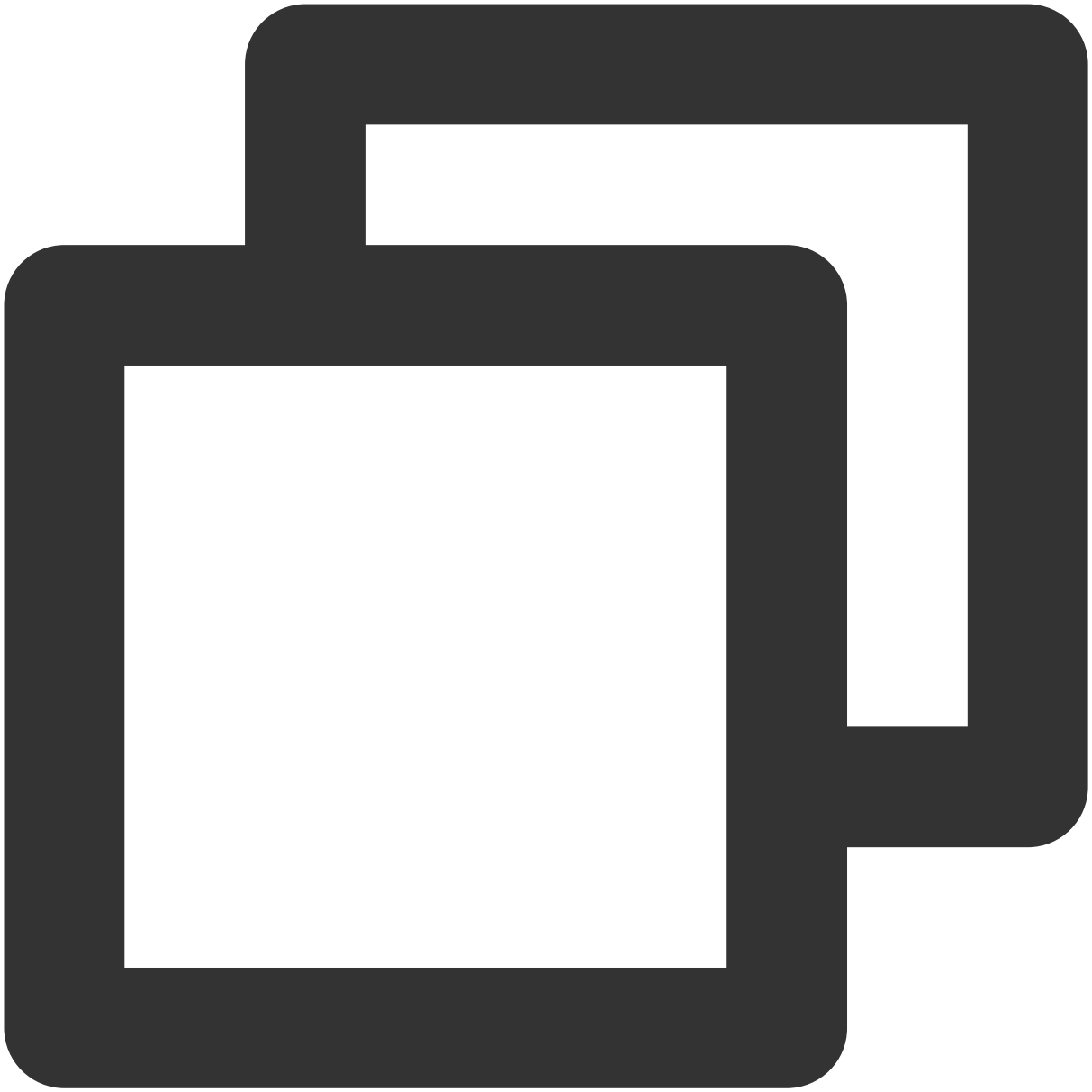
```
/* 等价于 apr_pool_t，用于内存管理的内存池，实现代码在 apr 库中 */
cos_pool_t *pool;
cos_request_options_t *options;
```

```
/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&pool, NULL);

/* 创建并初始化 options，这个参数内部主要包括 endpoint,access_key_id,acces_key_secret, is
 * options的内存是由pool分配的，后续释放掉pool后，options的内存也相当于释放掉了，不再需要单独释
 */
options = cos_request_options_create(pool);
options->config = cos_config_create(options->pool);

/* cos_str_set 是用 char* 类型的字符串初始化 cos_string_t 类型*/
cos_str_set(&options->config->endpoint, "<用户的Endpoint>");                //Endpoint
cos_str_set(&options->config->access_key_id, "<用户的SecretId>");           //用户注册 (
cos_str_set(&options->config->access_key_secret, "<用户的SecretKey>");      //用户注册 (
cos_str_set(&options->config->sts_token, "<用户的StsToken>");               //临时密钥的
cos_str_set(&options->config->appid, "<用户的AppId>");                      //用户注册 (

/* 可以通过设置 sts_token 来使用临时密钥，当使用临时密钥时，access_key_id 和access_key_secr
//cos_str_set(&options->config->sts_token, "MyTokenString");
/* 是否使用了 CNAME */
options->config->is_cname = 0;
/* 使用自定义域名访问 COS */
/*
options->config->is_cname = 1;
cos_str_set(&options->config->endpoint, "<自定义域名>");
*/

/* 用于设置网络相关参数，例如超时时间等*/
options->ctl = cos_http_controller_create(options->pool, 0);

/* 用于设置上传请求是否自动添加 Content-MD5 头部，enable 为 COS_FALSE 时上传请求将不自动添加
cos_set_content_md5_enable(options->ctl, COS_FALSE);

/* 用于设置请求路由地址和端口，一般情况下无需设置此参数，请求将按域名解析结果路由 */
//cos_set_request_route(options->ctl, "192.168.12.34", 80);
```

说明

临时密钥生成和使用可参见 临时密钥生成及使用指引。

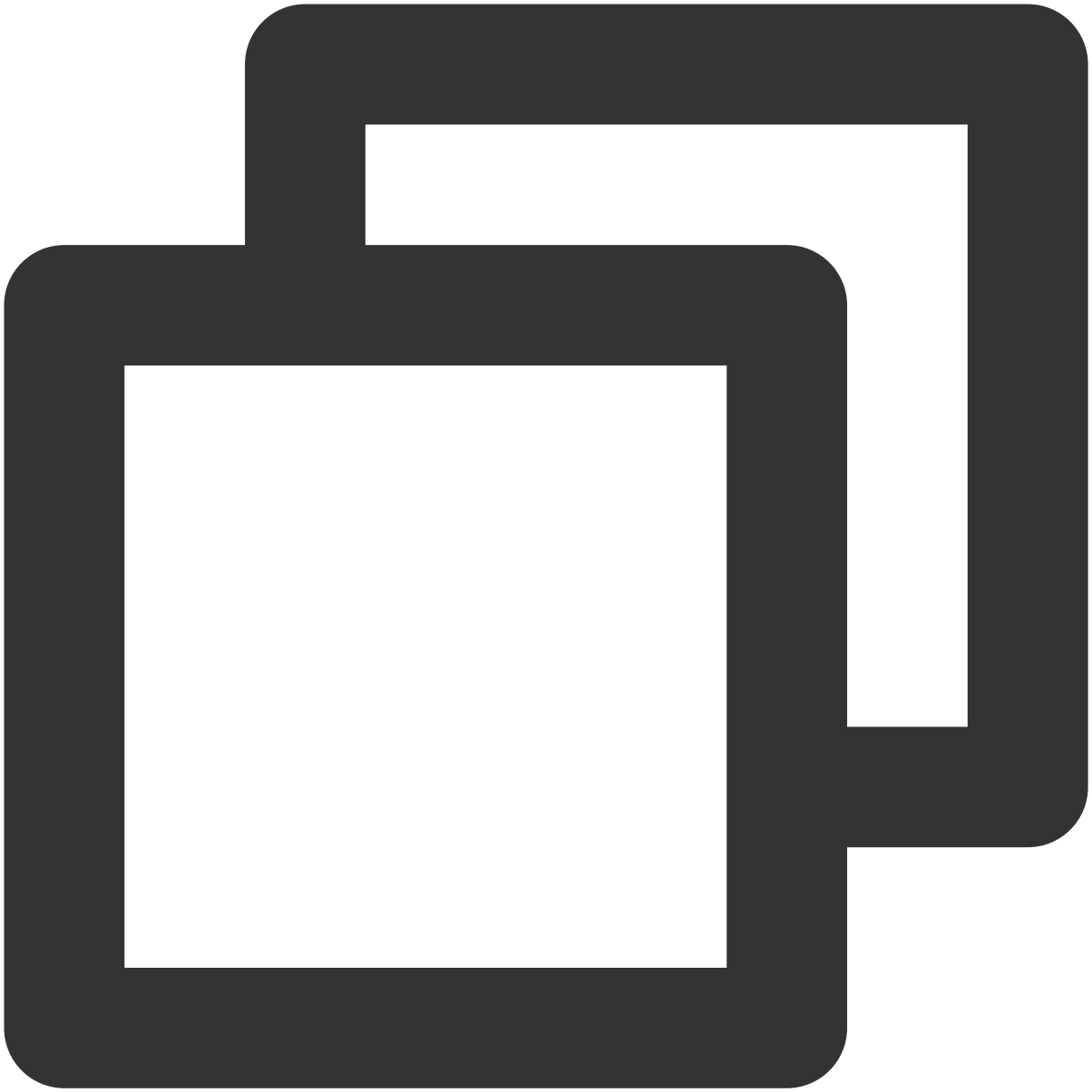
创建存储桶



```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_acl_e cos_acl = COS_ACL_PRIVATE;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&p, NULL);
```

```
/* 创建并初始化 options，这个参数内部主要包括endpoint,access_key_id,acces_key_secret, is_
 * options 的内存是由 pool 分配的，后续释放掉 pool 后，options的内存也相当于释放掉了，不再需要
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* 设置 appid, endpoint, access_key_id, acces_key_secret, is_cname, curl参数等配置信息
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* 存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式 */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* 调用 API 创建存储桶 */
s = cos_create_bucket(options, &bucket, cos_acl, &resp_headers);
if (cos_status_is_ok(s)) {
        printf("create bucket succeeded\\n");
} else {
        printf("create bucket failed\\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

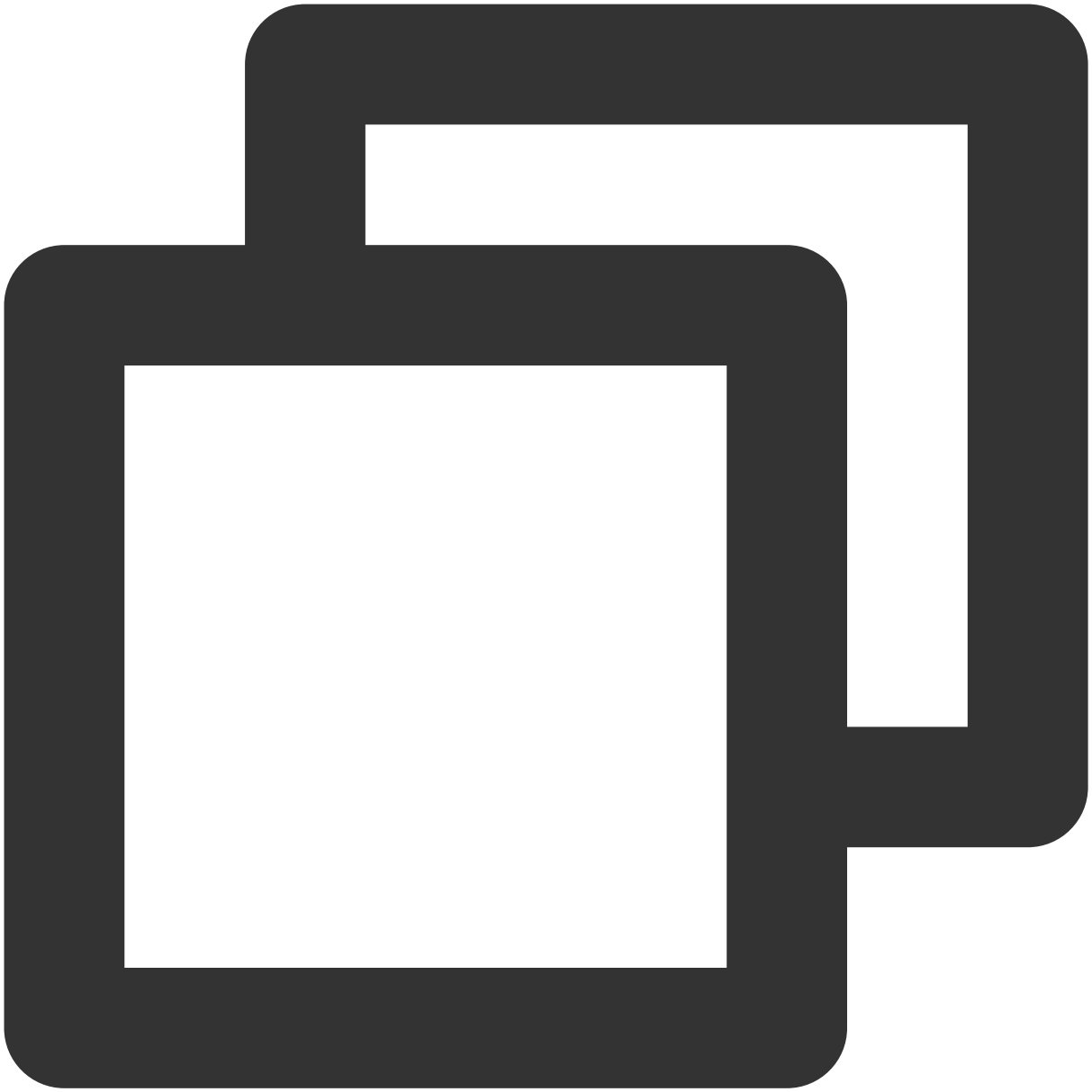
查询对象列表



```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_list_object_params_t *list_params = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&p, NULL);
```

```
/* 创建并初始化 options，这个参数内部主要包括 endpoint,access_key_id,acces_key_secret, is
 * options的内存是由pool分配的，后续释放掉 pool 后，options 的内存也相当于释放掉了，不再需要单
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* 设置 appid, endpoint, access_key_id, acces_key_secret, is_cname, curl 参数等配置信息
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* 存储桶的命名格式为 BucketName-APPID, , 此处填写的存储桶名称必须为此格式 */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* 调用 API 查询对象列表 */
list_params = cos_create_list_object_params(p);
cos_str_set(&list_params->encoding_type, "url");
s = cos_list_object(options, &bucket, list_params, &resp_headers);
if (cos_status_is_ok(s)) {
        printf("list object succeeded\\n");
} else {
        printf("list object failed\\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

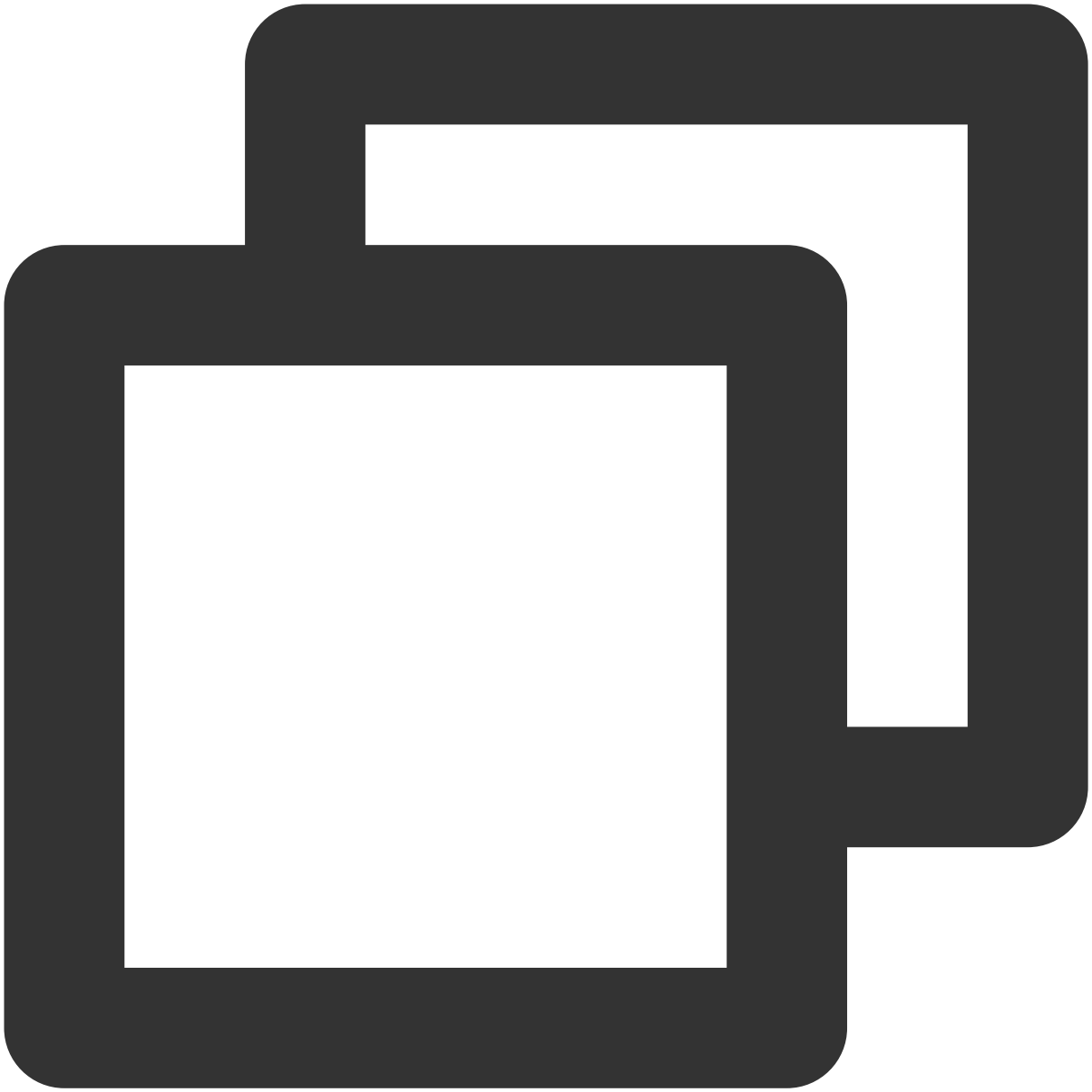
上传对象



```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&p, NULL);
```

```
/* 创建并初始化 options，这个参数内部主要包括 endpoint,access_key_id,acces_key_secret, is
 * options的内存是由pool分配的，后续释放掉pool后，options的内存也相当于释放掉了，不再需要单独释
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* 设置 appid, endpoint, access_key_id, acces_key_secret, is_cname, curl 参数等配置信息
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* 存储桶的命名格式为 BucketName-APPID, , 此处填写的存储桶名称必须为此格式 */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* 调用 API 上传对象 */
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_put_object_from_file(options, &bucket, &object, &file, NULL, &resp_headers)
if (cos_status_is_ok(s)) {
        printf("put object succeeded\\n");
} else {
        printf("put object failed\\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

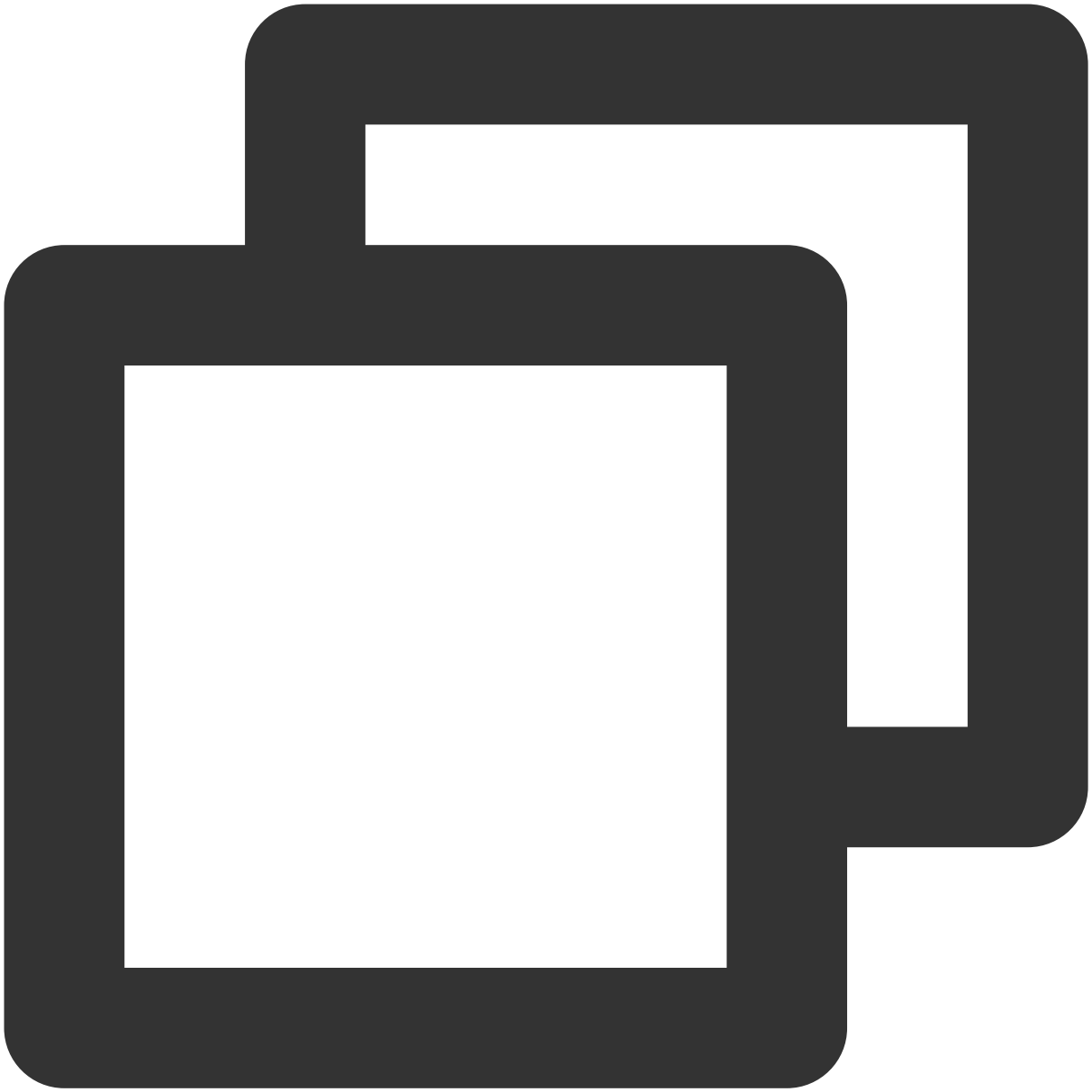
下载对象



```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&p, NULL);
```

```
/* 创建并初始化 options，这个参数内部主要包括 endpoint,access_key_id,acces_key_secret, is
 * options的内存是由pool分配的，后续释放掉pool后，options的内存也相当于释放掉了，不再需要单独释
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* 设置 appid, endpoint, access_key_id, acces_key_secret, is_cname, curl 参数等配置信息
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* 存储桶的命名格式为 BucketName-APPID, , 此处填写的存储桶名称必须为此格式 */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* 调用 API 下载对象 */
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_get_object_to_file(options, &bucket, &object, NULL, NULL, &file, &resp_head
if (cos_status_is_ok(s)) {
        printf("get object succeeded\\n");
} else {
        printf("get object failed\\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

删除对象



```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

/* 重新创建一个新的内存池，第二个参数是 NULL，表示没有继承自其它内存池 */
cos_pool_create(&p, NULL);
```

```
/* 创建并初始化 options，这个参数内部主要包括 endpoint,access_key_id,acces_key_secret, is
 * options的内存是由pool分配的，后续释放掉 pool 后，options 的内存也相当于释放掉了，不再需要单
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* 设置 appid, endpoint, access_key_id, acces_key_secret, is_cname, curl参数等配置信息
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* 存储桶的命名格式为 BucketName-APPID, , 此处填写的存储桶名称必须为此格式 */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* 调用 API 删除对象 */
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_delete_object(options, &bucket, &object, &resp_headers);
if (cos_status_is_ok(s)) {
        printf("delete object succeeded\\n");
} else {
        printf("delete object failed\\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

图片处理
图片持久化处理

最近更新时间：2024-03-13 15:17:11

简介

本文档提供关于图片持久化处理的 API 概览以及 SDK 示例代码。

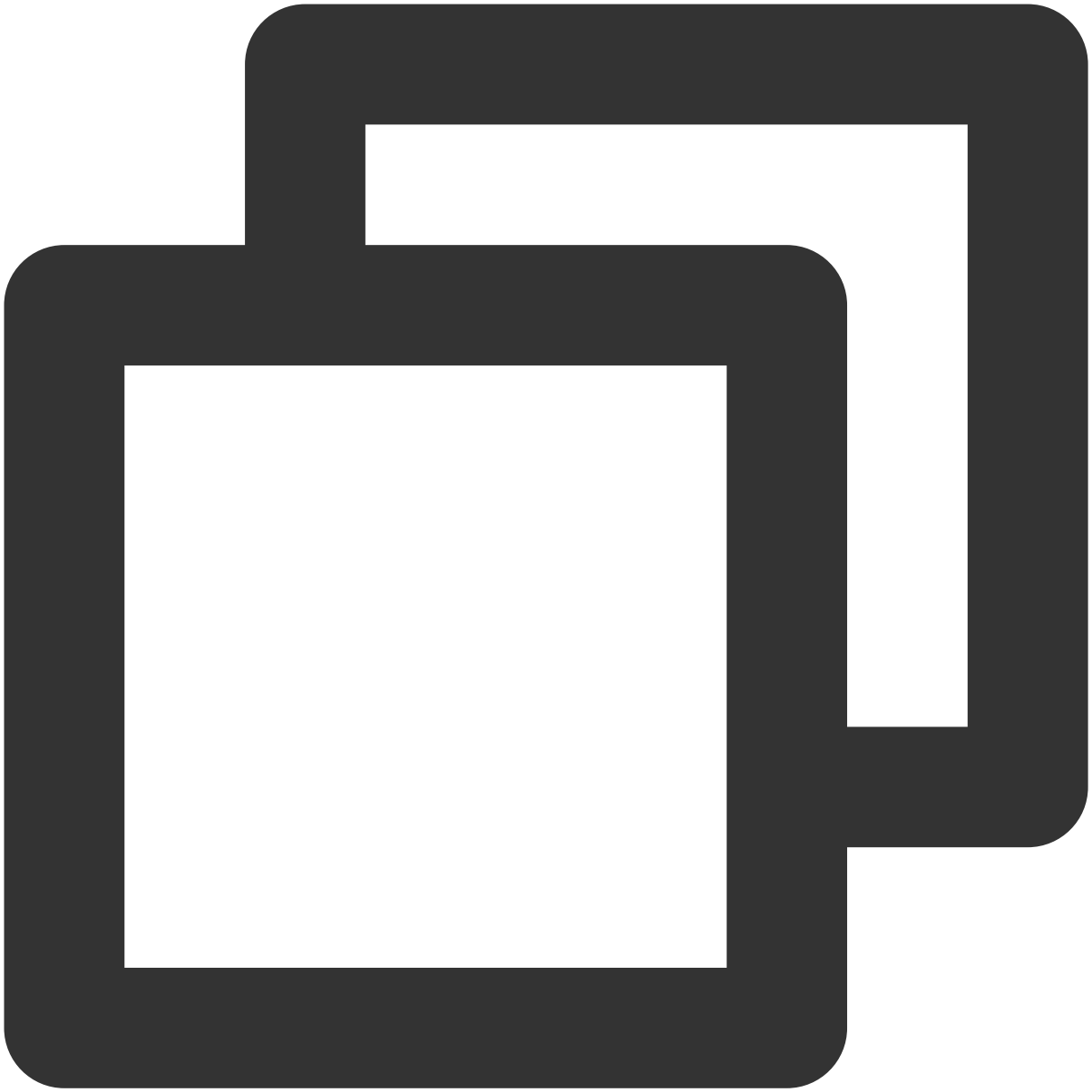
API	说明
图片持久化处理	对象存储提供的上传时处理功能可以帮助使用者在上传时实现图片处理。此外能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到对象存储（Cloud Object Storage，COS）

上传时处理

功能说明

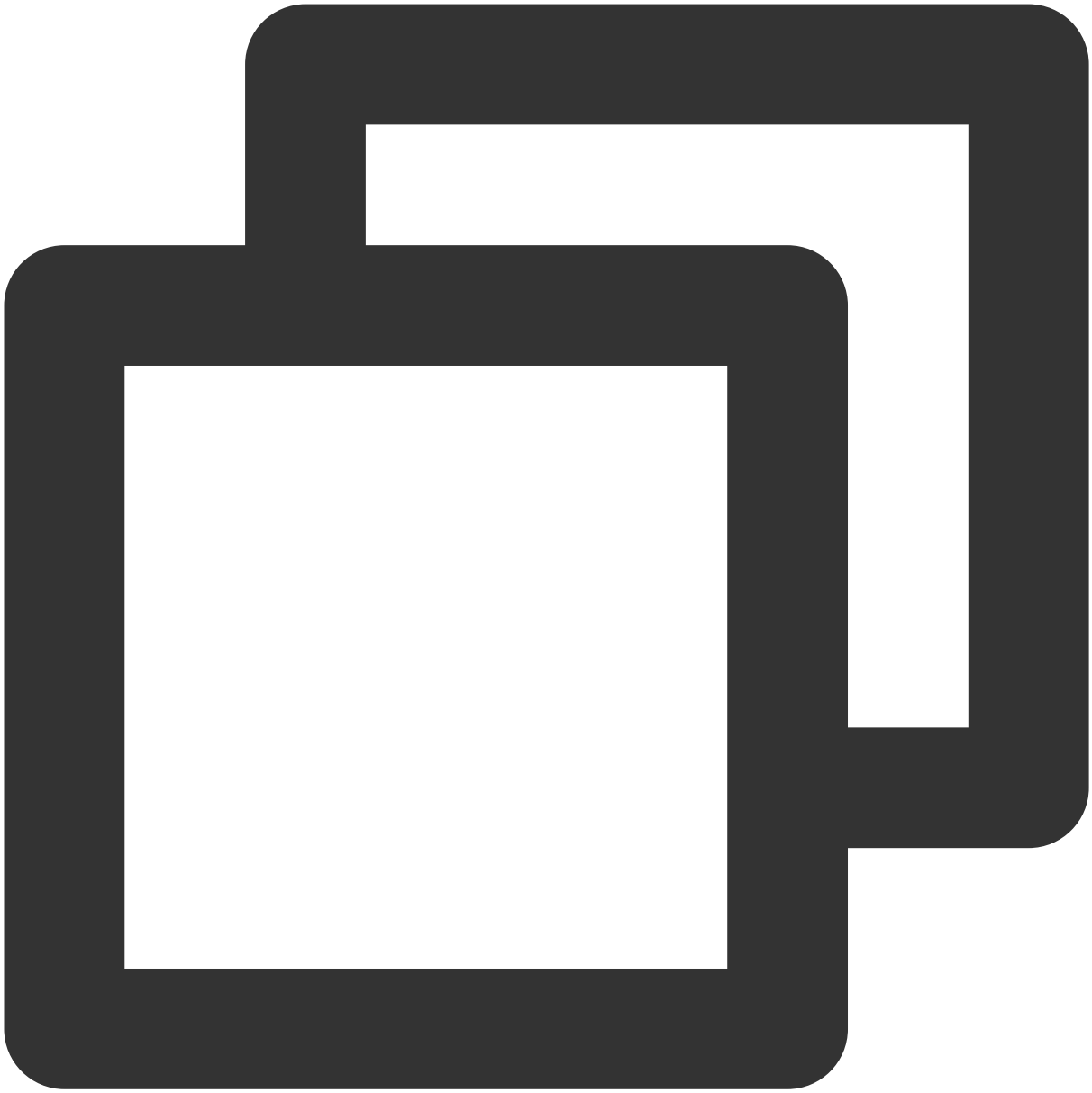
COS 提供的上传时处理功能可以帮助使用者在上传时实现图片处理。您只需要在请求包头部中加入 Pic-Operations 项并设置好相应参数，即可在图片上传时实现相应的图片处理，并可将原图和处理结果存入到 COS。目前支持20M以内文件处理。

方法原型



```
cos_status_t *ci_put_object_from_file(const cos_request_options_t *options,
                                      const cos_string_t *bucket,
                                      const cos_string_t *object,
                                      const cos_string_t *filename,
                                      cos_table_t *headers,
                                      cos_table_t **resp_headers,
                                      ci_operation_result_t **results)
```

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_put_image_process()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *headers = NULL;
    cos_table_t *resp_headers;
    ci_operation_result_t *results = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    //上传时处理
```

```
    headers = cos_table_make(p, 1);
    apr_table_addn(headers, "pic-operations", "{\\"is_pic_info\\":1,\\"rules\\":[{\
    cos_str_set(&file, "test.jpg");
    cos_str_set(&object, "test.jpg");
    s = ci_put_object_from_file(options, &bucket, &object, &file, headers, &resp_he
    if (!cos_status_is_ok(s)) {
        printf("put object failed: %s\\n", s->req_id);
    }
    printf("origin key: %s\\n", results->origin.key.data);
    printf("process key: %s\\n", results->object.key.data);

    //销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_put_image_process();

    cos_http_io_deinitialize();

    return 0;
}
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String

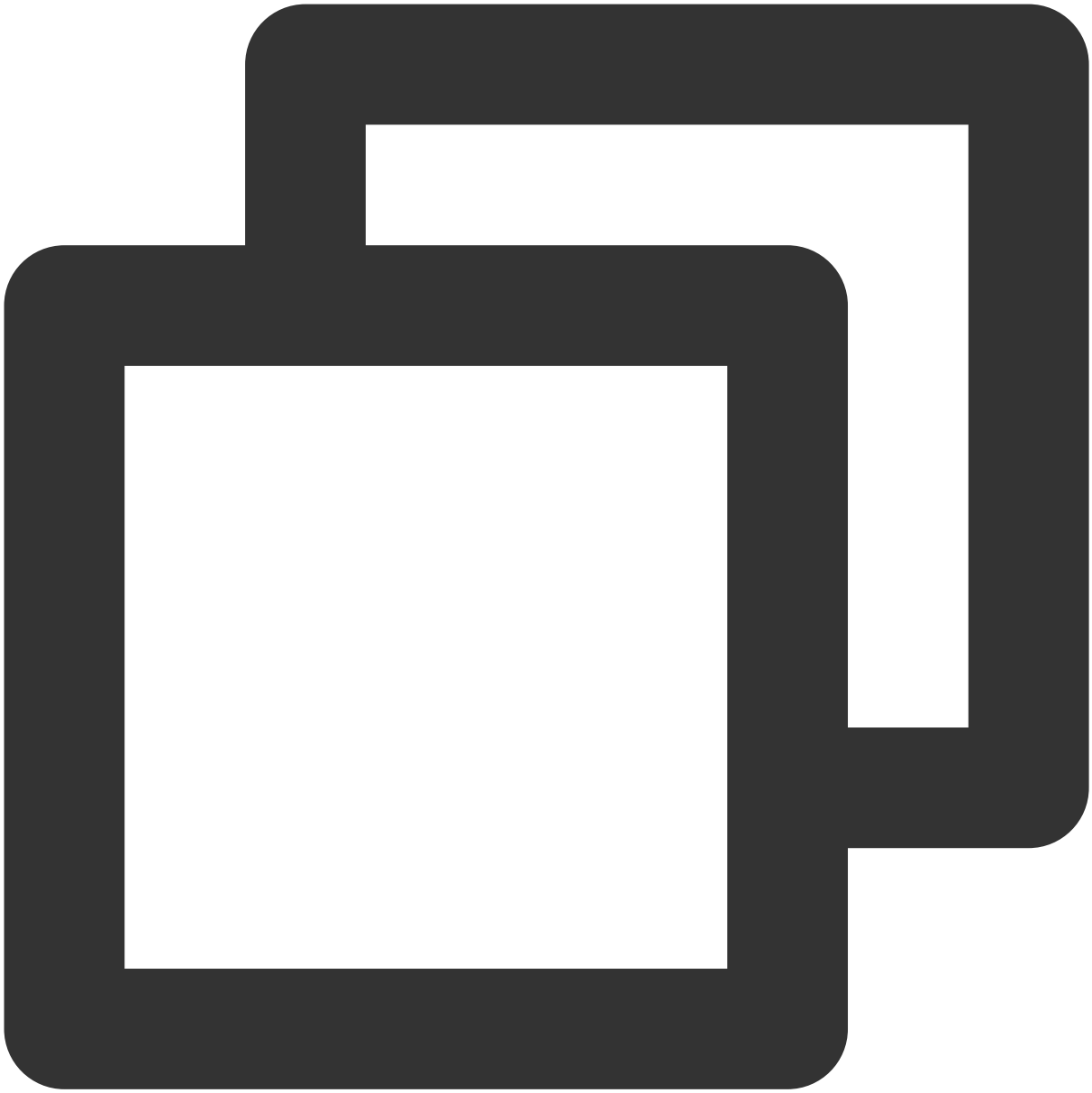
object	Object 名称	String
filename	Object 本地保存文件名称	String
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
results	云上数据处理返回信息	Struct

云上数据处理

功能说明

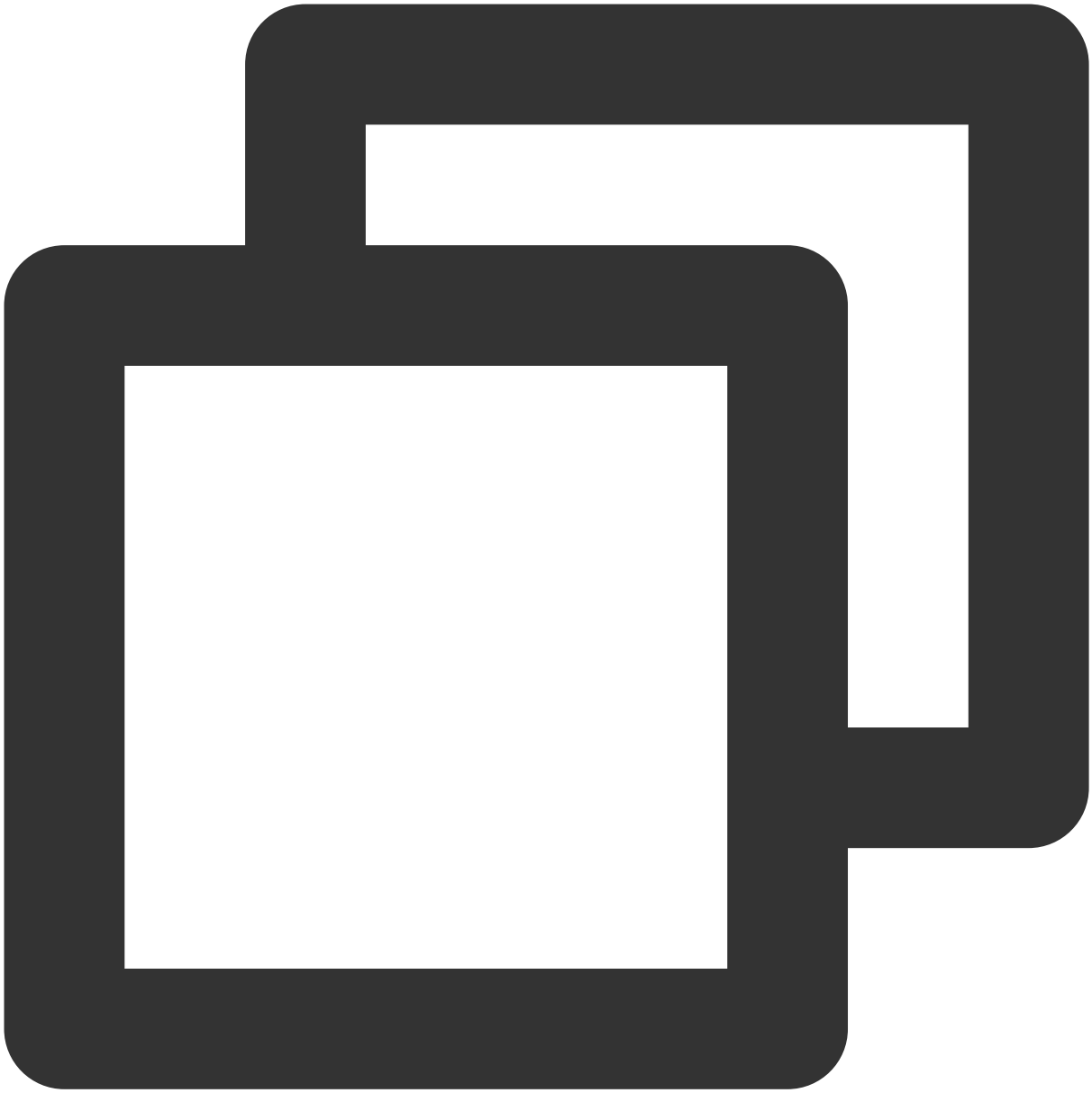
该接口能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

方法原型



```
cos_status_t *ci_image_process(const cos_request_options_t *options,
                               const cos_string_t *bucket,
                               const cos_string_t *object,
                               cos_table_t *headers,
                               cos_table_t **resp_headers,
                               ci_operation_result_t *results);
```

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_ci_image_process()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_table_t *headers = NULL;
    cos_table_t *resp_headers;
    ci_operation_result_t *results = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    cos_str_set(&object, "test.jpg");
    headers = cos_table_make(p, 1);
```

```
    apr_table_addn(headers, "pic-operations", "{\\"is_pic_info\\":1,\\"rules\\":[{\

    s = ci_image_process(options, &bucket, &object, headers, &resp_headers, &result
    if (!cos_status_is_ok(s)) {
        printf("ci image process fail, req_id:%s\\n", s->req_id);
    }
    printf("origin key: %s\\n", results->origin.key.data);
    printf("process key: %s\\n", results->object.key.data);
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_image_process();

    cos_http_io_deinitialize();

    return 0;
}
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，Bucket 的命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	string
object	Object 名称	string
headers	COS 请求附加头域，需要自行添加 Pic-Operations 万象处理头部	Struct

resp_headers	返回 HTTP 响应消息的头域	Struct
results	云上数据处理返回信息	Struct

版权所有：腾讯云计算（北京）有限责任公司

基础图片处理

最近更新时间：2024-03-13 15:17:16

简介

本文档提供关于基础图片处理的 API 概览以及 SDK 示例代码。

API	操作描述
缩放	对图片进行缩小或放大
裁剪	对图片进行裁剪，包括普通裁剪、缩放裁剪、内切圆裁剪、圆角裁剪和人脸智能裁剪
旋转	对图片进行旋转，包括普通旋转和自适应旋转
格式转换	对图片进行格式转换、gif 格式优化、渐进显示
质量变换	对图片质量进行调节
高斯模糊	对图片进行模糊处理
锐化	对图片进行锐化
图片水印	对图片进行水印处理
文字水印	对图片进行实时文字水印处理
获取图片基本信息	查询图片基本信息，包括格式、长、宽等
获取图片 EXIF	查询 EXIF 信息
获取图片主色调	查询图片主色调信息
去除元信息	去除图片元信息，包括 exif 信息
快速缩略模板	通过图片处理模板，生成相应的缩略图
管道操作符	实现对图片按顺序进行多种处理

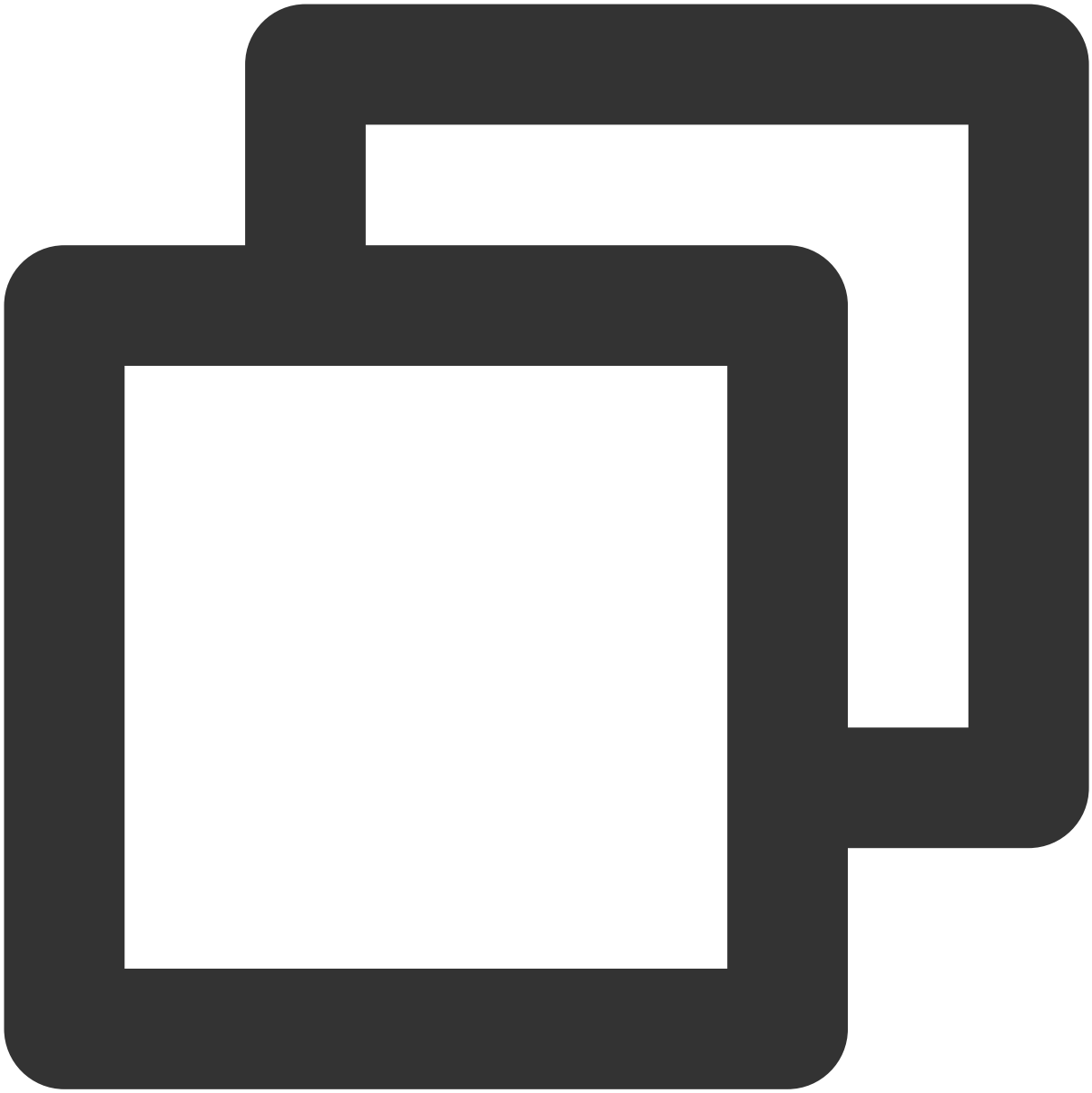
缩放

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供图片缩放功能。

版权所有：腾讯云计算（北京）有限责任公司

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
```

```
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_image_scale()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *resp_headers;
    cos_table_t *params = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
```

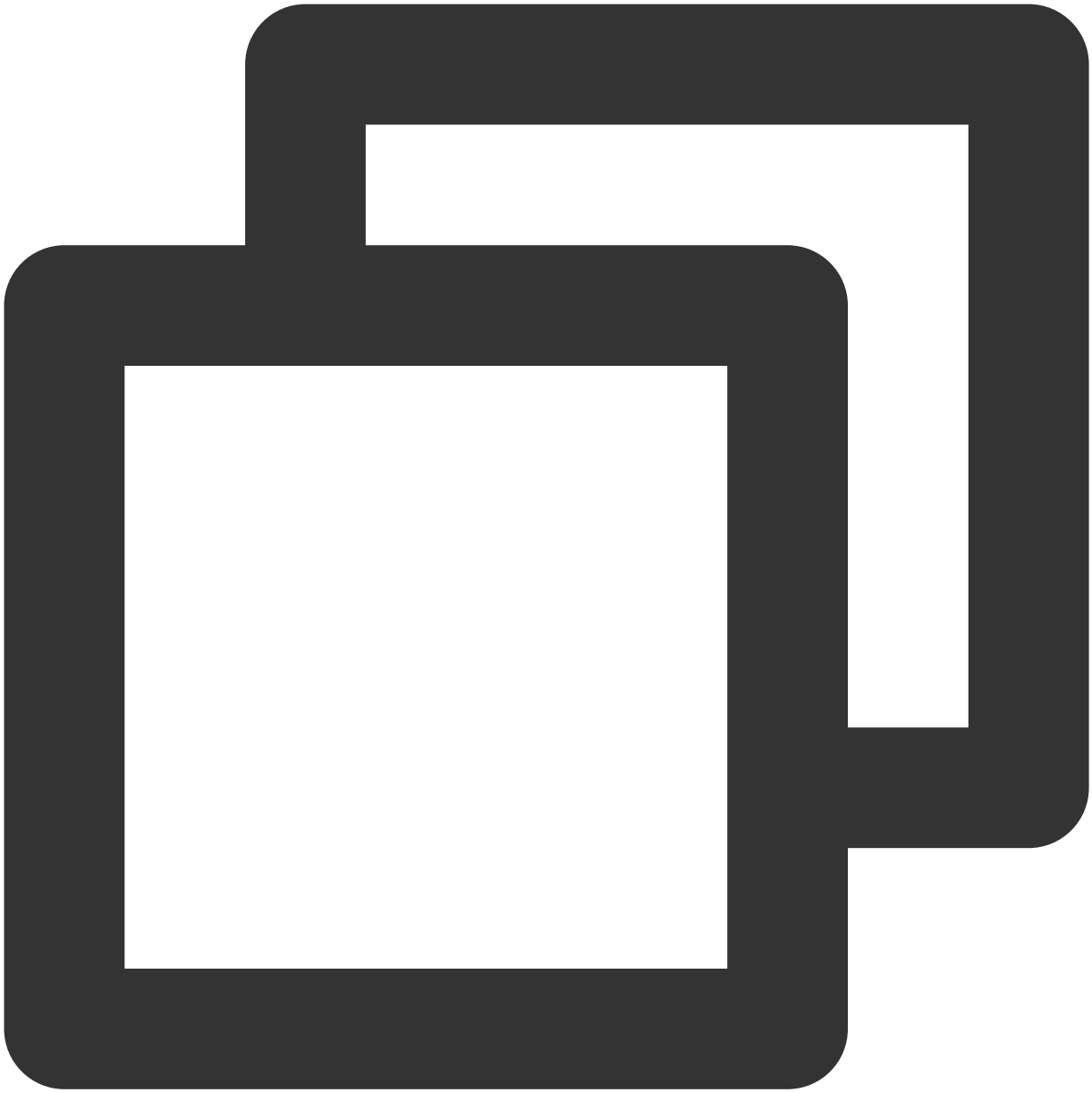
```
    params = cos_table_make(p, 1);
    apr_table_addn(params, "imageMogr2/thumbnail/!50p", "");
    cos_str_set(&file, "test.jpg");
    cos_str_set(&object, "test.jpg");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, params, &file, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("cos_get_object_to_file fail, req_id:%s\\n", s->req_id);
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_image_scale();

    cos_http_io_deinitialize();

    return 0;
}
```

裁剪

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供裁剪功能，包括普通裁剪、缩放裁剪、内切圆裁剪、圆角裁剪和人脸智能裁剪。

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

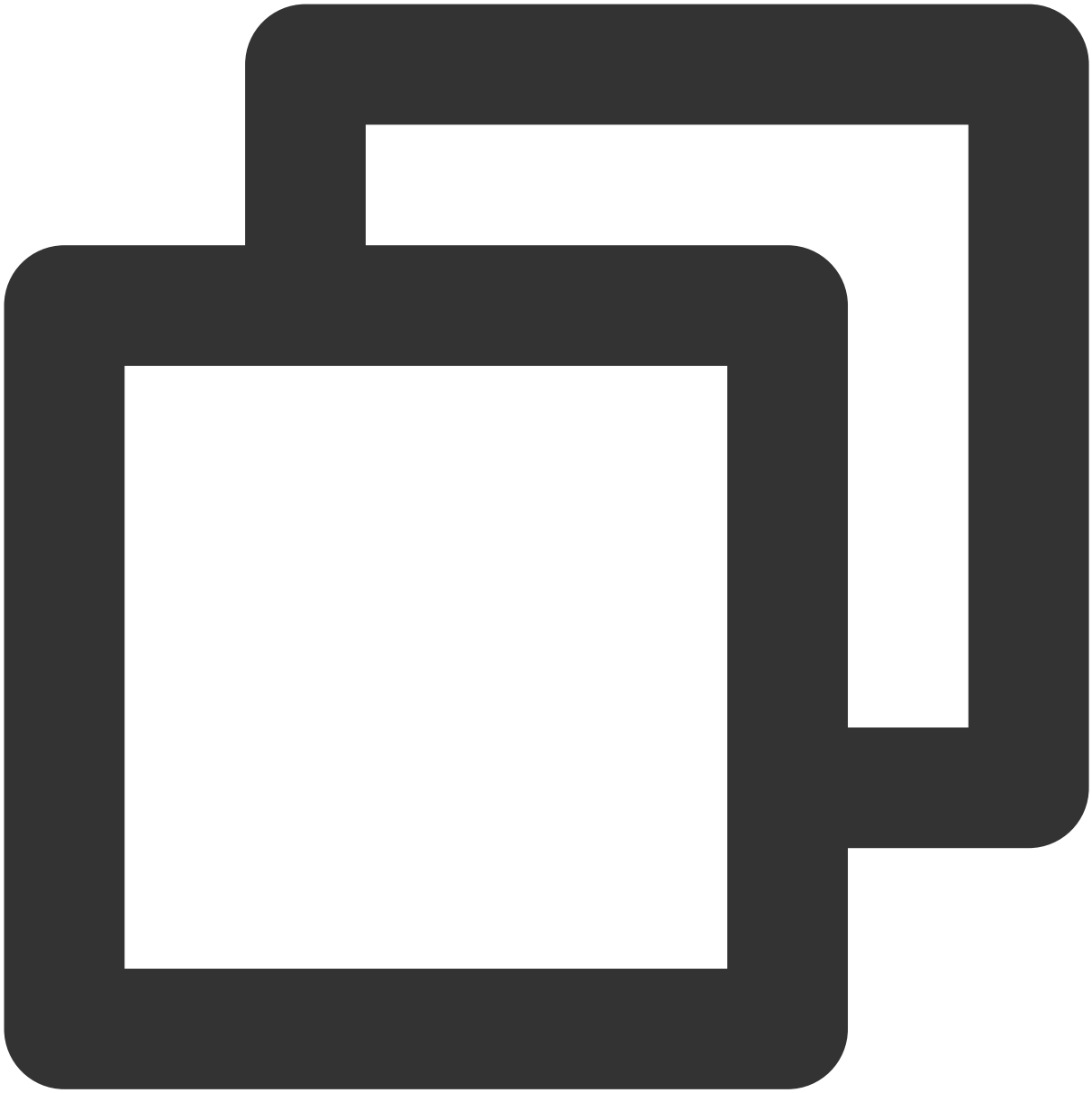
```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_image_cut()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *resp_headers;
    cos_table_t *params = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    params = cos_table_make(p, 1);
    apr_table_addn(params, "imageMogr2/cut/600x600x100x10", "");
```

```
    cos_str_set(&file, "test.jpg");
    cos_str_set(&object, "test.jpg");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, params, &file, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("cos_get_object_to_file fail, req_id:%s\\n", s->req_id);
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_image_cut();

    cos_http_io_deinitialize();

    return 0;
}
```

其他基础图片处理

功能说明

基础图片处理统一使用上述方法完成，其他基础图片处理只需要修改 operation 参数的值即可。例如，如下样例：将图片顺时针旋转90度。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_image_rotate()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *resp_headers;
    cos_table_t *params = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    params = cos_table_make(p, 1);
    apr_table_addn(params, "imageMogr2/rotate/90", "");
```

```
    cos_str_set(&file, "test.jpg");
    cos_str_set(&object, "test.jpg");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, params, &file, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("cos_get_object_to_file fail, req_id:%s\\n", s->req_id);
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_image_rotate();

    cos_http_io_deinitialize();

    return 0;
}
```

图片高级压缩

最近更新时间：2024-03-13 15:17:16

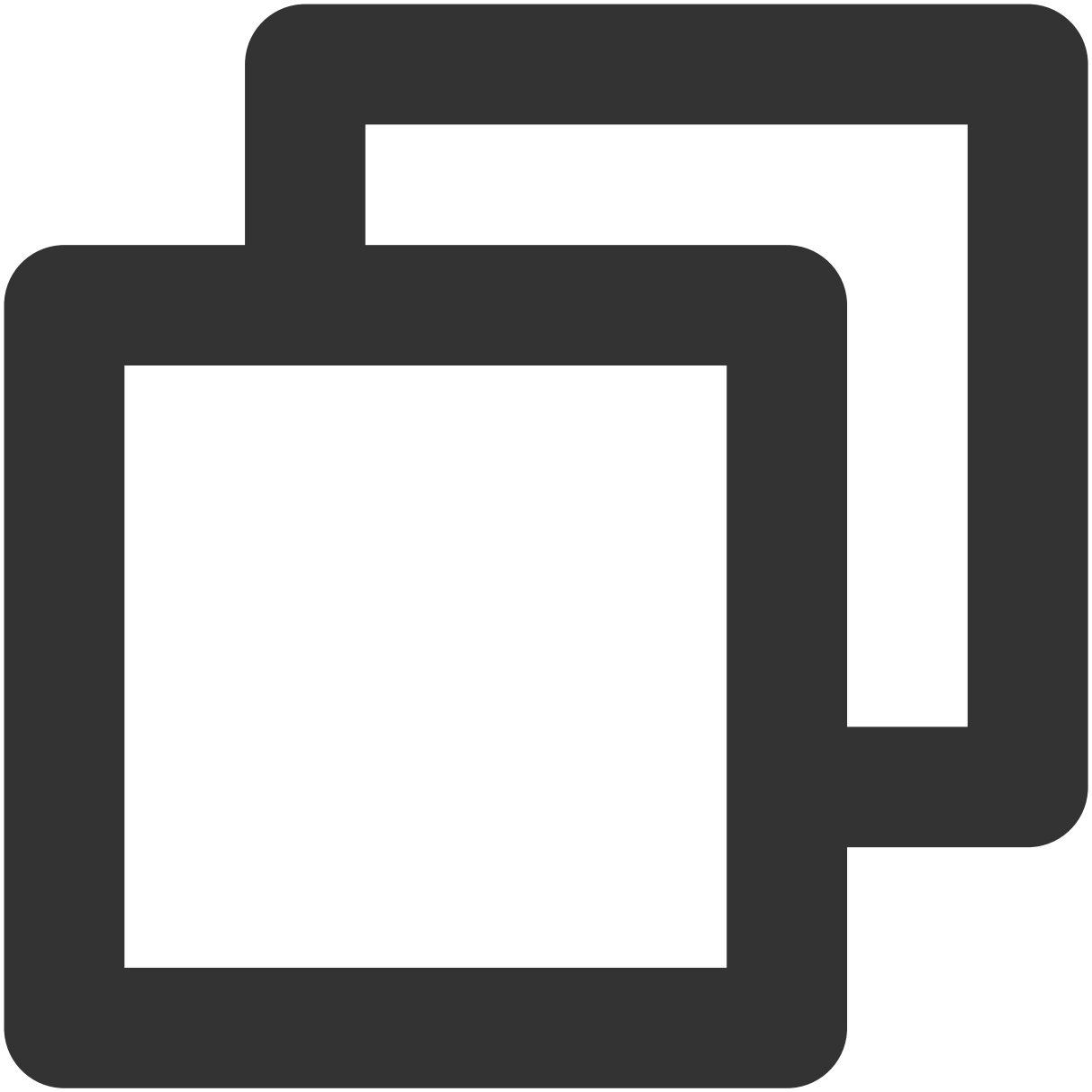
简介

本文档提供关于图片高级压缩的 API 概览以及 SDK 示例代码。

API	操作描述
图片高级压缩	图片高级压缩可以更加高效地将图片转换为 TPG 或 HEIF 等高压缩比格式，有效降低图片传输链路及加载耗时，降低带宽及流量成本

方法原型

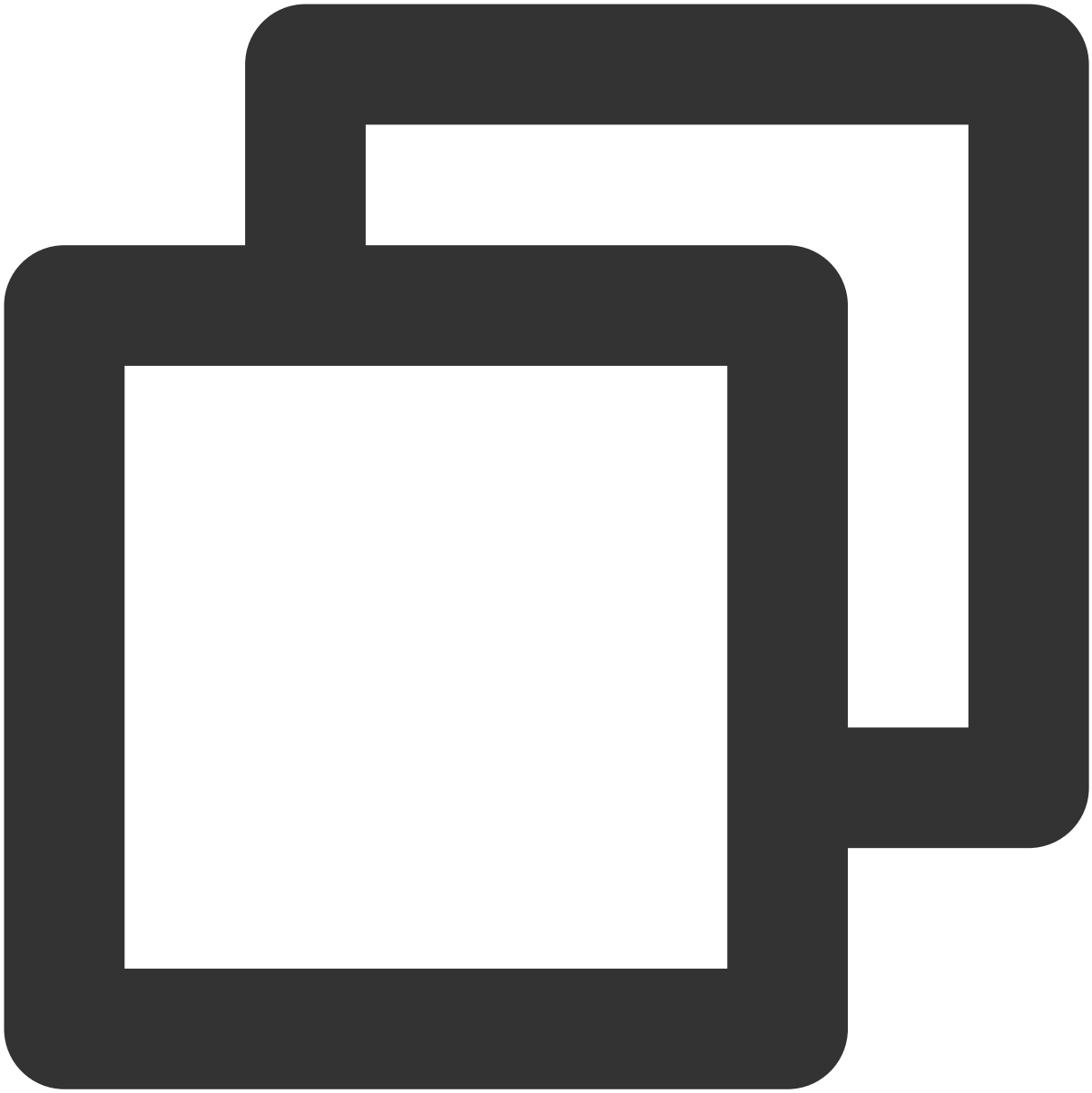
通过下载对象方法实现，用户只需要添加请求参数，请求参数格式如下，关于下载对象详细说明请参见 下载对象。



```
imageMogr2/format/<Format>
```

参数	含义
format	压缩格式，目标缩略图的图片格式为 TPG 或 HEIF。

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://www.tencentcloud.com/document/produ
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_image_compress()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *resp_headers;
    cos_table_t *params = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    params = cos_table_make(p, 1);
    apr_table_addn(params, "imageMogr2/format/tpg", "");
```

```
    cos_str_set(&object, "test.jpg");
    cos_str_set(&file, "test.tpg");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, params, &file, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("cos_get_object_to_file fail, req_id:%s\\n", s->req_id);
    }

    params = cos_table_make(p, 1);
    apr_table_addn(params, "imageMogr2/format/heif", "");
    cos_str_set(&file, "test.heif");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, params, &file, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("cos_get_object_to_file fail, req_id:%s\\n", s->req_id);
    }

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_image_compress();

    cos_http_io_deinitialize();

    return 0;
}
```

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 15:17:16

简介

本文档提供关于媒体 bucket 的 API 概览和 SDK 示例代码。

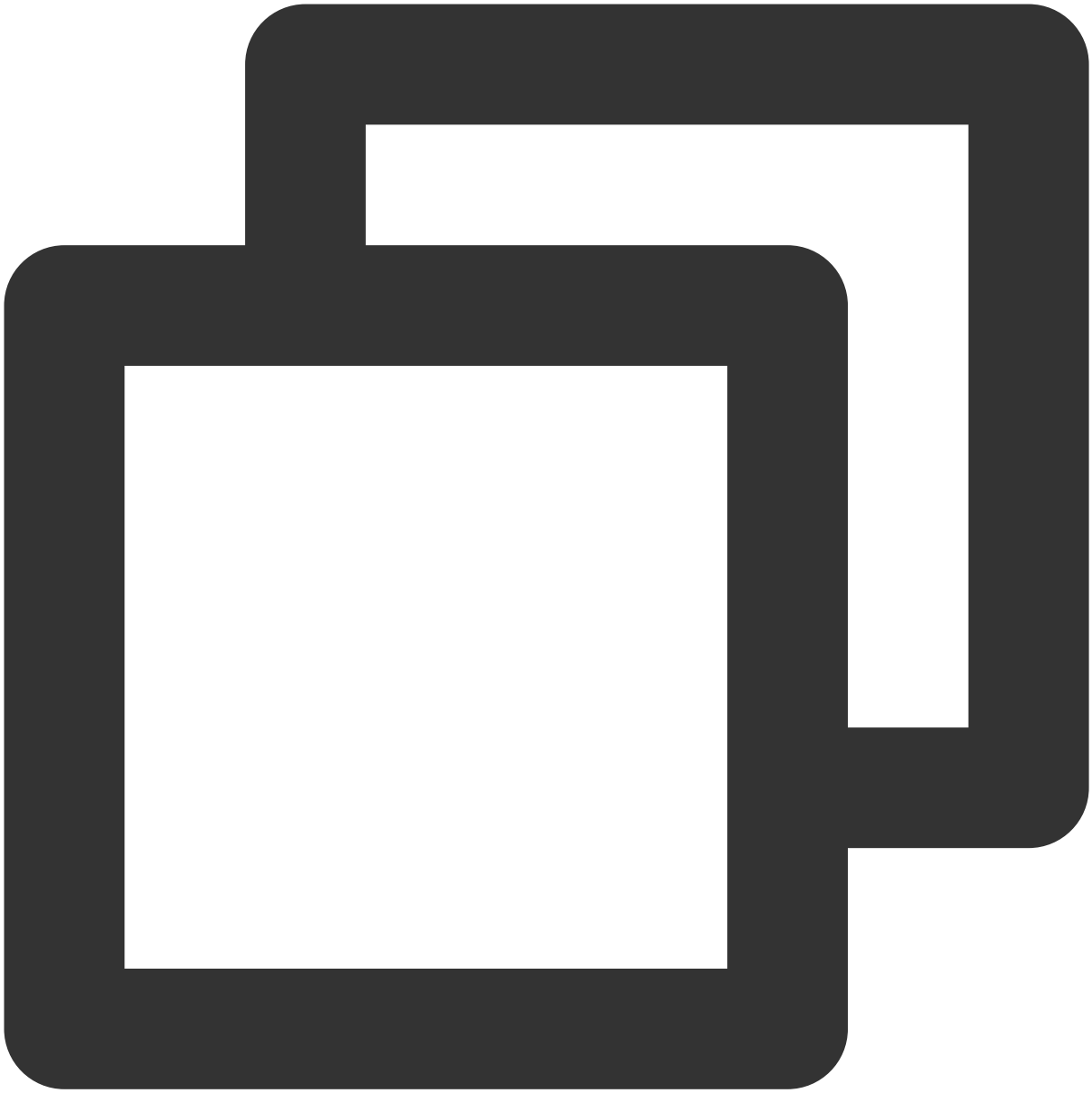
API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

功能说明

用于查询已经开通媒体处理功能的存储桶。

方法原型



```
cos_status_t *ci_describe_media_buckets(const cos_request_options_t *options,
                                        const ci_media_buckets_request_t *media_req
                                        cos_table_t *headers,
                                        cos_table_t **resp_headers,
                                        ci_media_buckets_result_t **media_result);
```

参数说明

参数名称	参数描述	类型

options	COS 请求选项	Struct
media_request	查询媒体桶的请求参数	Struct
regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域	String
bucket_names	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	String
bucket_name	存储桶名称前缀，前缀搜索	String
page_number	第几页	String
page_size	每页个数	String
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
media_result	返回 开通媒体处理功能的桶信息	Struct
total_count	媒体 Bucket 总数	Int
page_number	当前页数，同请求中的 pageNumber	Int
page_size	每页个数，同请求中的 pageSize	Int
media_bucket_list	媒体 Bucket 列表	Struct
bucket_id	存储桶 ID	String
name	存储桶名称，同 BucketId	String
region	所在的地域	String
create_time	创建时间	String

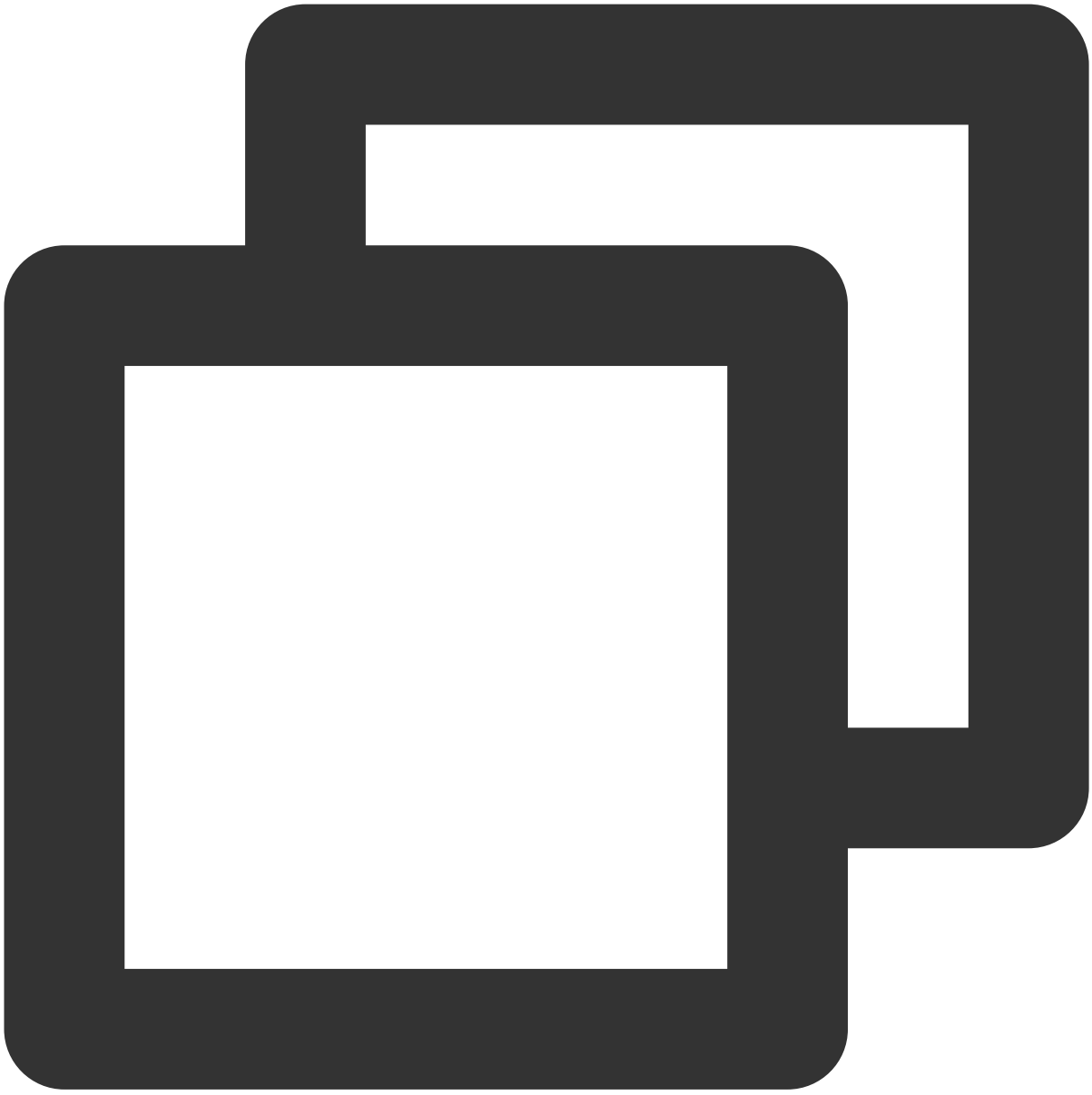
返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

版权所有：腾讯云计算（北京）有限责任公司

示例

完整代码请参见cos_demo.c中test_ci_media_process_media_bucket()函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// 数据万象的访问域名，详情请参见 https://www.tencentcloud.com/document/product/1045/334
static char TEST_CI_ENDPOINT[] = "https://ci.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
```

```
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

static void log_media_buckets_result(ci_media_buckets_result_t *result)
{
    int index = 0;
    ci_media_bucket_list_t *media_bucket;

    cos_warn_log("total_count: %d", result->total_count);
    cos_warn_log("page_number: %d", result->page_number);
    cos_warn_log("page_size: %d", result->page_size);

    cos_list_for_each_entry(ci_media_bucket_list_t, media_bucket, &result->media_bu
        cos_warn_log("media_bucket index:%d", ++index);
        cos_warn_log("media_bucket->bucket_id: %s", media_bucket->bucket_id.data);
        cos_warn_log("media_bucket->name: %s", media_bucket->name.data);
        cos_warn_log("media_bucket->region: %s", media_bucket->region.data);
        cos_warn_log("media_bucket->create_time: %s", media_bucket->create_time.dat
    }
}

void test_ci_media_process_media_bucket()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_table_t *resp_headers;
    ci_media_buckets_request_t *media_buckets_request;
    ci_media_buckets_result_t *media_buckets_result;

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_CI_ENDPOINT);     // https://ci.<R
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
```

```
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    media_buckets_request = ci_media_buckets_request_create(p);
    cos_str_set(&media_buckets_request->regions, "");
    cos_str_set(&media_buckets_request->bucket_names, "");
    cos_str_set(&media_buckets_request->bucket_name, "");
    cos_str_set(&media_buckets_request->page_number, "1");
    cos_str_set(&media_buckets_request->page_size, "10");
    s = ci_describe_media_buckets(options, media_buckets_request, NULL, &resp_heade
    log_status(s);
    if (s->code == 200) {
        log_media_buckets_result(media_buckets_result);
    }

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_media_process_media_bucket();

    cos_http_io_deinitialize();

    return 0;
}
```

媒体信息接口

最近更新时间：2024-03-13 15:17:16

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

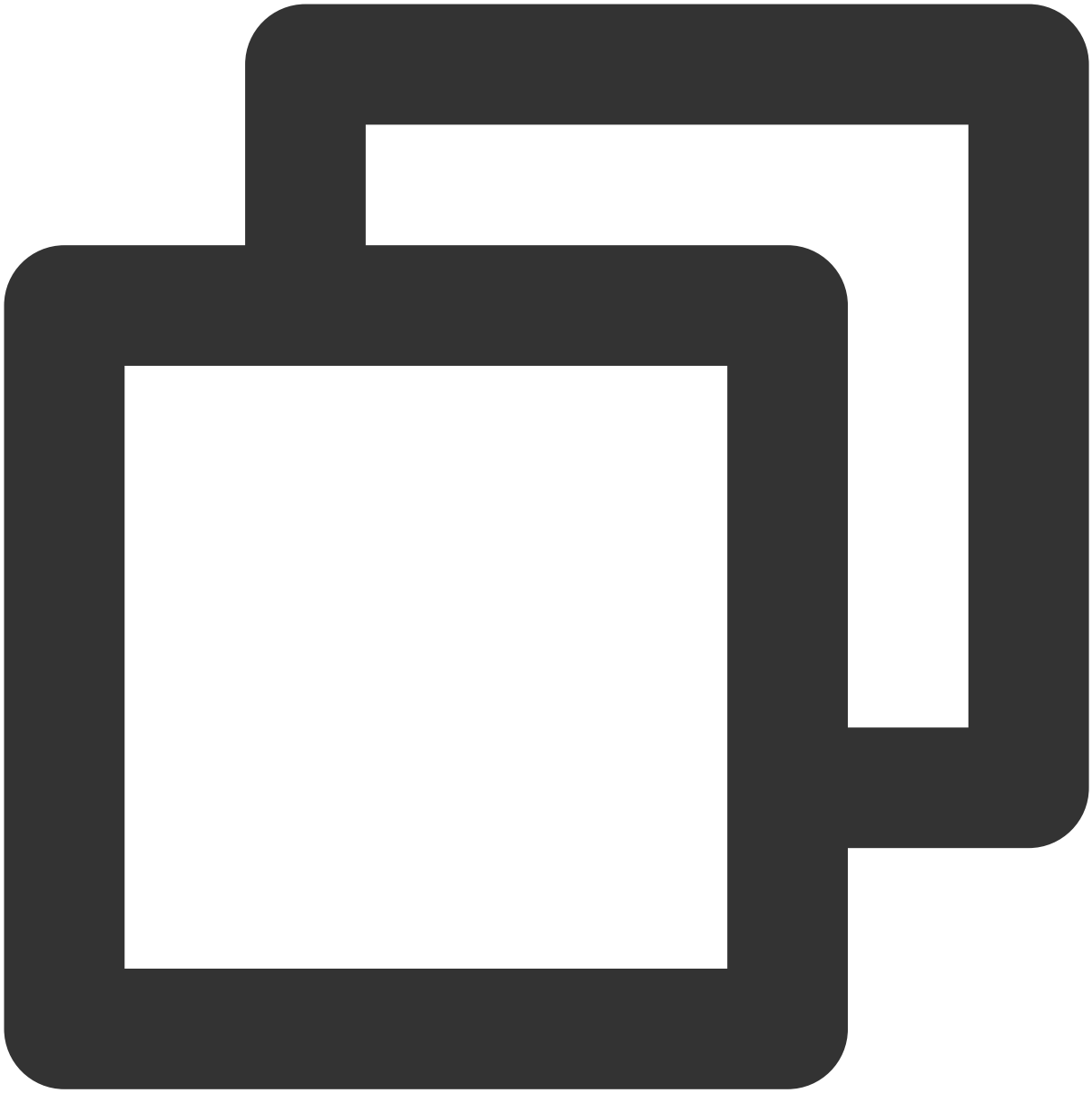
注意：
使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。详情请参见 开通媒体处理。

查询文件信息

功能说明

用于获取媒体文件的信息。

方法原型



```
cos_status_t *ci_get_media_info(const cos_request_options_t *options,
                                const cos_string_t *bucket,
                                const cos_string_t *object,
                                cos_table_t *headers,
                                cos_table_t **resp_headers,
                                ci_media_info_result_t **media_result);
```

参数说明

参数名称	参数描述	类型

options	COS 请求选项	Struct
bucket	存储桶名称，存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String
object	Object 名称	String
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
media_result	返回 媒体文件信息	Struct
format	格式信息	Struct
num_stream	Stream（包含 Video、Audio、Subtitle）的数量	Int
num_program	节目的数量	Int
format_name	容器格式名字	String
format_long_name	容器格式的详细名字	String
start_time	起始时间，单位为秒	Float
duration	时长，单位为秒	Float
bit_rate	比特率，单位为 kbps	Int
size	大小，单位为 Byte	Int
stream	流信息	Struct
video	视频信息	Struct
audio	音频信息	Struct
subtitle	字幕信息	Struct

video所对应的结构体说明

参数名称	参数描述	类型
index	该流的编号	Int
codec_name	编解码格式名字	String
codec_long_name	编解码格式的详细名字	String

版权所有：腾讯云计算（北京）有限责任公司

codec_time_base	编码时基	String
codec_tag_string	编码标签名	String
codec_tag	编码标签	String
profile	视频编码档位	String
height	视频高，单位 px	Int
width	视频宽，单位 px	Int
has_b_frame	是否有B帧。1表示有，0表示无	Int
ref_frames	视频编码的参考帧个数	Int
sar	采样宽高比	String
dar	显示宽高比	String
pix_format	像素格式	String
field_order	场的顺序	String
level	视频编码等级	Int
fps	视频帧率	Int
avg_fps	平均帧率	String
timebase	时基	String
start_time	视频开始时间，单位为秒	Float
duration	视频时长，单位为秒	Float
bit_rate	比特率，单位为 kbps	Float
num_frames	总帧数	Int
language	语言	String

audio所对应的结构体说明

参数名称	参数描述	类型
index	该流的编号	Int
codec_name	编解码格式名字	String

版权所有：腾讯云计算（北京）有限责任公司

codec_long_name	编解码格式的详细名字	String
codec_time_base	编码时基	String
codec_tag_string	编码标签名	String
codec_tag	编码标签	String
sample_fmt	采样格式	String
sample_rate	采样率	Int
channel	通道数量	Int
channel_layout	通道格式	String
timebase	时基	String
start_time	音频开始时间，单位为秒	Float
duration	音频时长，单位为秒	Float
bit_rate	比特率，单位为 kbps	Float
language	语言	String

subtitle所对应的结构体说明

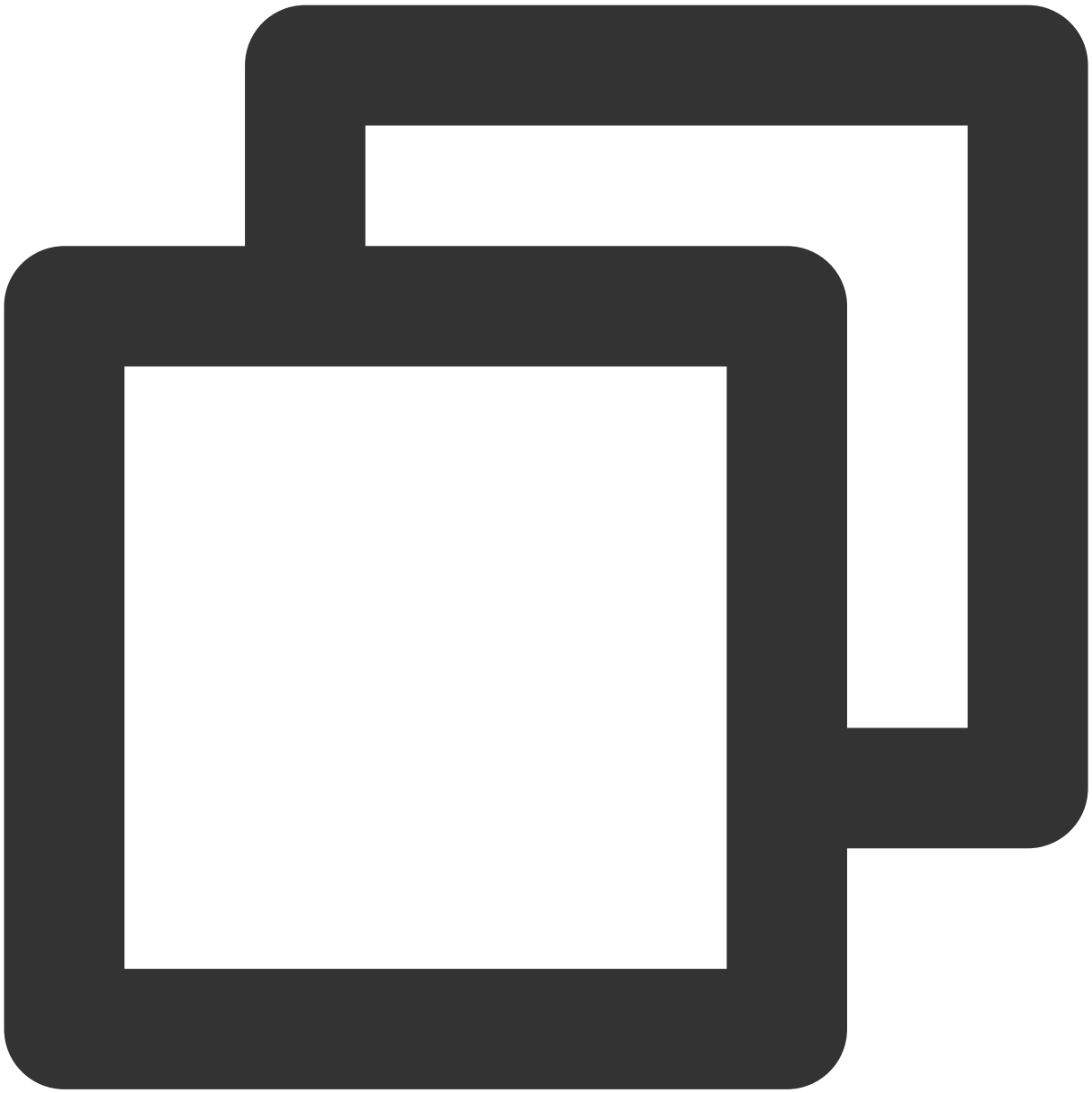
参数名称	参数描述	类型
index	该流的编号	Int
language	语言，und 表示无查询结果	String

返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

示例

完整代码请参见cos_demo.c中test_ci_media_process_media_info()函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <sys/stat.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://www.tencentcloud.com/document/produ
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
```

```
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

static void log_media_info_result(ci_media_info_result_t *result)
{
    // format
    cos_warn_log("format.num_stream: %d", result->format.num_stream);
    cos_warn_log("format.num_program: %d", result->format.num_program);
    cos_warn_log("format.format_name: %s", result->format.format_name.data);
    cos_warn_log("format.format_long_name: %s", result->format.format_long_name.dat
    cos_warn_log("format.start_time: %f", result->format.start_time);
    cos_warn_log("format.duration: %f", result->format.duration);
    cos_warn_log("format.bit_rate: %d", result->format.bit_rate);
    cos_warn_log("format.size: %d", result->format.size);

    // stream.video
    cos_warn_log("stream.video.index: %d", result->stream.video.index);
    cos_warn_log("stream.video.codec_name: %s", result->stream.video.codec_name.dat
    cos_warn_log("stream.video.codec_long_name: %s", result->stream.video.codec_lon
    cos_warn_log("stream.video.codec_time_base: %s", result->stream.video.codec_tim
    cos_warn_log("stream.video.codec_tag_string: %s", result->stream.video.codec_ta
    cos_warn_log("stream.video.codec_tag: %s", result->stream.video.codec_tag.data)
    cos_warn_log("stream.video.profile: %s", result->stream.video.profile.data);
    cos_warn_log("stream.video.height: %d", result->stream.video.height);
    cos_warn_log("stream.video.width: %d", result->stream.video.width);
    cos_warn_log("stream.video.has_b_frame: %d", result->stream.video.has_b_frame);
    cos_warn_log("stream.video.ref_frames: %d", result->stream.video.ref_frames);
    cos_warn_log("stream.video.sar: %s", result->stream.video.sar.data);
    cos_warn_log("stream.video.dar: %s", result->stream.video.dar.data);
    cos_warn_log("stream.video.pix_format: %s", result->stream.video.pix_format.dat
    cos_warn_log("stream.video.field_order: %s", result->stream.video.field_order.d
    cos_warn_log("stream.video.level: %d", result->stream.video.level);
    cos_warn_log("stream.video.fps: %d", result->stream.video.fps);
    cos_warn_log("stream.video.avg_fps: %s", result->stream.video.avg_fps.data);
    cos_warn_log("stream.video.timebase: %s", result->stream.video.timebase.data);
    cos_warn_log("stream.video.start_time: %f", result->stream.video.start_time);
```

版权所有：腾讯云计算（北京）有限责任公司

```
    cos_warn_log("stream.video.duration: %f", result->stream.video.duration);
    cos_warn_log("stream.video.bit_rate: %f", result->stream.video.bit_rate);
    cos_warn_log("stream.video.num_frames: %d", result->stream.video.num_frames);
    cos_warn_log("stream.video.language: %s", result->stream.video.language.data);

    // stream.audio
    cos_warn_log("stream.audio.index: %d", result->stream.audio.index);
    cos_warn_log("stream.audio.codec_name: %s", result->stream.audio.codec_name.dat
    cos_warn_log("stream.audio.codec_long_name: %s", result->stream.audio.codec_lon
    cos_warn_log("stream.audio.codec_time_base: %s", result->stream.audio.codec_tim
    cos_warn_log("stream.audio.codec_tag_string: %s", result->stream.audio.codec_ta
    cos_warn_log("stream.audio.codec_tag: %s", result->stream.audio.codec_tag.data)
    cos_warn_log("stream.audio.sample_fmt: %s", result->stream.audio.sample_fmt.dat
    cos_warn_log("stream.audio.sample_rate: %d", result->stream.audio.sample_rate);
    cos_warn_log("stream.audio.channel: %d", result->stream.audio.channel);
    cos_warn_log("stream.audio.channel_layout: %s", result->stream.audio.channel_la
    cos_warn_log("stream.audio.timebase: %s", result->stream.audio.timebase.data);
    cos_warn_log("stream.audio.start_time: %f", result->stream.audio.start_time);
    cos_warn_log("stream.audio.duration: %f", result->stream.audio.duration);
    cos_warn_log("stream.audio.bit_rate: %f", result->stream.audio.bit_rate);
    cos_warn_log("stream.audio.language: %s", result->stream.audio.language.data);

    // stream.subtitle
    cos_warn_log("stream.subtitle.index: %d", result->stream.subtitle.index);
    cos_warn_log("stream.subtitle.language: %s", result->stream.subtitle.language.d
}

void test_ci_media_process_media_info()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers;
    ci_media_info_result_t *media_info;
    cos_string_t object;

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
```

```
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, "test.mp4");

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    s = ci_get_media_info(options, &bucket, &object, NULL, &resp_headers, &media_in
    log_status(s);
    if (s->code == 200) {
        log_media_info_result(media_info);
    }

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_media_process_media_info();

    cos_http_io_deinitialize();

    return 0;
}
```

媒体截图接口

最近更新时间：2024-03-13 15:17:14

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

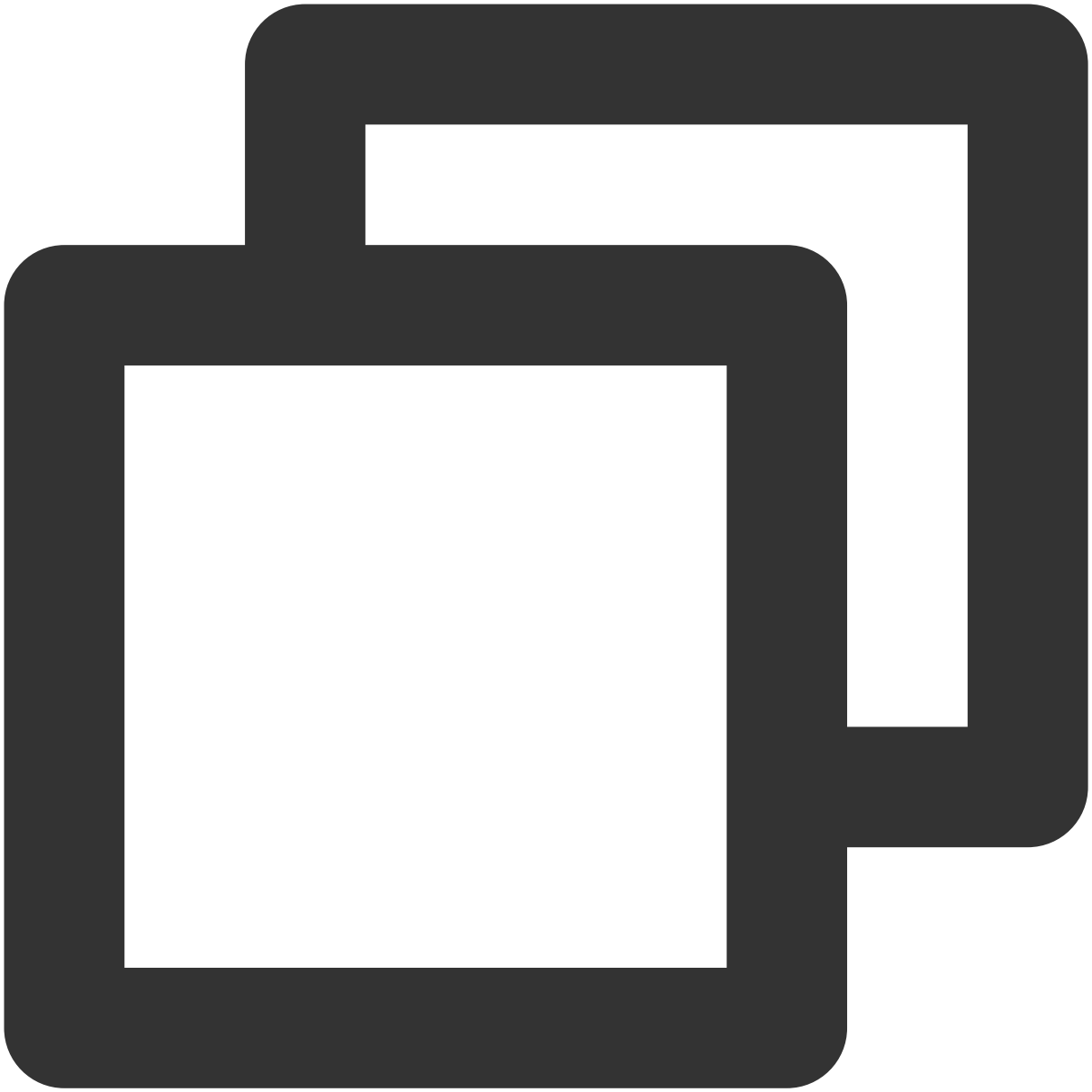
注意：
使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。详情请参见 开通媒体处理。

查询截图(内存)

功能说明

用于获取媒体文件某个时间的截图，截图信息放到内存buffer。

方法原型



```
cos_status_t *ci_get_snapshot_to_buffer(const cos_request_options_t *options,
                                        const cos_string_t *bucket,
                                        const cos_string_t *object,
                                        const ci_get_snapshot_request_t *snapshot_r
                                        cos_table_t *headers,
                                        cos_list_t *buffer,
                                        cos_table_t **resp_headers);
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String
object	Object 名称	String
snapshot_request	获取媒体截图的请求参数	Struct
time	截图的时间点，单位为秒	Float
width	截图的宽。默认为0	Int
height	截图的高。默认为0 width 和 height 都为0时表示使用视频的宽高，如果单个为0，则以另外一个值按视频宽高比例自动适应	Int
format	截图的格式，支持 jpg 和 png，默认 jpg	String
rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转 off：不旋转 默认值为 auto	String
mode	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 默认值为 exactframe	String
headers	COS 请求附加头域	Struct
buffer	返回 截图图片的内容	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct

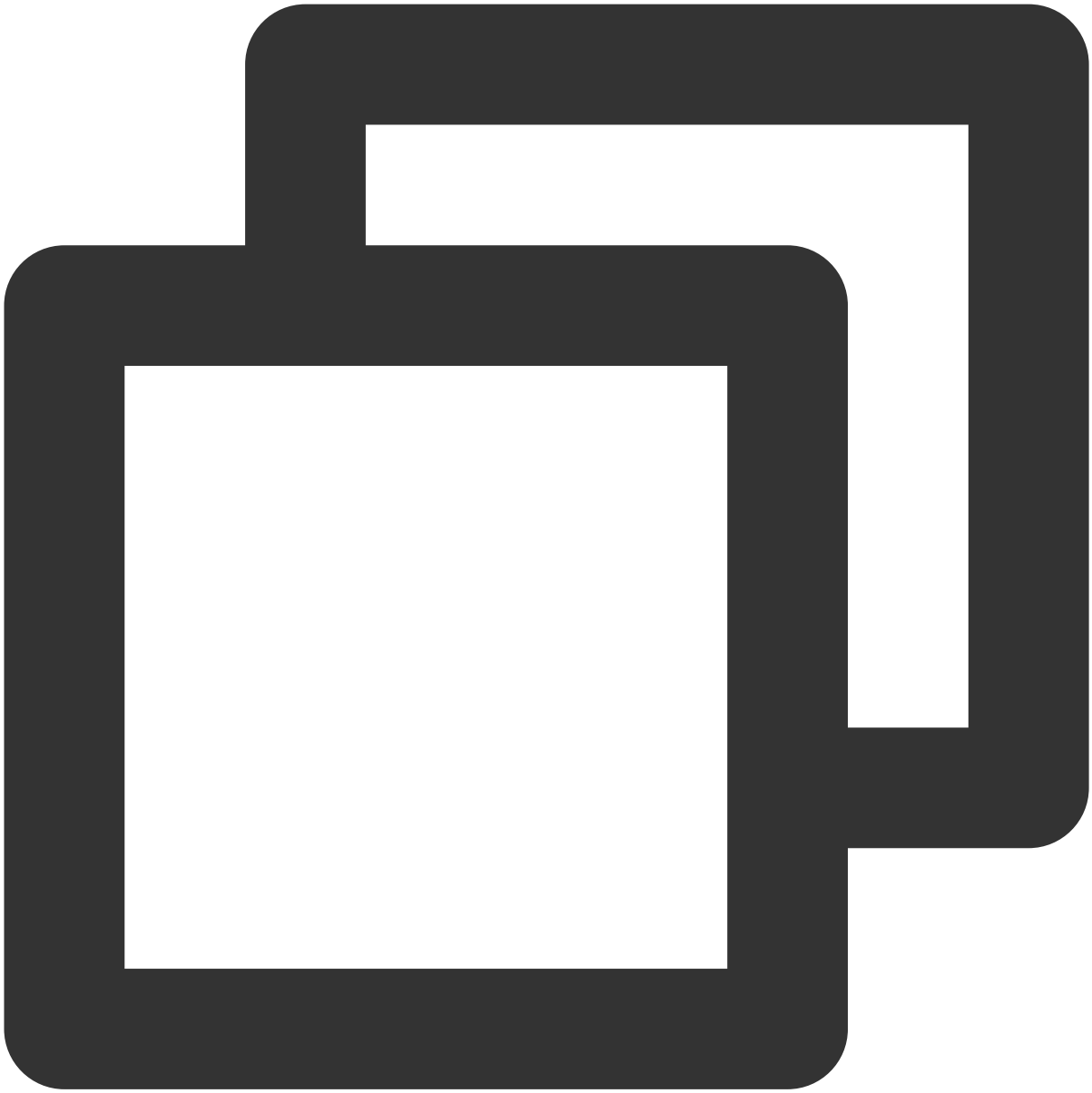
返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

版权所有：腾讯云计算（北京）有限责任公司

示例

完整代码请参见cos_demo.c中test_ci_media_process_snapshot()函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// endpoint 是 COS 访问域名信息，详情请参见 https://www.tencentcloud.com/document/produ
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
```

```
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void test_ci_media_process_snapshot()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers;
    cos_list_t download_buffer;
    cos_string_t object;
    ci_get_snapshot_request_t *snapshot_request;

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, "test.mp4");

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    snapshot_request = ci_snapshot_request_create(p);
    snapshot_request->time = 7.5;
    snapshot_request->width = 0;
    snapshot_request->height = 0;
    cos_str_set(&snapshot_request->format, "jpg");
    cos_str_set(&snapshot_request->rotate, "auto");
    cos_str_set(&snapshot_request->mode, "exactframe");
```

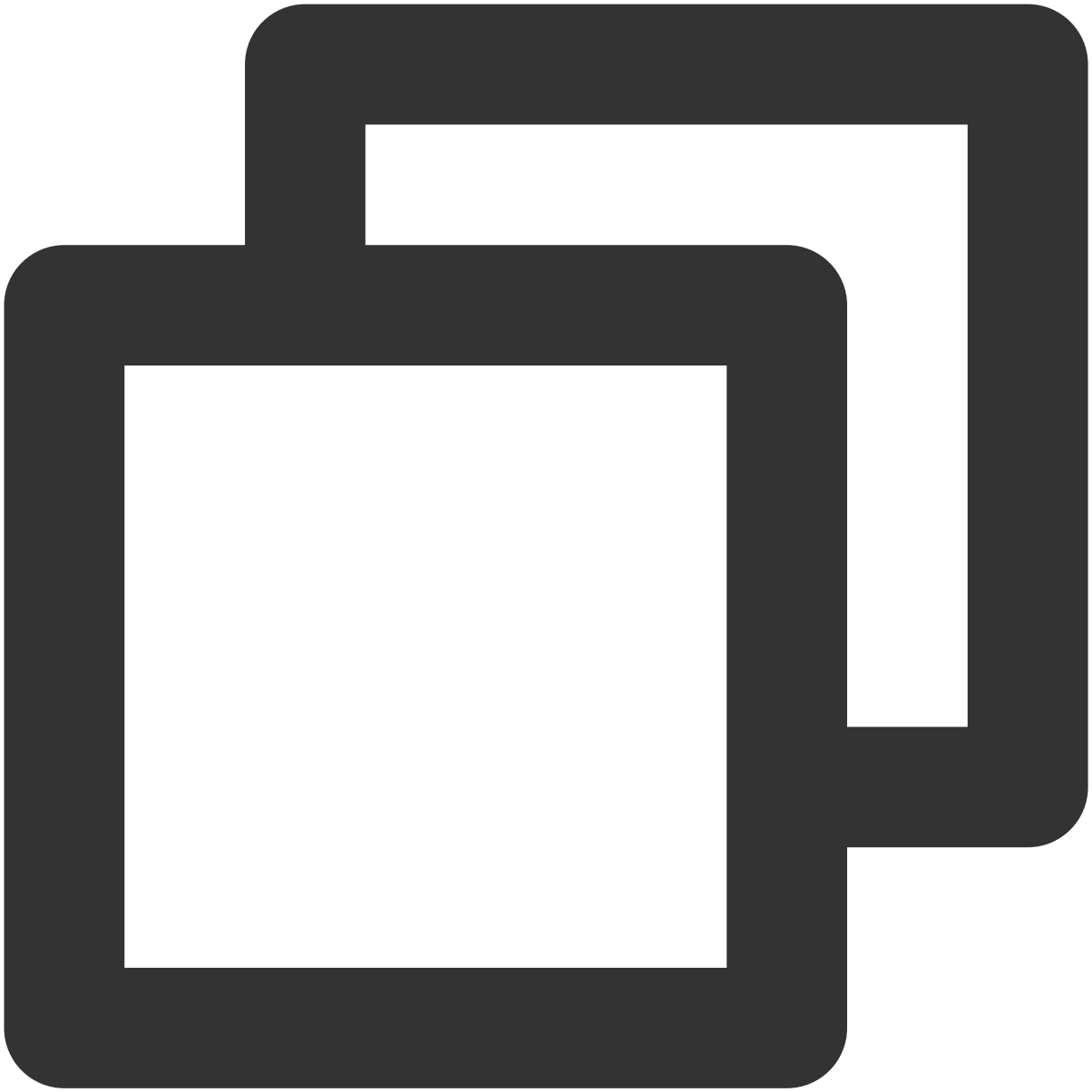
```
    cos_list_init(&download_buffer);

    s = ci_get_snapshot_to_buffer(options, &bucket, &object, snapshot_request, NULL
    log_status(s);

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_media_process_snapshot();

    cos_http_io_deinitialize();

    return 0;
}
```

查询截图(文件)

功能说明

用于获取媒体文件某个时间的截图，截图信息放到本地文件中。

方法原型



```
cos_status_t *ci_get_snapshot_to_file(const cos_request_options_t *options,
                                      const cos_string_t *bucket,
                                      const cos_string_t *object,
                                      const ci_get_snapshot_request_t *snapshot_req
                                      cos_table_t *headers,
                                      cos_string_t *filename,
                                      cos_table_t **resp_headers);
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String
object	Object 名称	String
snapshot_request	获取媒体截图的请求参数	Struct
time	截图的时间点，单位为秒	Float
width	截图的宽。默认为0	Int
height	截图的高。默认为0 width 和 height 都为0时表示使用视频的宽高，如果单个为0，则以另外一个值按视频宽高比例自动适应	Int
format	截图的格式，支持 jpg 和 png，默认 jpg	String
rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转 off：不旋转 默认值为 auto	String
mode	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 默认值为 exactframe	String
headers	COS 请求附加头域	Struct
filename	存放截图信息的文件名	String
resp_headers	返回 HTTP 响应消息的头域	Struct

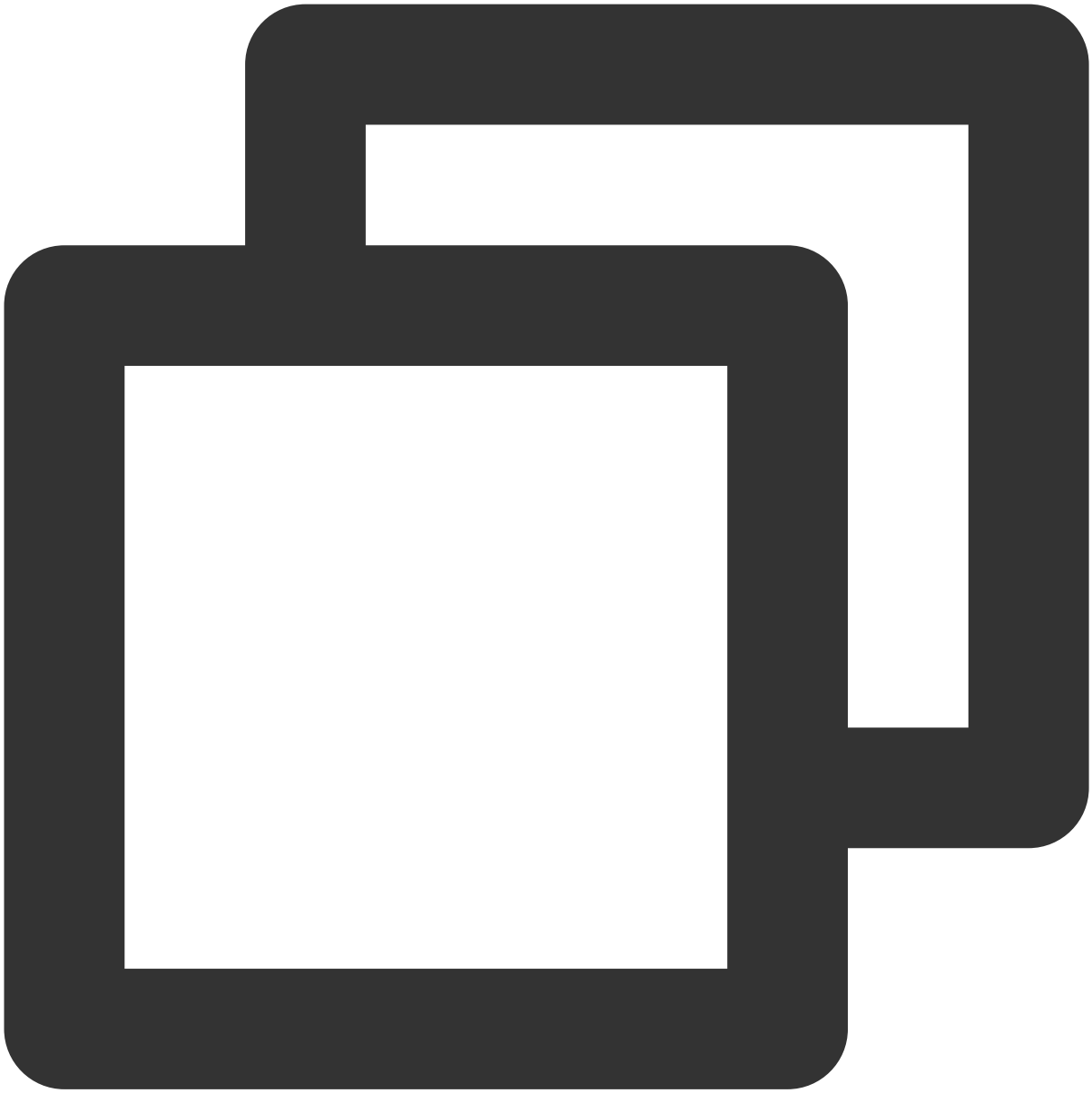
返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

版权所有：腾讯云计算（北京）有限责任公司

示例

完整代码请参见cos_demo.c中test_ci_media_process_snapshot()函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <sys/stat.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://www.tencentcloud.com/document/produ
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
```

```
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void test_ci_media_process_snapshot()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers;
    cos_string_t object;
    ci_get_snapshot_request_t *snapshot_request;
    cos_string_t pic_file = cos_string("snapshot.jpg");

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, "test.mp4");

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    snapshot_request = ci_snapshot_request_create(p);
    snapshot_request->time = 7.5;
    snapshot_request->width = 0;
    snapshot_request->height = 0;
    cos_str_set(&snapshot_request->format, "jpg");
    cos_str_set(&snapshot_request->rotate, "auto");
```

```
    cos_str_set(&snapshot_request->mode, "exactframe");

    s = ci_get_snapshot_to_file(options, &bucket, &object, snapshot_request, NULL,
    log_status(s);

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_media_process_snapshot();

    cos_http_io_deinitialize();

    return 0;
}
```

内容审核
视频审核

最近更新时间：2024-03-13 15:17:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核的 API 概览和 SDK 示例代码。

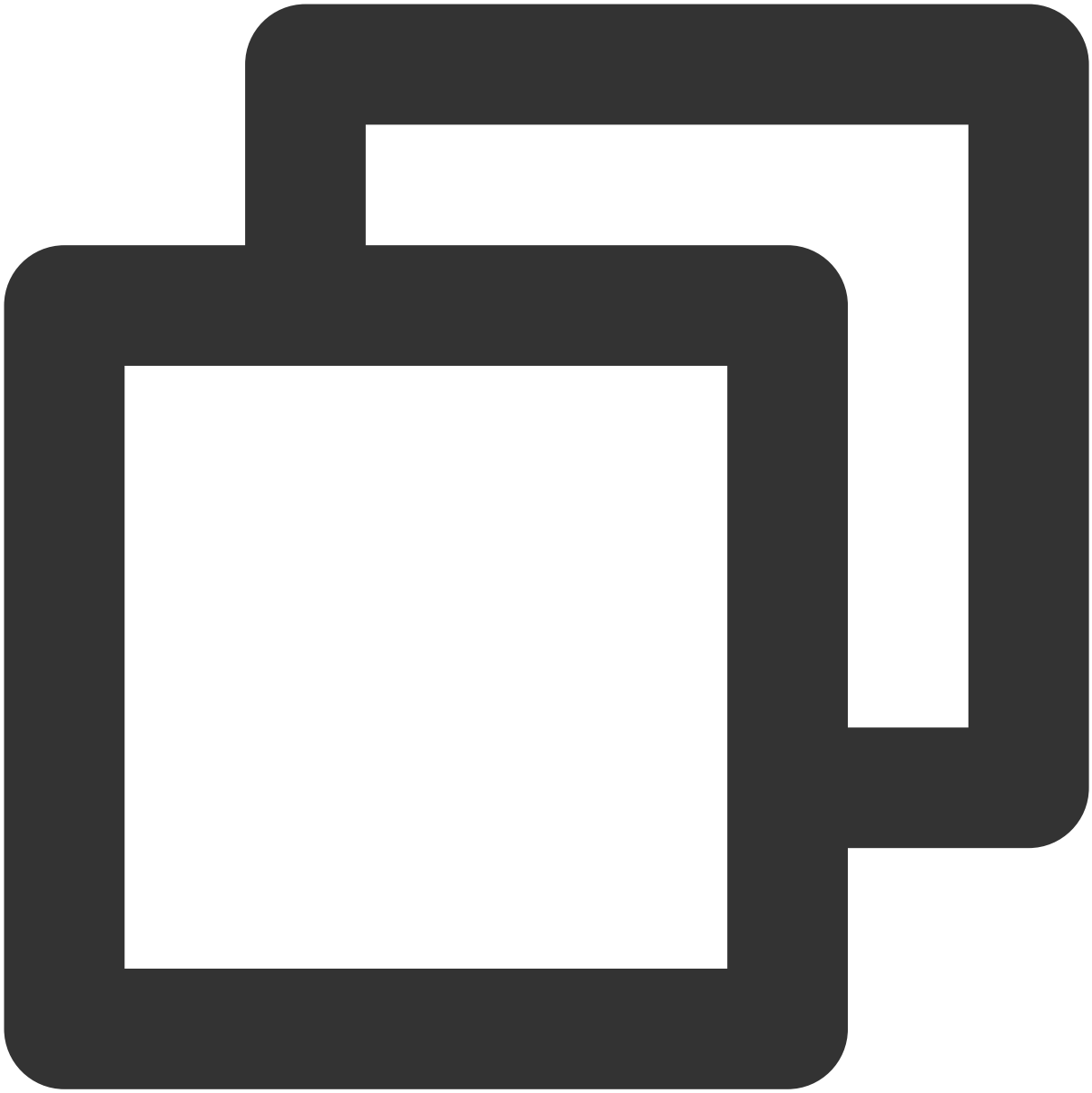
API	操作描述
提交视频审核任务	用于提交一个视频审核任务
查询视频审核任务结果	用于查询指定的视频审核任务

提交视频审核任务

功能说明

用于提交一个视频审核任务。

方法原型



```
cos_status_t *ci_create_video_auditing_job(const cos_request_options_t *options,
                                           const cos_string_t *bucket,
                                           const ci_video_auditing_job_options_t *j
                                           cos_table_t *headers,
                                           cos_table_t **resp_headers,
                                           ci_video_auditing_job_result_t **job_res
```

参数说明

参数名称	参数描述	类型

options	COS 请求选项	Struct
bucket	存储桶名称，Bucket 的命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String
job_options	提交视频审核任务的参数配置	Struct
input_object	对象名	String
detect_type	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads	String
callback	回调地址，以http://或者https://开头的地址	String
callback_version	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple	String
biz_type	审核策略，不带审核策略时使用默认策略	String
detect_content	用于指定是否审核视频声音，当值为0时：表示只审核视频画面截图；值为1时：表示同时审核视频画面截图和视频声音。默认值为0	Integer
snapshot	视频画面的审核通过视频截帧能力截取出一定量的截图，通过对截图逐一审核而实现的，该参数用于指定视频截帧的配置	Struct
mode	截帧模式。Interval 表示间隔模式，Average 表示平均模式，Fps 表示固定帧率模式。 Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片 Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片 Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧	String
count	视频截帧数量，范围为(0, 10000]	Integer
time_interval	视频截帧频率，范围为(0, 60]，单位为秒，支持 float 格式，执行精度精确到毫秒	Float
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
job_result	返回提交任务的响应信息	Struct
job_id	本次视频审核任务的 ID	String
state	视频审核任务的状态，值为 Submitted（已提交审核）、Snapshoting（视频截帧中）、Success（审核成功）、Failed（审核失败）、Auditing（审核	String

版权所有：腾讯云计算（北京）有限责任公司

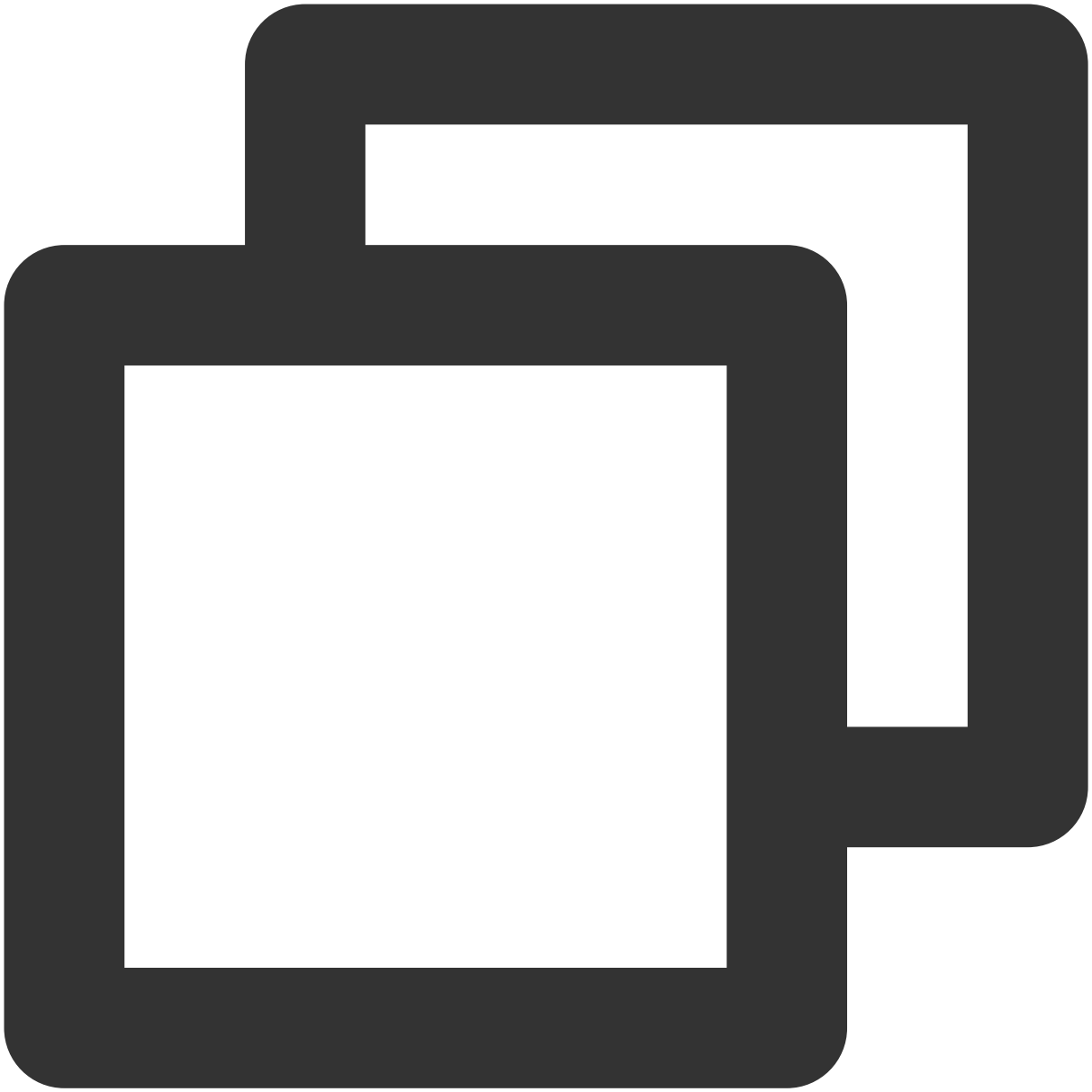
	中）其中一个	
creation_time	视频审核任务的创建时间	String

返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

示例

完整代码请参见 cos_demo.c 中的 test_ci_video_auditing() 函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <sys/stat.h>

// 数据万象的访问域名，详情请参见 https://www.tencentcloud.com/document/product/1045/334
static char TEST_CI_ENDPOINT[] = "https://ci.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

static void log_video_auditing_result(ci_video_auditing_job_result_t *result)
{
    cos_warn_log("jobid: %s", result->jobs_detail.job_id.data);
    cos_warn_log("state: %s", result->jobs_detail.state.data);
    cos_warn_log("creation_time: %s", result->jobs_detail.creation_time.data);
}

void test_ci_create_video_auditing()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers;
    ci_video_auditing_job_options_t *job_options;
    ci_video_auditing_job_result_t *job_result;

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_CI_ENDPOINT);     // https://ci.<R
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    job_options = ci_video_auditing_job_options_create(p);
    cos_str_set(&job_options->input_object, "test.mp4");
    cos_str_set(&job_options->job_conf.detect_type, "Porn,Ads");
    cos_str_set(&job_options->job_conf.callback_version, "Detail");
```

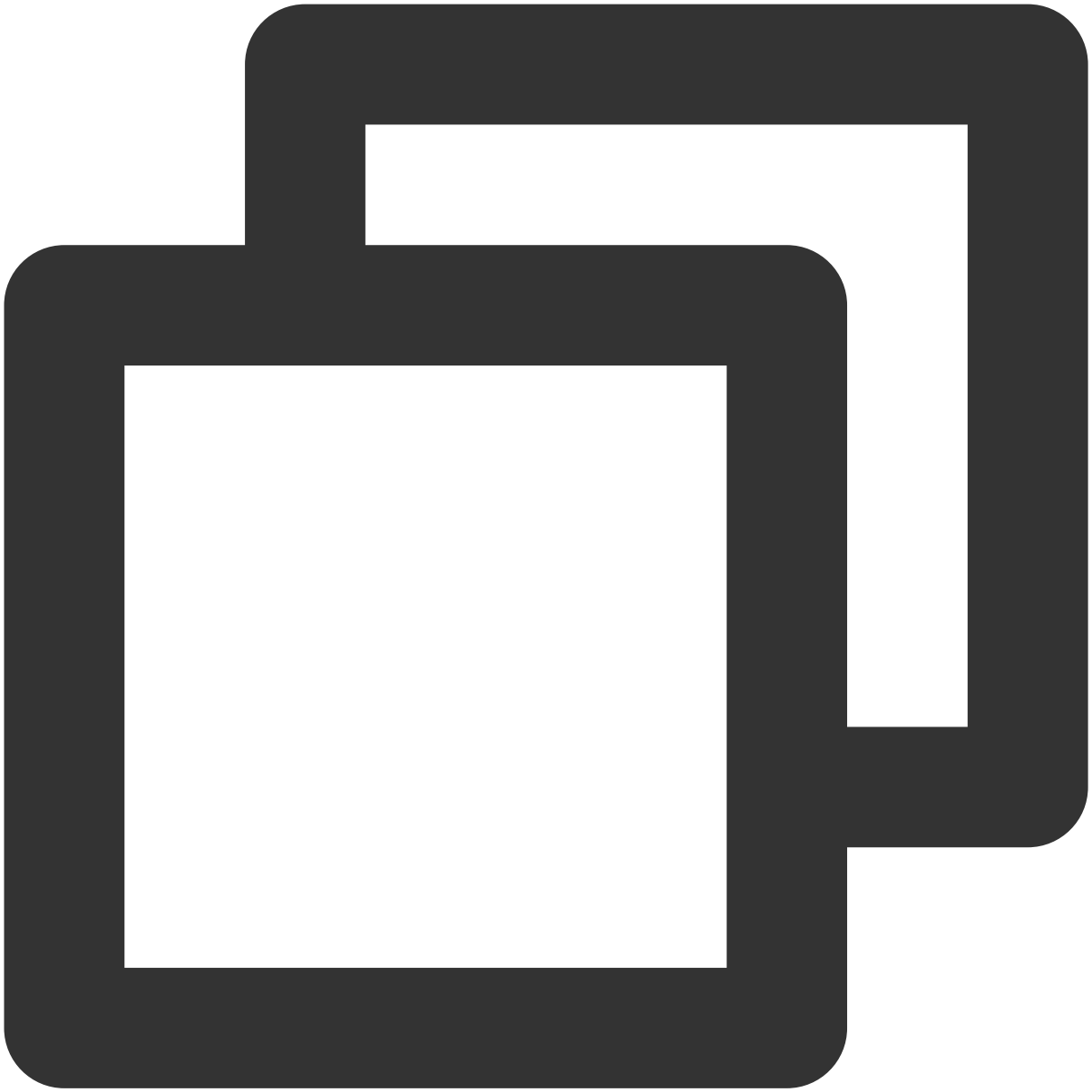
```
    job_options->job_conf.detect_content = 1;
    cos_str_set(&job_options->job_conf.snapshot.mode, "Interval");
    job_options->job_conf.snapshot.time_interval = 1.5;
    job_options->job_conf.snapshot.count = 10;

    // 提交一个视频审核任务
    s = ci_create_video_auditing_job(options, &bucket, job_options, NULL, &resp_hea
    log_status(s);
    if (s->code == 200) {
        log_video_auditing_result(job_result);
    }

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_create_video_auditing();

    cos_http_io_deinitialize();

    return 0;
}
```

查询视频审核任务

功能说明

用于查询视频审核任务执行情况和结果。

方法原型



```
cos_status_t *ci_get_auditing_job(const cos_request_options_t *options,
                                  const cos_string_t *bucket,
                                  const cos_string_t *job_id,
                                  cos_table_t *headers,
                                  cos_table_t **resp_headers,
                                  ci_auditing_job_result_t **job_result);
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，Bucket 的命名规则为 BucketName-APPID ，此处填写的存储桶名称必须为此格式	String
job_id	本次视频审核任务的 ID	String
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
job_result	返回查询任务的响应信息	Struct
nonexist_job_ids	查询的 ID 中不存在任务，所有任务都存在时不返回	String
jobs_detail	视频审核任务的详细信息	Struct
code	错误码，只有 State为 Failed 时有意义	String
message	错误描述，只有 State 为 Failed 时有意义	String
job_id	视频审核任务的 ID	String
state	视频审核任务的状态，值为 Submitted（已提交审核）、Snapshoting（视频截帧中）、Success（审核成功）、Failed（审核失败）、Auditing（审核中）其中一个	String
creation_time	视频审核任务的创建时间	String
object	被审核的视频文件的名称	String
snapshot_count	视频截图的总数量	String
result	该字段表示本次判定的审核结果，您可以根据该结果，进行后续的操作；建议您按照业务所需，对不同的审核结果进行相应处理。有效值：0（审核正常），1 （判定为违规敏感文件），2（疑似敏感，建议人工复核）	Integer
porn_info	审核场景为涉黄的审核结果信息	Struct
ads_info	审核场景为广告引导的审核结果信息	Struct
hit_flag	是否命中该审核分类，0表示未命中，1表示命中，2表示疑似	Integer
count	命中该审核分类的截图张数	Integer
snapshot_info_list	该字段用于返回视频中视频画面截图审核的结果。注意：每次查看数	Struct

版权所有：腾讯云计算（北京）有限责任公司

	据的有效期为2小时，2小时后如需查看，请重新发起查询请求	
audio_section_info_list	字段用于返回视频中视频声音审核的结果。注意：每次查看数据的有效期为2小时，2小时后如需查看，请重新发起查询请求	Struct

snapshot_info_list 所对应的结构体说明

参数名称	参数描述	类型
url	视频截图的访问地址，您可以通过该地址查看该截图内容，地址格式为标准 URL 格式	String
snapshot_time	该字段用于返回当前截图位于视频中的时间，单位为毫秒。例如5000（视频开始后5000毫秒）	Integer
text	本该字段用于返回当前截图的图片 OCR 文本识别的检测结果（仅在审核策略开启文本内容检测时返回），识别上限为5000字节	String
porn_info	审核场景为涉黄的审核结果信息	Struct
ads_info	审核场景为广告引导的审核结果信息	Struct
hit_flag	是否命中该审核分类，0表示未命中，1表示命中，2表示疑似	Integer
score	该字段表示审核结果命中审核信息的置信度，取值范围：0（置信度最低）-100（置信度最高），越高代表该内容越有可能属于当前返回审核信息。例如：色情 99，则表明该内容非常有可能属于色情内容	Integer
label	该字段为兼容旧版本的保留字段，表示该截图的结果标签（可能为 SubLabel，可能为人物名字等）	String
sub_lable	该字段表示审核命中的具体子标签，例如：Porn 下的 SexBehavior 子标签。注意：该字段可能返回空，表示未命中具体的子标签	String

audio_section_info_list 所对应的结构体说明

参数名称	参数描述	类型
url	视频声音片段的访问地址，您可以通过该地址获取该声音片段的内容，地址格式为标准 URL 格式	String
text	该字段用于返回当前视频声音的 ASR 文本识别的检测结果（仅在审核策略开启文本内容检测时返回），识别上限为5小时	String
offset_time	视该字段用于返回当前声音片段位于视频中的时间，单位为毫秒，例如5000（视频开始后5000毫秒）	Integer

版权所有：腾讯云计算（北京）有限责任公司

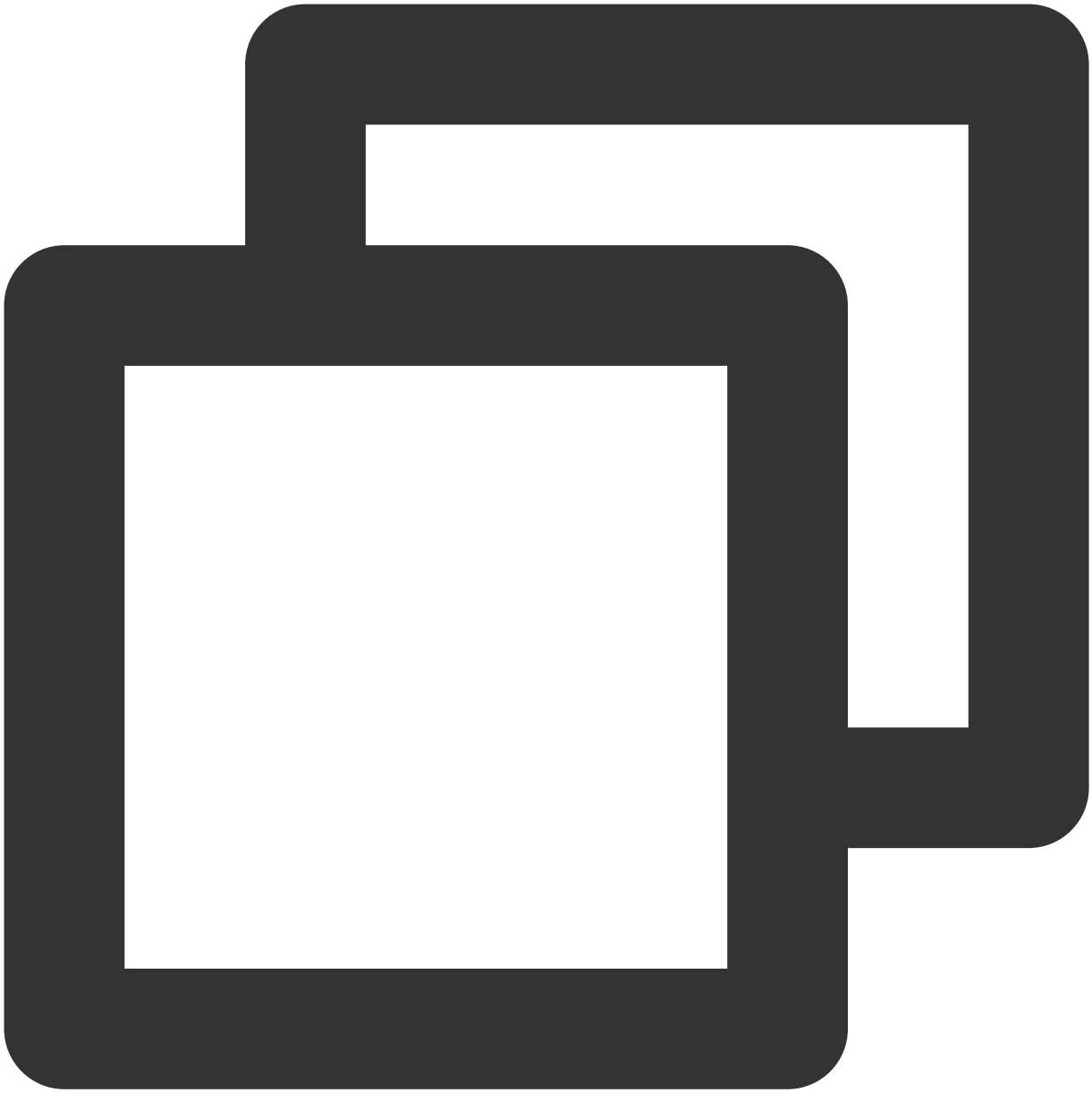
duration	当前视频声音片段的时长，单位毫秒	Integer
porn_info	审核场景为涉黄的审核结果信息	Struct
ads_info	审核场景为广告引导的审核结果信息	Struct
hit_flag	是否命中该审核分类，0表示未命中，1表示命中，2表示疑似	Integer
score	该字段表示审核结果命中审核信息的置信度，取值范围：0（置信度最低）-100（置信度最高），越高代表该内容越有可能属于当前返回审核信息。例如：色情 99，则表明该内容非常有可能属于色情内容	Integer
key_words	在当前审核场景下命中的关键词	String

返回结果说明

返回结果	描述	类型
code	错误码	Int
error_code	错误码内容	String
error_msg	错误码描述	String
req_id	请求消息 ID	String

示例

完整代码请参见 cos_demo.c 中的 test_ci_video_auditing() 函数。



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// 数据万象的访问域名，详情请参见 https://www.tencentcloud.com/document/product/1045/334
static char TEST_CI_ENDPOINT[] = "https://ci.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
```

```
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

static void log_get_auditing_result(ci_auditing_job_result_t *result)
{
    int index = 0;
    ci_auditing_snapshot_result_t *snapshot_info;
    ci_auditing_audio_section_result_t *audio_section_info;

    cos_warn_log("nonexist_job_ids: %s", result->nonexist_job_ids.data);
    cos_warn_log("code: %s", result->jobs_detail.code.data);
    cos_warn_log("message: %s", result->jobs_detail.message.data);
    cos_warn_log("state: %s", result->jobs_detail.state.data);
    cos_warn_log("creation_time: %s", result->jobs_detail.creation_time.data);
    cos_warn_log("object: %s", result->jobs_detail.object.data);
    cos_warn_log("snapshot_count: %s", result->jobs_detail.snapshot_count.data);
    cos_warn_log("result: %d", result->jobs_detail.result);

    cos_warn_log("porn_info.hit_flag: %d", result->jobs_detail.porn_info.hit_flag);
    cos_warn_log("porn_info.count: %d", result->jobs_detail.porn_info.count);
    cos_warn_log("ads_info.hit_flag: %d", result->jobs_detail.ads_info.hit_flag);
    cos_warn_log("ads_info.count: %d", result->jobs_detail.ads_info.count);

    cos_list_for_each_entry(ci_auditing_snapshot_result_t, snapshot_info, &result->
        cos_warn_log("snapshot index:%d", ++index);
        cos_warn_log("snapshot_info->url: %s", snapshot_info->url.data);
        cos_warn_log("snapshot_info->snapshot_time: %d", snapshot_info->snapshot_ti
        cos_warn_log("snapshot_info->text: %s", snapshot_info->text.data);

        cos_warn_log("snapshot_info->porn_info.hit_flag: %d", snapshot_info->porn_i
        cos_warn_log("snapshot_info->porn_info.score: %d", snapshot_info->porn_info
        cos_warn_log("snapshot_info->porn_info.label: %s", snapshot_info->porn_info
        cos_warn_log("snapshot_info->porn_info.sub_lable: %s", snapshot_info->porn_
        cos_warn_log("snapshot_info->ads_info.hit_flag: %d", snapshot_info->ads_inf
        cos_warn_log("snapshot_info->ads_info.score: %d", snapshot_info->ads_info.s
        cos_warn_log("snapshot_info->ads_info.label: %s", snapshot_info->ads_info.l
        cos_warn_log("snapshot_info->ads_info.sub_lable: %s", snapshot_info->ads_in
    }
```

```
    index = 0;
    cos_list_for_each_entry(ci_auditing_audio_section_result_t, audio_section_info,
        cos_warn_log("audio_section index:%d", ++index);
        cos_warn_log("audio_section_info->url: %s", audio_section_info->url.data);
        cos_warn_log("audio_section_info->text: %s", audio_section_info->text.data)
        cos_warn_log("audio_section_info->offset_time: %d", audio_section_info->off
        cos_warn_log("audio_section_info->duration: %d", audio_section_info->durati

        cos_warn_log("audio_section_info->porn_info.hit_flag: %d", audio_section_in
        cos_warn_log("audio_section_info->porn_info.score: %d", audio_section_info-
        cos_warn_log("audio_section_info->porn_info.key_words: %s", audio_section_i
        cos_warn_log("audio_section_info->ads_info.hit_flag: %d", audio_section_inf
        cos_warn_log("audio_section_info->ads_info.score: %d", audio_section_info->
        cos_warn_log("audio_section_info->ads_info.key_words: %s", audio_section_in
    }
}

void test_ci_get_video_auditing()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers;
    ci_video_auditing_job_options_t *job_options;
    ci_video_auditing_job_result_t *job_result;
    ci_auditing_job_result_t *auditing_result;

    // 基本配置
    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, TEST_CI_ENDPOINT);     // https://ci.<R
    cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&options->config->appid, TEST_APPID);
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // 替换为您的配置信息，可参见文档 https://www.tencentcloud.com/document/product/436/
    job_options = ci_video_auditing_job_options_create(p);
    cos_str_set(&job_options->input_object, "test.mp4");
    cos_str_set(&job_options->job_conf.detect_type, "Porn,Ads");
    cos_str_set(&job_options->job_conf.callback_version, "Detail");
```

```
    job_options->job_conf.detect_content = 1;
    cos_str_set(&job_options->job_conf.snapshot.mode, "Interval");
    job_options->job_conf.snapshot.time_interval = 1.5;
    job_options->job_conf.snapshot.count = 10;

    // 提交一个视频审核任务
    s = ci_create_video_auditing_job(options, &bucket, job_options, NULL, &resp_hea
    log_status(s);

    // 等待视频审核任务完成，此处可修改您的等待时间
    sleep(300);

    // 获取审核任务结果
    s = ci_get_auditing_job(options, &bucket, &job_result->jobs_detail.job_id, NULL
    log_status(s);
    if (s->code == 200) {
        log_get_auditing_result(auditing_result);
    }

    // 销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_get_video_auditing();

    cos_http_io_deinitialize();

    return 0;
}
```

内容识别
二维码识别

最近更新时间：2024-03-13 15:17:11

简介

本文档提供关于图片二维码相关的 API 概览以及 SDK 示例代码。

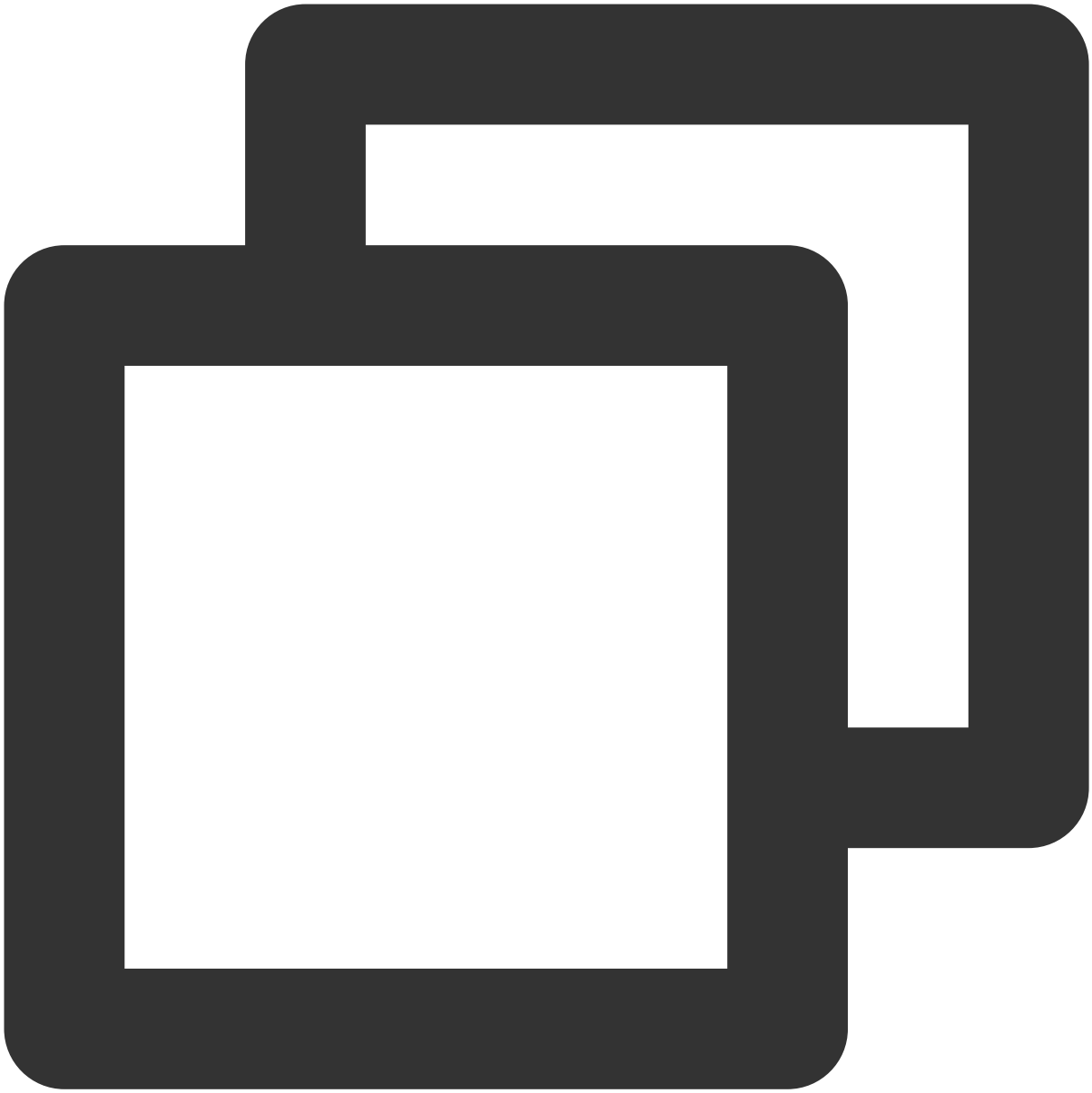
API	操作描述
二维码识别	二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克

二维码识别

二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克。

上传时识别

方法原型

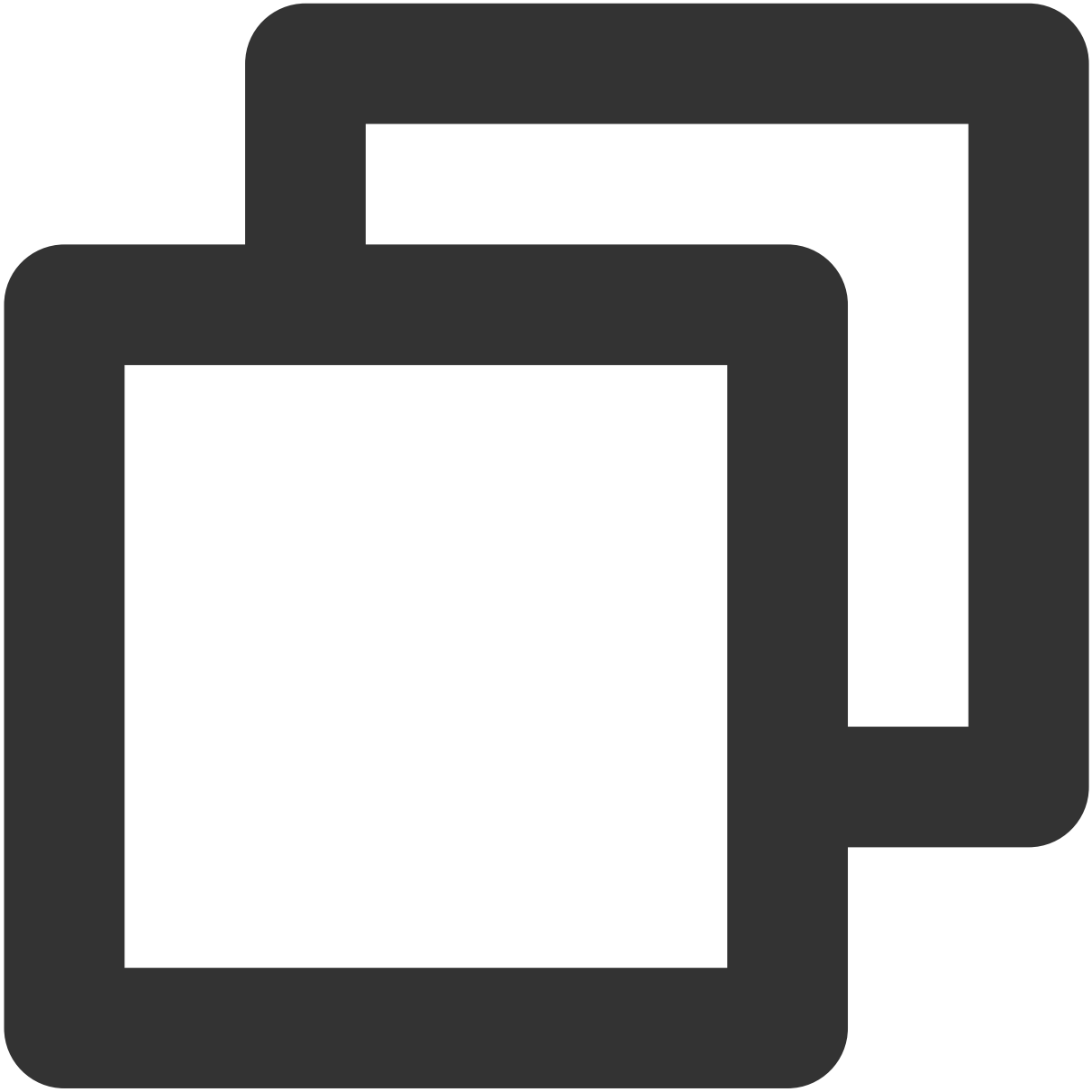


```
cos_status_t *ci_put_object_from_file(const cos_request_options_t *options,
                                      const cos_string_t *bucket,
                                      const cos_string_t *object,
                                      const cos_string_t *filename,
                                      cos_table_t *headers,
                                      cos_table_t **resp_headers,
                                      ci_operation_result_t **results)
```

功能说明

图片上传时识别二维码的请求包与 COS 简单上传文件接口一致，只需在请求包头部增加图片处理参数 Pic-Operations。

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
```

```
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 数据万象的访问域名，详情请参见 https://cloud.tencent.com/document/product/460/31066 文
static char TEST_CI_ENDPOINT[] = "https://ci.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_ci_image_qrcode()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    cos_table_t *resp_headers;
    cos_table_t *headers = NULL;
    ci_operation_result_t *results = NULL;
```

```
    ci_qrcode_info_t *content = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, "test.jpg");

    headers = cos_table_make(p, 1);
    apr_table_addn(headers, "pic-operations", "{\\"is_pic_info\\":1,\\"rules\\":[{\
    // 上传时识别
    cos_str_set(&file, "test.jpg");
    cos_str_set(&object, "test.jpg");
    s = ci_put_object_from_file(options, &bucket, &object, &file, headers, &resp_he
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("put object failed\\n");
    }
    printf("CodeStatus: %d\\n", results->object.code_status);
    cos_list_for_each_entry(ci_qrcode_info_t, content, &results->object.qrcode_info
        printf("CodeUrl: %s\\n", content->code_url.data);
        printf("Point: %s\\n", content->point[0].data);
        printf("Point: %s\\n", content->point[1].data);
        printf("Point: %s\\n", content->point[2].data);
        printf("Point: %s\\n", content->point[3].data);
    }

    //销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);
```

```
    test_ci_image_qrcode();

    cos_http_io_deinitialize();

    return 0;
}
```

参数说明

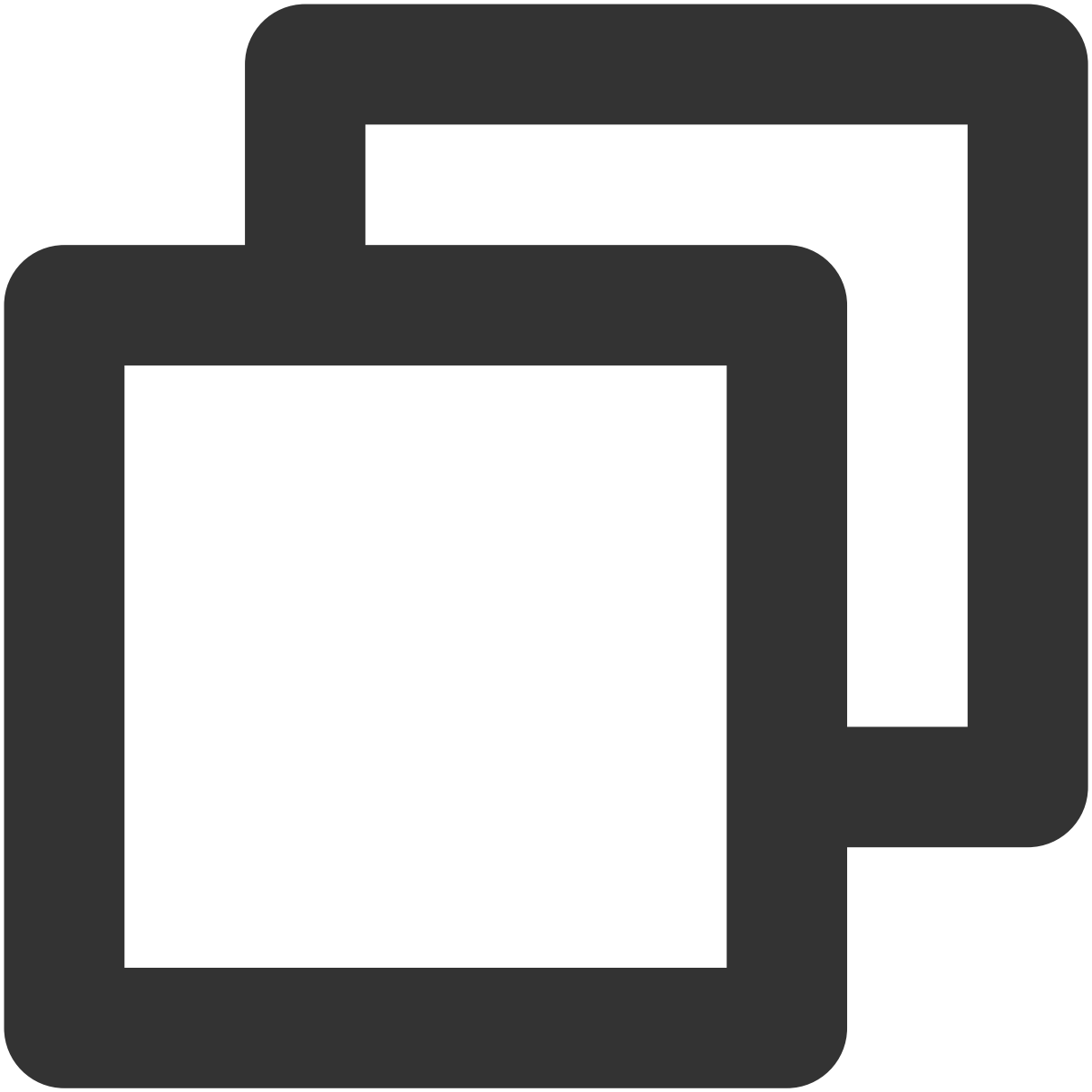
参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，存储桶的命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String
object	Object 名称	String
filename	Object 本地保存文件名称	String
headers	COS 请求附加头域	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
results	云上数据处理返回信息	Struct

下载时识别

功能说明

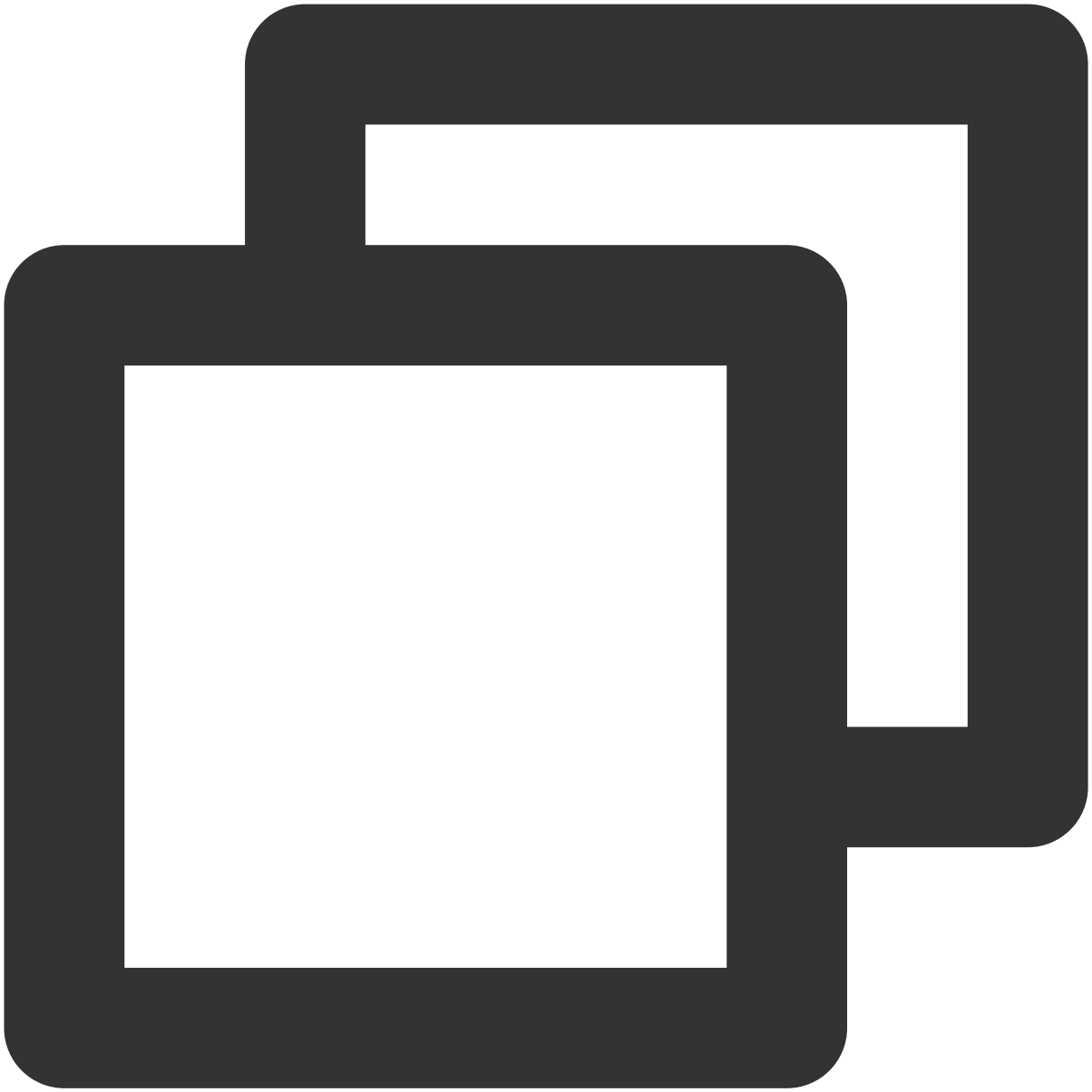
下载文件时进行二维码识别。

方法原型



```
cos_status_t *ci_get_qrcode(const cos_request_options_t *options,
                            const cos_string_t *bucket,
                            const cos_string_t *object,
                            int cover,
                            cos_table_t *headers,
                            cos_table_t *query_params,
                            cos_table_t **resp_headers,
                            ci_qrcode_result_t **results)
```

请求示例



```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// endpoint 是 COS 访问域名信息，详情请参见 https://cloud.tencent.com/document/product/
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// 数据万象的访问域名，详情请参见 https://cloud.tencent.com/document/product/460/31066 文
static char TEST_CI_ENDPOINT[] = "https://ci.ap-guangzhou.myqcloud.com";
// 开发者拥有的项目身份ID/密钥，可在 https://console.tencentcloud.com/cam/capi 页面获取
static char *TEST_ACCESS_KEY_ID;                //your secret_id
```

```
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// 开发者访问 COS 服务时拥有的用户维度唯一资源标识，用以标识资源，可在 https://console.tencent
static char TEST_APPID[] = "<APPID>";    //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666,
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
    if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
    if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_ci_get_image_qrcode()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_status_t *s = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_table_t *resp_headers;
    ci_qrcode_info_t *content = NULL;
    ci_qrcode_result_t *result2 = NULL;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, "test.jpg");
```

```
    // 下载时识别
    s = ci_get_qrcode(options, &bucket, &object, 1, NULL, NULL, &resp_headers, &res
    log_status(s);
    if (!cos_status_is_ok(s)) {
        printf("get object failed\\n");
    }
    printf("CodeStatus: %d\\n", result2->code_status);
    cos_list_for_each_entry(ci_qrcode_info_t, content, &result2->qrcode_info, node)
        printf("CodeUrl: %s\\n", content->code_url.data);
        printf("Point: %s\\n", content->point[0].data);
        printf("Point: %s\\n", content->point[1].data);
        printf("Point: %s\\n", content->point[2].data);
        printf("Point: %s\\n", content->point[3].data);
    }
    printf("ImageResult: %s\\n", result2->result_image.data);

    //销毁内存池
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // 通过环境变量获取 SECRETID 和 SECRETKEY
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_ci_get_image_qrcode();

    cos_http_io_deinitialize();

    return 0;
}
```

参数说明

参数名称	参数描述	类型
options	COS 请求选项	Struct
bucket	存储桶名称，Bucket 的命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	string
object	Object 名称	string
cover	二维码覆盖功能，对识别出的二维码覆盖上马赛克。取值为0或1。0表示不开启二维码覆盖，1表示开启二维码覆盖	int
headers	COS 请求附加头域	Struct
query_params	COS 请求操作参数	Struct
resp_headers	返回 HTTP 响应消息的头域	Struct
results	二维码识别结果	Struct

版权所有：腾讯云计算（北京）有限责任公司

C++ SDK
快速入门

最近更新时间：2024-03-14 09:53:07

下载与安装

相关资源

对象存储 COS 的 XML C++ SDK 源码下载地址：

Linux 版本/Windows 版本/macOS 版本： XML Linux C++ SDK。

SDK 快速下载地址：XML C++ SDK。

示例 Demo 下载地址：COS XML C++ SDK 示例。

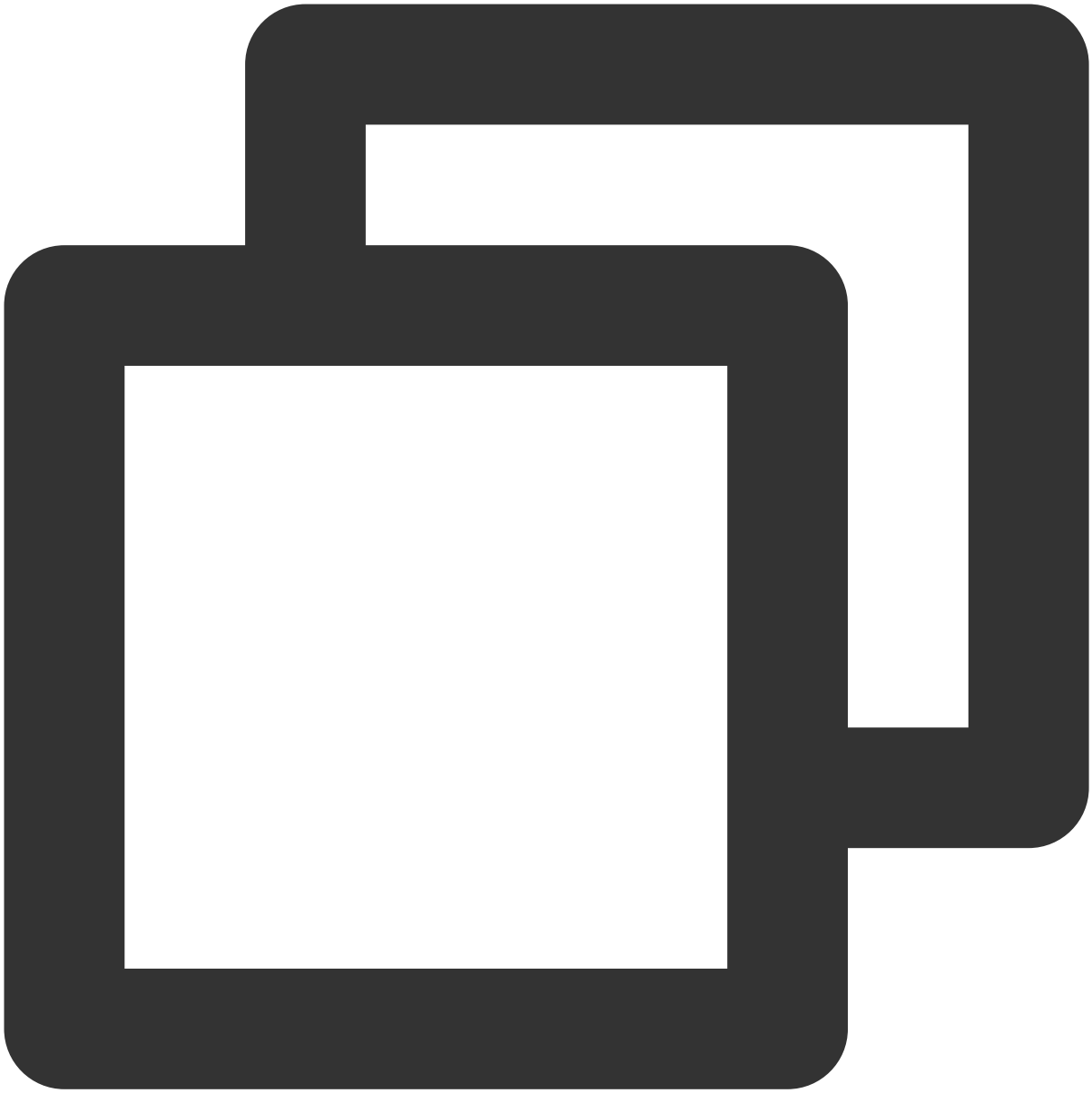
SDK 更新日志请参见 ChangeLog。

说明

如果您在使用 XML 版本 SDK 时遇到函数或方法不存在等错误，请先将 XML 版本 SDK 升级到最新版再重试。如果您仍在使用 JSON 版本 SDK，请升级到 XML C++ SDK。

已编译（推荐）

下载 XML C++ SDK 源码，libs 目录中有编译完成的库，可以直接使用，使用时请选择对应的系统版本。

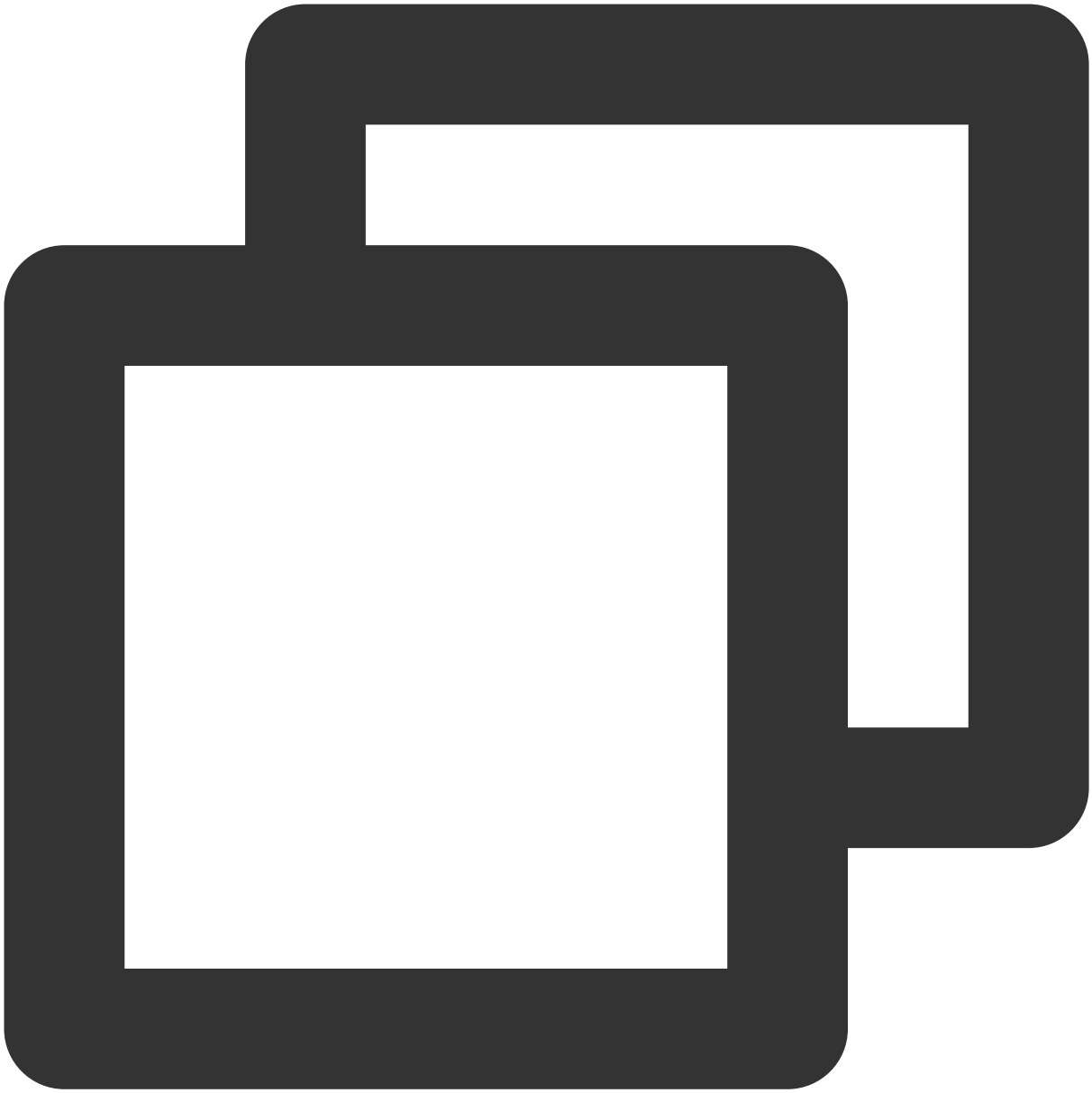


```
libs/linux/libcossdk.a #linux 的静态库，libcossdk.a 是基于 gcc version 4.8.5版本编译的，
libs/linux/libcossdk-shared.so #linux 动态库
libs/Win32/cossdk.lib #Win32库
libs/x64/cossdk.lib #Win64 库
libs/macOS/libcossdk.a #macOS 静态库
libs/macOS/libcossdk-shared.dylib #macOS 动态库
```

说明

使用时，请将对应系统的库文件和 SDK include 头文件拷贝至您的工程中。

Third-party 目录下有第三方依赖库，如下：



```
third_party/lib/linux/poco/ #linux下依赖的 poco 动态库, poco 库是基于 OpenSSL 1.0.2版本编
third_party/lib/Win32/openssl/ #Win32依赖的 openssl 库
third_party/lib/Win32/poco/ #Win32依赖的 poco 库
third_party/lib/x64/openssl/ #Win64依赖的 openssl 库
third_party/lib/x64/poco/ #Win64依赖的 poco 库
third_party/lib/macOS/poco/ #macOS 依赖的 poco 库
```

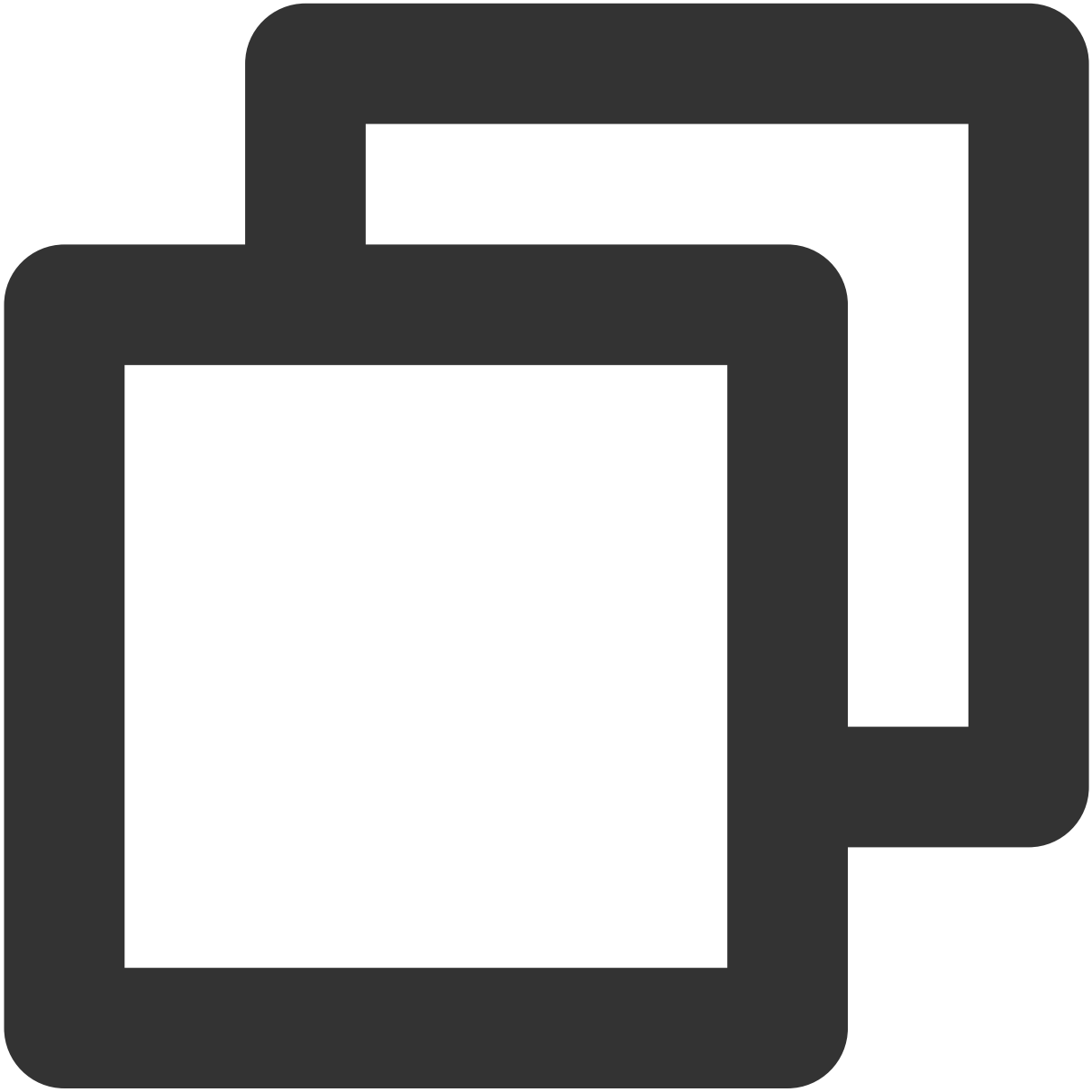
说明

使用时，请将对应系统的依赖库拷贝至您的工程中。

手动编译

编译选项

在根目录下的 CMakeLists.txt 文件可以配置编译选项，有以下编译选项：



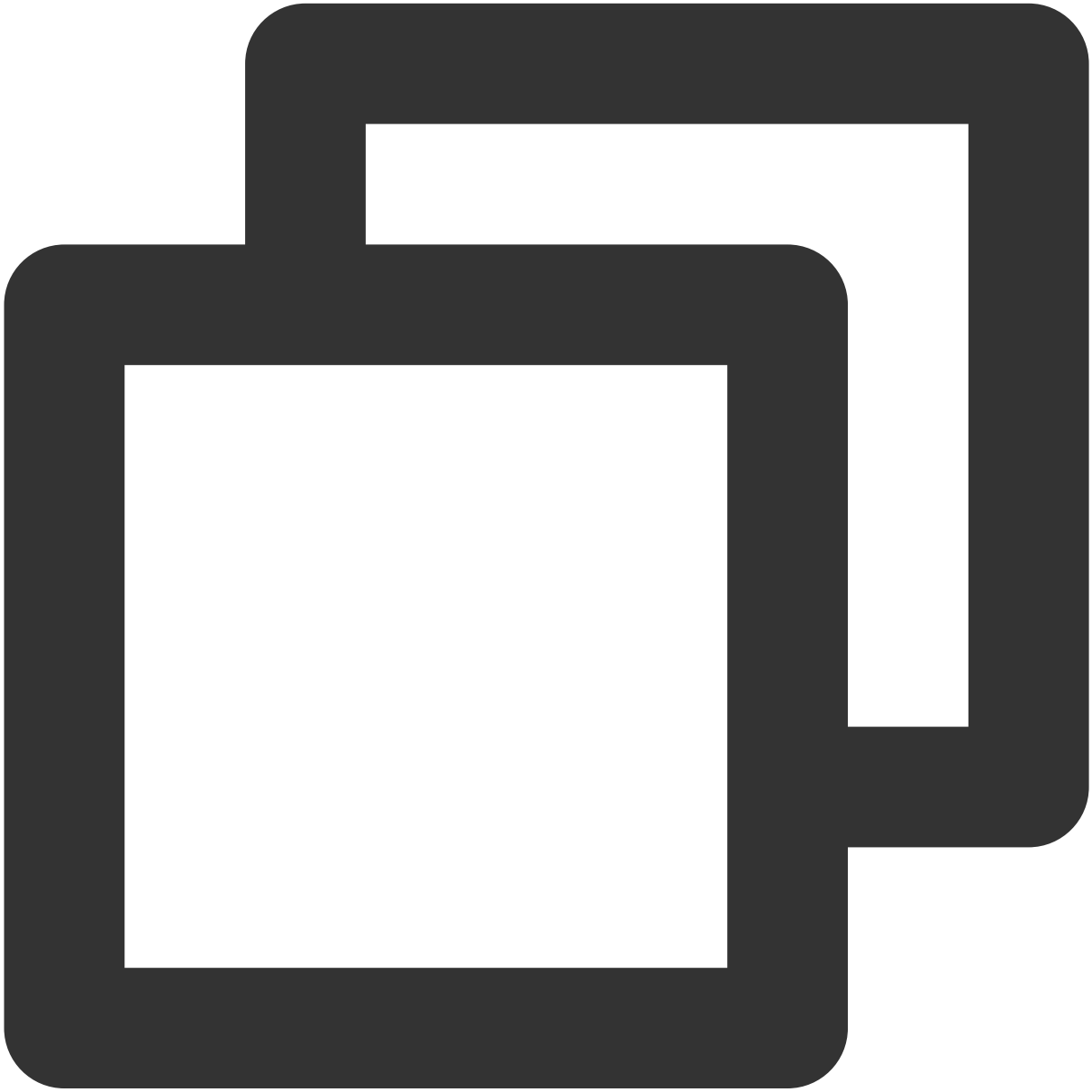
```
option(BUILD_UNITTEST "Build unittest" OFF) #配置编译单元测试
option(BUILD_DEMO "Build demo" ON) #配置编译 demo 测试代码
option(BUILD_SHARED_LIB "Build shared library" OFF) #配置编译动态库
```

依赖库

依赖动态库：poco、openssl、crypto。

编译 Linux 版本 SDK

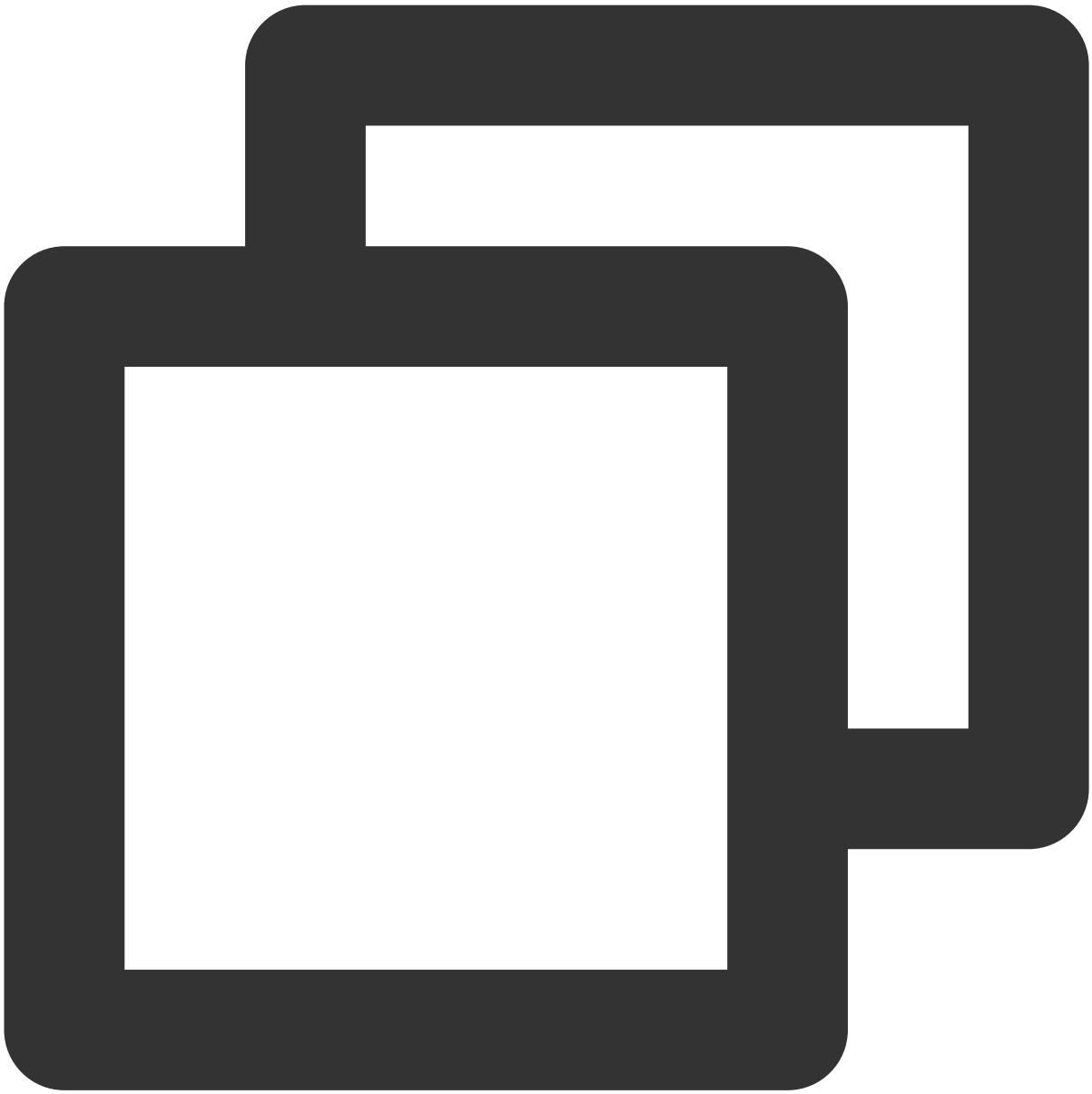
1. 安装编译工具及依赖库



```
yum install -y gcc gcc-c++ make cmake openssl
#cmake版本要求大于2.8
```

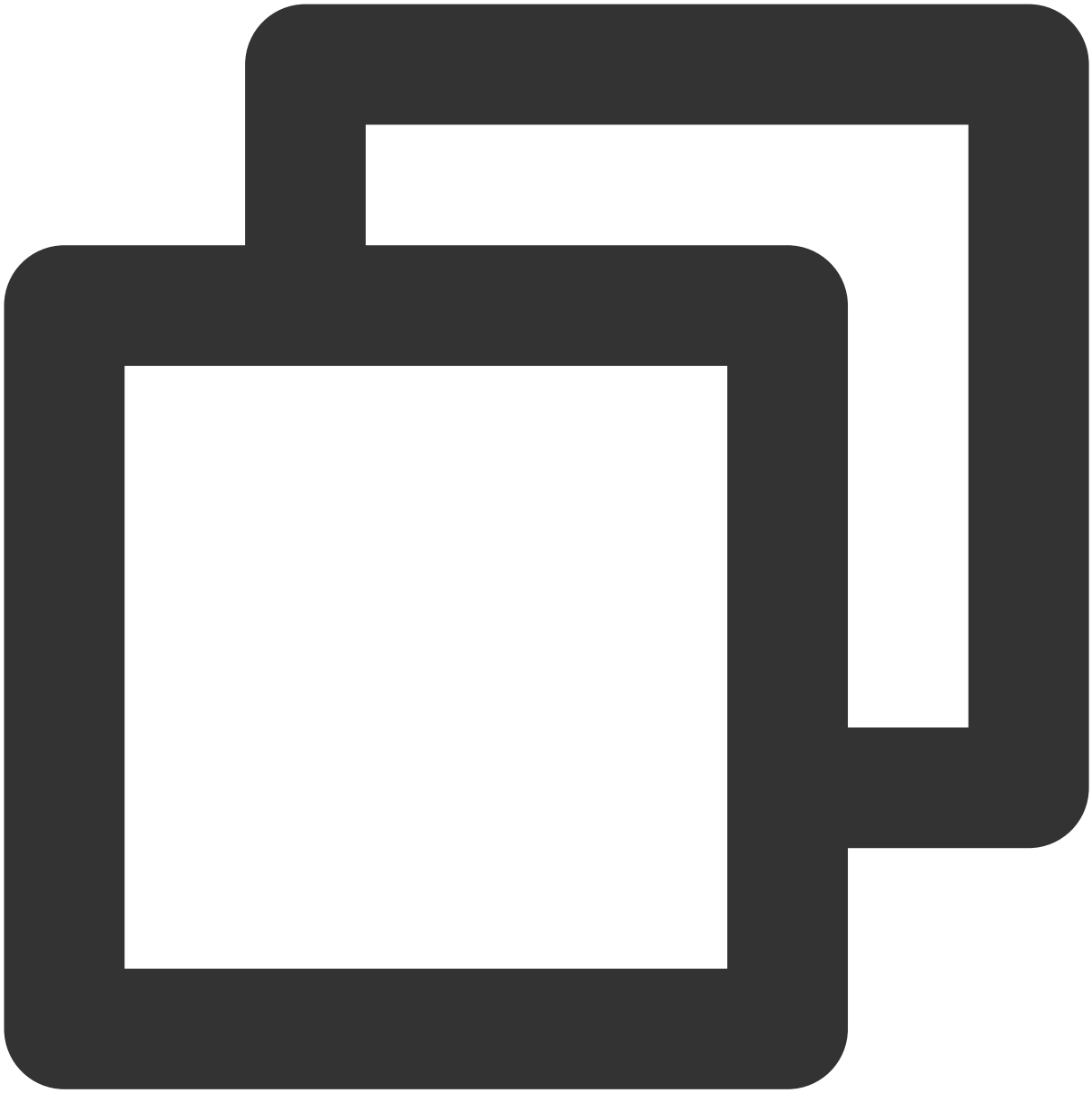
2. 编译 SDK

下载 XML C++ SDK 源码 到您的开发环境，并执行以下命令：



```
cd ${cos-cpp-sdk}
mkdir -p build
cd build
cmake ..
make
```

3. 安装 Poco 库



```
cd ${cos-cpp-sdk}
sh install-libpoco.sh
```

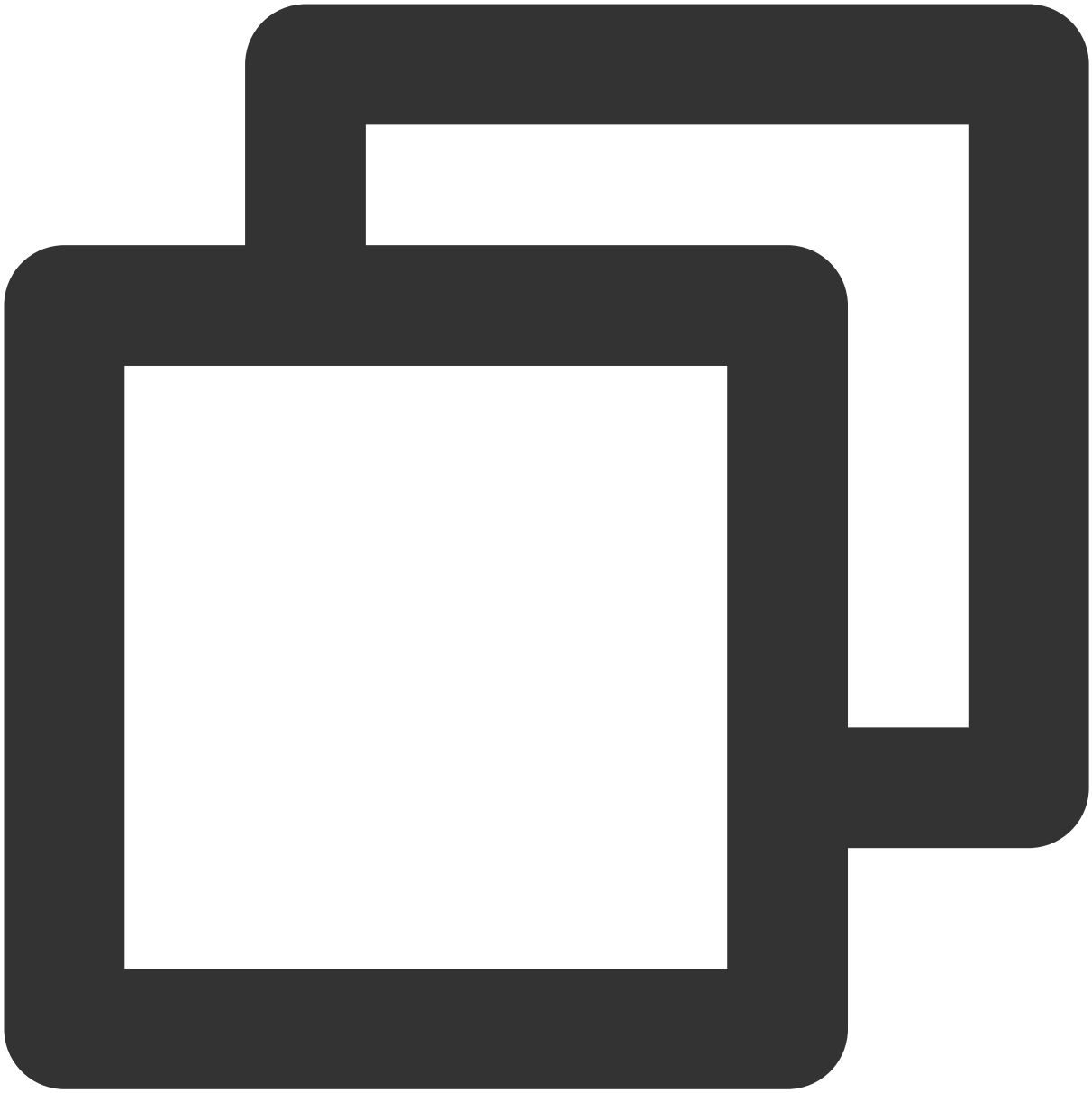
说明

该脚本将 Poco 动态库安装到 /usr/lib64 目录中，并创建软链接，如果要在生产环境中使用 COS SDK，也需要安装 Poco 库到生产环境中。

4. 测试 demo

说明

如果不需要测试 demo，可跳过此步骤。



```
cd ${cos-cpp-sdk}
vim demo/cos_demo.cpp  #修改 demo 中的存储桶名以及测试代码
vim CMakeLists.txt #修改根目录下的 CMakeLists.txt 中的 BUILD_DEMO 为 ON，开启编译demo
cd build && make #编译 demo
ls bin/cos_demo #生成的可执行文件在 bin 目录
vim bin/config.json #修改密钥和园区
cd bin && ./cos_demo #运行 demo
```

5. 使用 SDK

编译生成的库文件在 build/lib 目录中，静态库名称为 `libcossdk.a` ，动态库名称为 `libcossdk-`

`shared.so` 。使用时，请将库拷贝至您的工程中，同时将 include 目录拷贝至您的工程中的 include 路径下。

编译 Windows 版本 SDK

1. 安装 visual studio 2017

安装 visual studio 2017 开发环境。

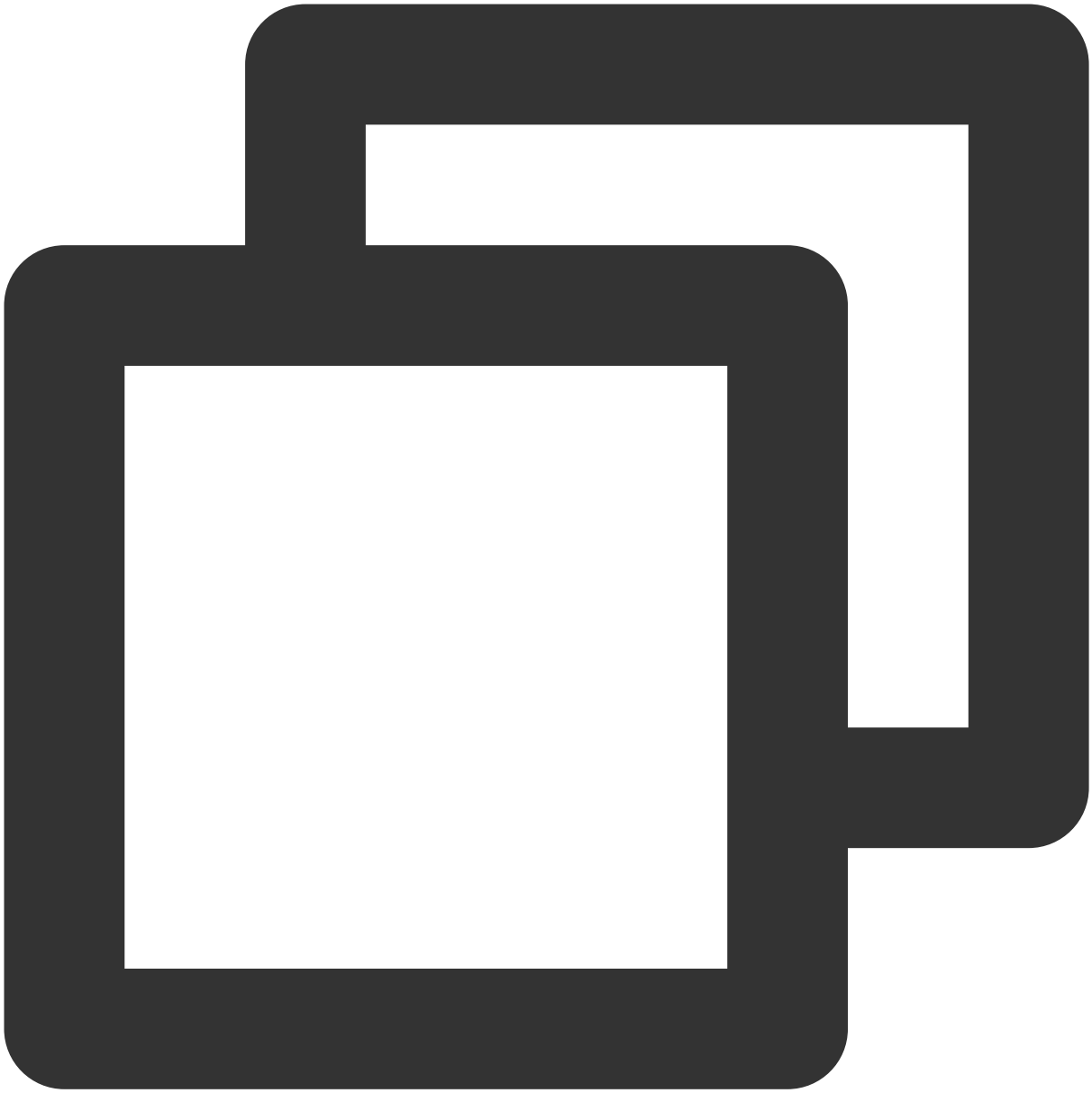
2. 安装 CMake 工具

从 CMake官网 下载 Windows 版本的 CMake 编译工具，并将 `${CMake的安装路径}\\bin` 配置在 Windows 的系统环境变量 Path 中。

3. 编译 SDK

i. 下载 XML Windows C++ SDK 源码 到您的开发环境。

ii. 打开 Windows 的命令行，cd 到 C++ SDK 的源码目录下，执行以下命令：



```
mkdir build
cd build
cmake .. #生成 Win32 makefile
cmake -G "Visual Studio 15 Win64" .. #生成 Win64 makefile
```

iii. 使用 visual studio 2017 打开解决方案资源管理器，进行编译。

4. 测试 demo

说明

如果不需要测试 demo，可跳过此步骤。

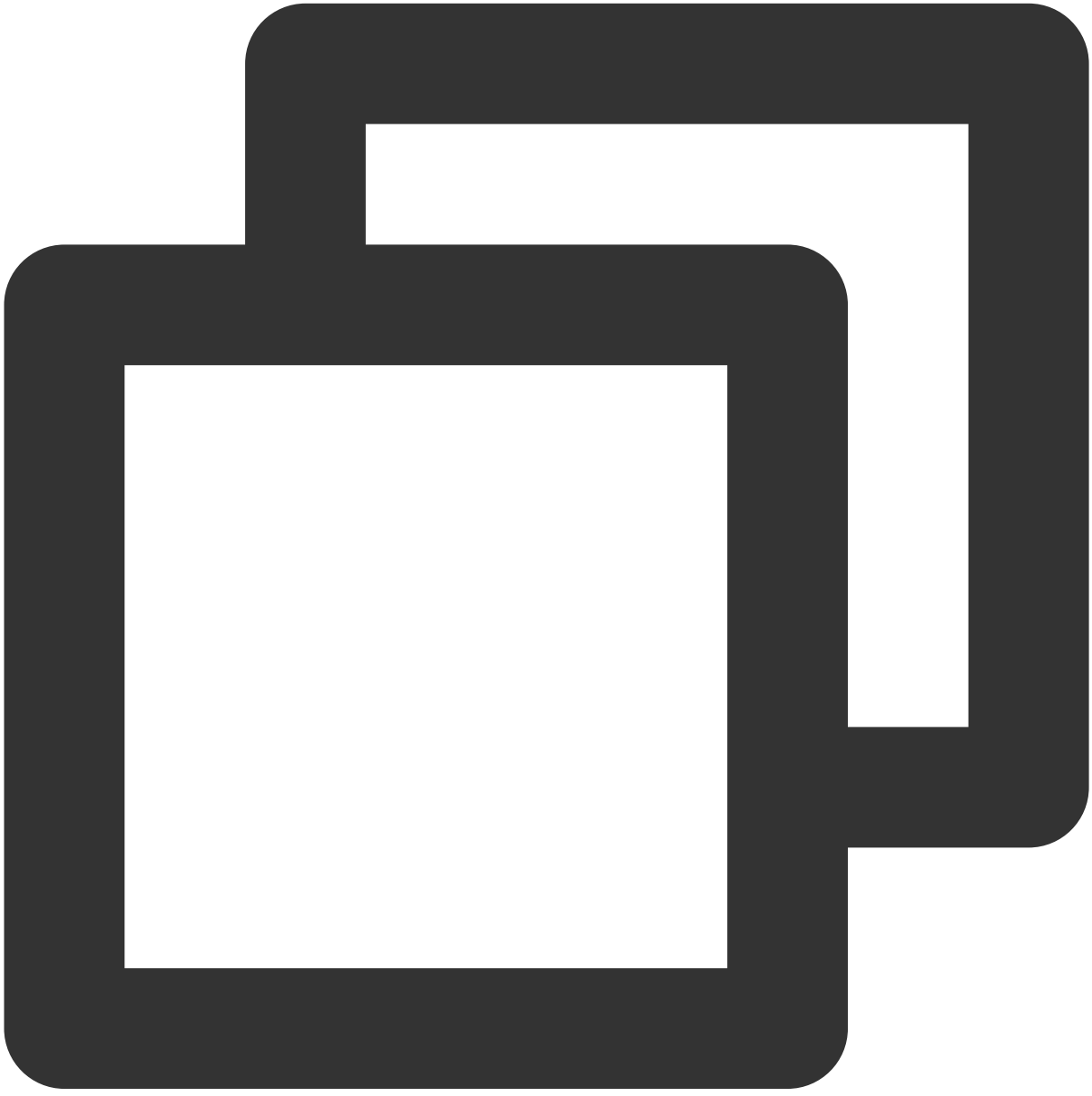
修改 demo 代码并编译，生成的 cos_demo.exe 在 bin 目录中，修改 bin/config.json 可运行 cos_demo.exe。

5. 使用 SDK

编译生成的库文件在 build/Release 目录中，静态库名称为 `cossdk.lib` 。使用时，请将库拷贝至您的工程中，同时将 include 目录拷贝至您的工程中的 include 路径下。

编译 Mac 版本 SDK

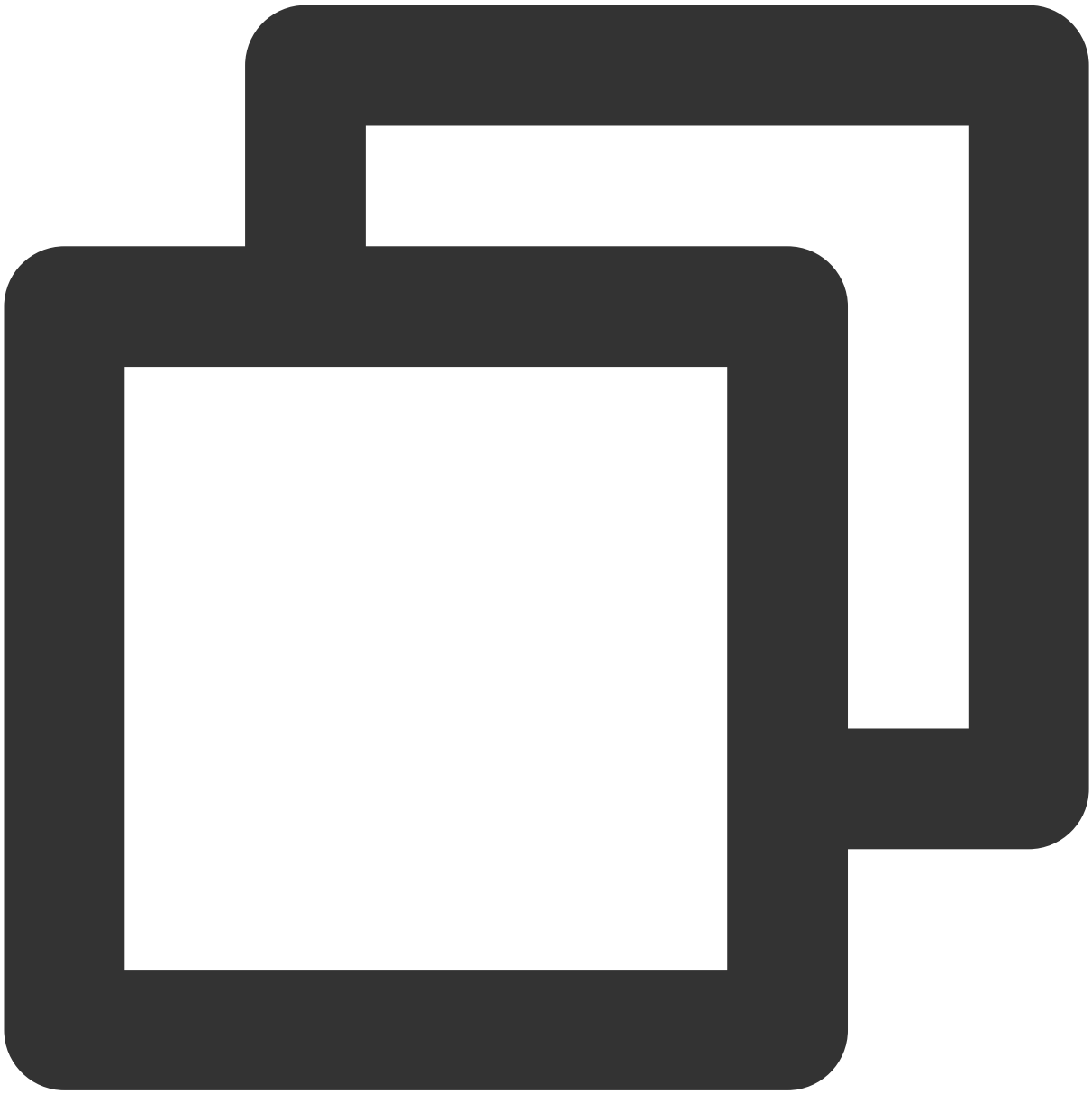
1. 安装编译工具及依赖库



```
brew install gcc make cmake openssl
```

2. 编译 SDK

下载 XML C++ SDK 源码 到您的开发环境，并执行以下命令：



```
cd ${cos-cpp-sdk}
mkdir -p build
cd build
cmake ..
make
```

3. 安装 Poco 库

Poco 库在 third_party/lib/macOS/poco 目录下，请自行安装。

4. 测试 demo

说明

如果不需要测试 demo，可跳过此步骤。

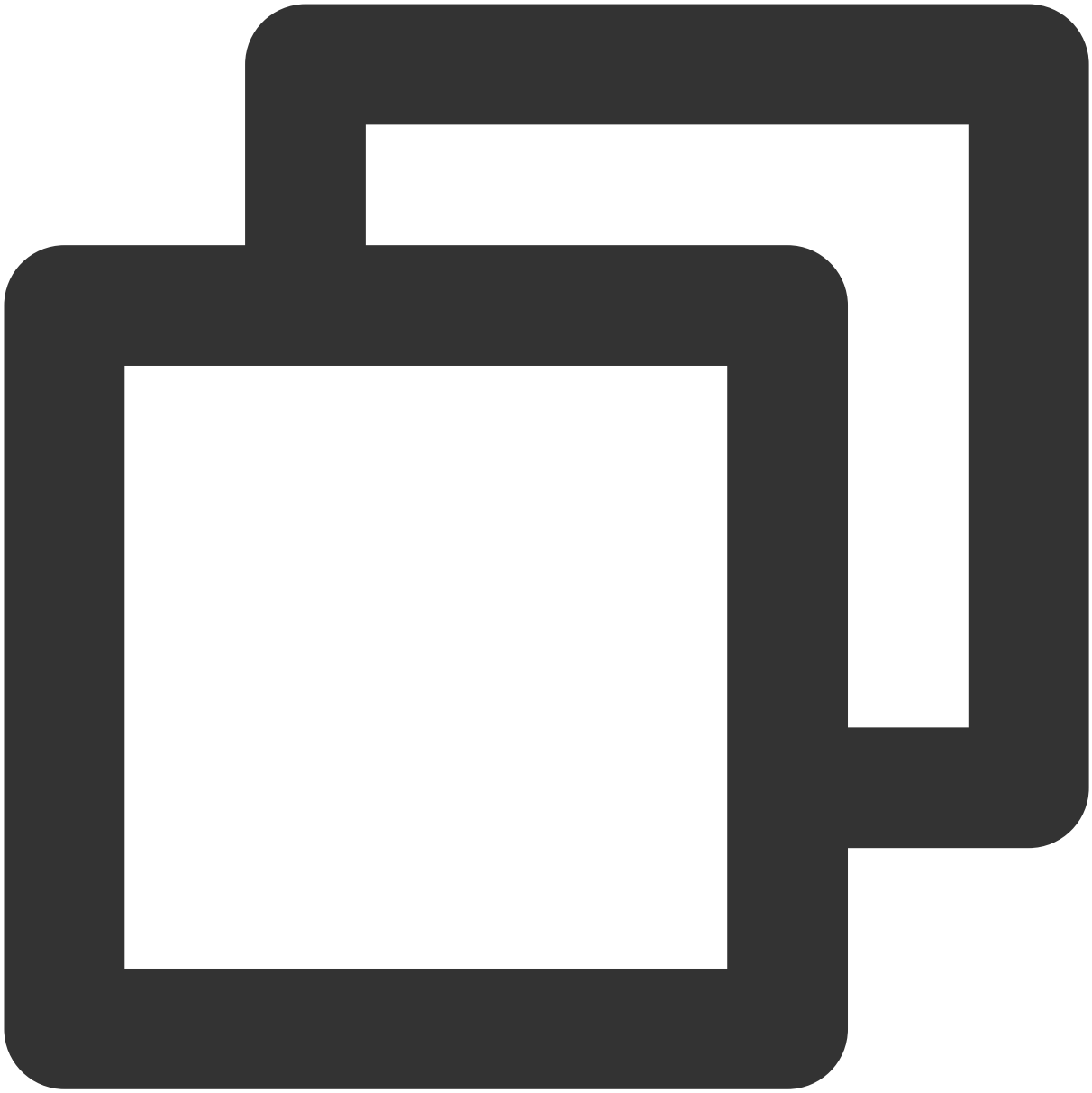
修改 demo 代码并编译，生成的 cos_demo 在 bin 目录中，拷贝 cos-cpp-sdk-v5/demo/config.json 到 bin 目录下，修改 bin/config.json 可运行 cos_demo。

5. 使用 SDK

编译生成的库文件在 build/lib 目录中，静态库名称为 `libcossdk.a`，动态库名称为 `libcossdk-shared.dylib`。使用时，请将库拷贝至您的工程中，同时将 include 目录拷贝至您的工程中的 include 路径下。

常见编译问题

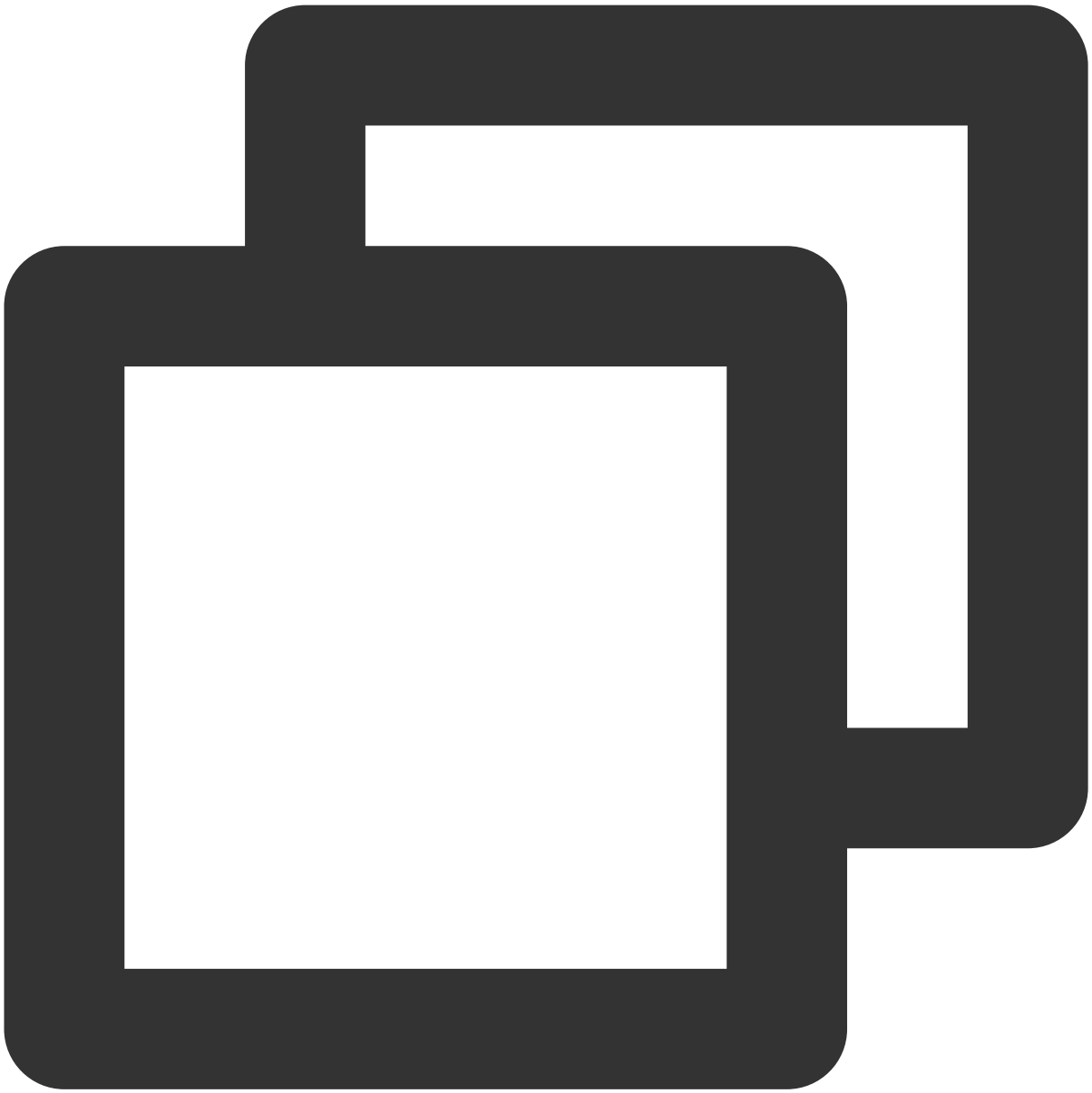
1. 编译可执行程序的时候提示错误：



```
PocoCrypto.so.64: undefined reference to `PEM_write_bio_PrivateKey@libcrypto.so.10'
libPocoNetSSL.so.64: undefined reference to `X509_check_host@libcrypto.so.10'
ibPocoCrypto.so.64: undefined reference to `ECDSA_sign@OPENSSL_1.0.1_EC'
libPocoCrypto.so.64: undefined reference to `CRYPTO_set_id_callback@libcrypto.so.10
ibPocoCrypto.so.64: undefined reference to `EVP_PKEY_id@libcrypto.so.10'
libPocoNetSSL.so.64: undefined reference to `SSL_get1_session@libssl.so.10'
libPocoNetSSL.so.64: undefined reference to `SSL_get_shutdown@libssl.so.10'
libPocoCrypto.so.64: undefined reference to `EVP_PKEY_set1_RSA@libcrypto.so.10'
libPocoCrypto.so.64: undefined reference to `SSL_load_error_strings@libssl.so.10'
```

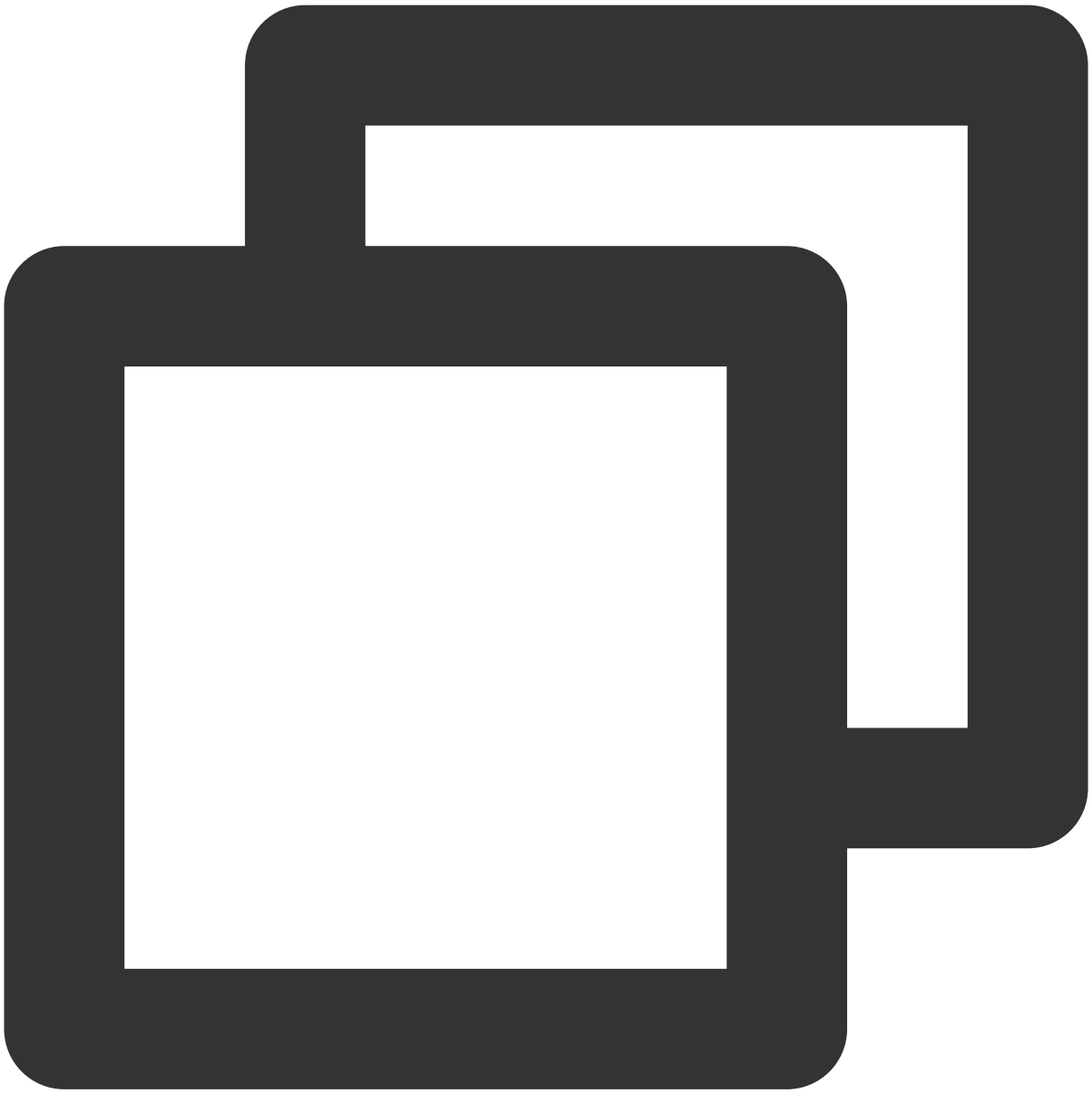
版权所有：腾讯云计算（北京）有限责任公司

这种情况一般是工程里自带的 poco 库的编译依赖的 SSL 版本与客户机器上的版本不一致导致的，需要用户重新编译 poco 库，并替换掉 third_party 里的poco库。



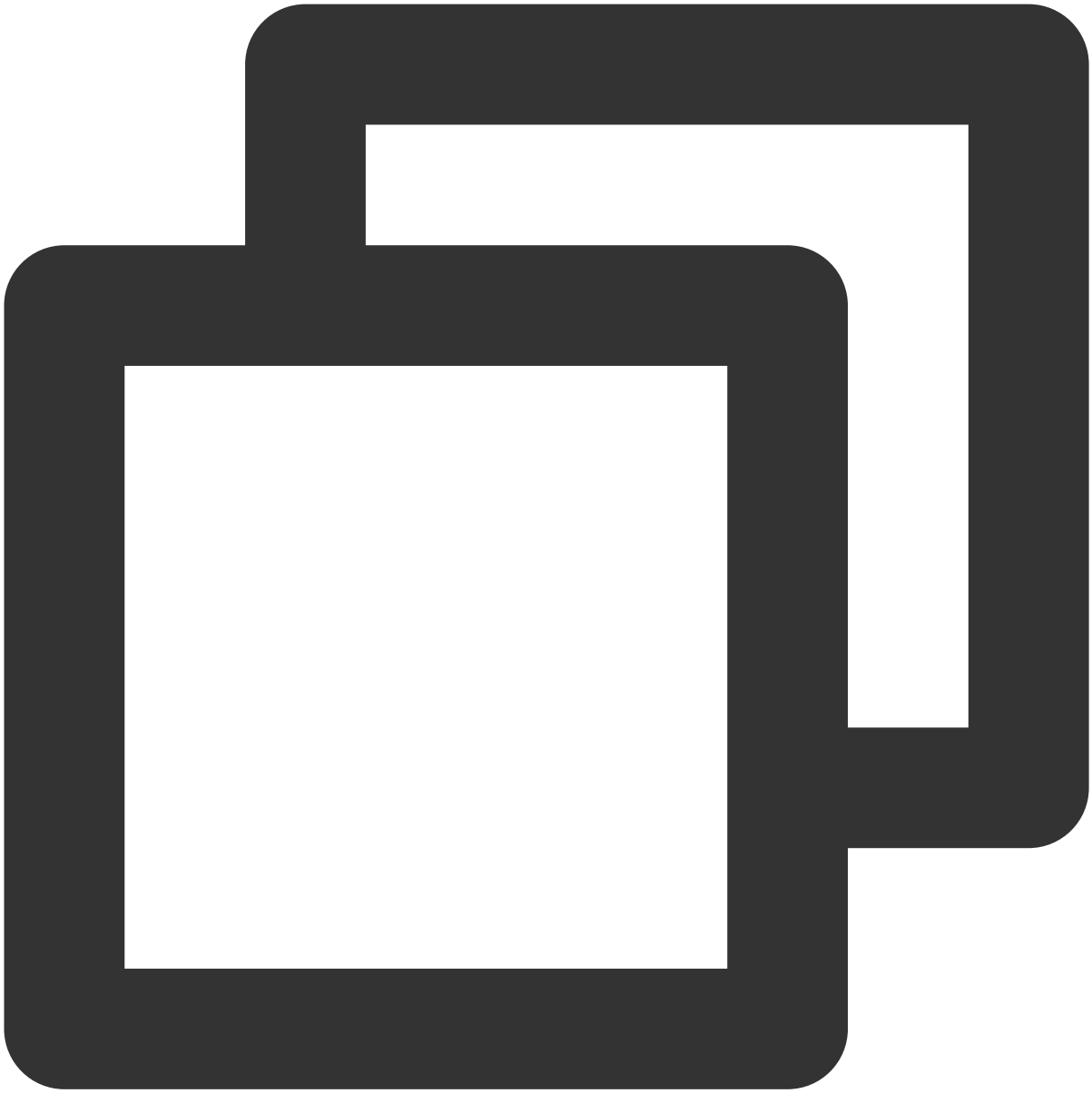
```
wget https://github.com/pocoproject/poco/archive/refs/tags/poco-1.9.4-release.zip
cd poco-poco-1.9.4-release/
./configure --omit=Data/ODBC,Data/MySQL
mkdir my_build
cd my_build
cmake ..
make -j5
```

2. 编译 poco 库的时候无法编译出 PocoNetSSL 库，一般是因为机器没装 openssl-devel 库。



```
yum install -y openssl-devel
```

3. 编译可执行程序的时候提示错误：



```
undefined reference to `qcloud_cos::CosConfig::CosConfig(std::__cxx11::basic_string
```

这种情况一般是因为工程自带的 libcossdk.a 编译使用的 gcc 版本与客户机器上的 gcc 版本不一致导致的，需要客户重新编译 poco 库和 libcossdk。

开始使用

下面为您介绍如何使用 COS C++ SDK 完成一个基础操作，如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。

说明

关于文章中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。

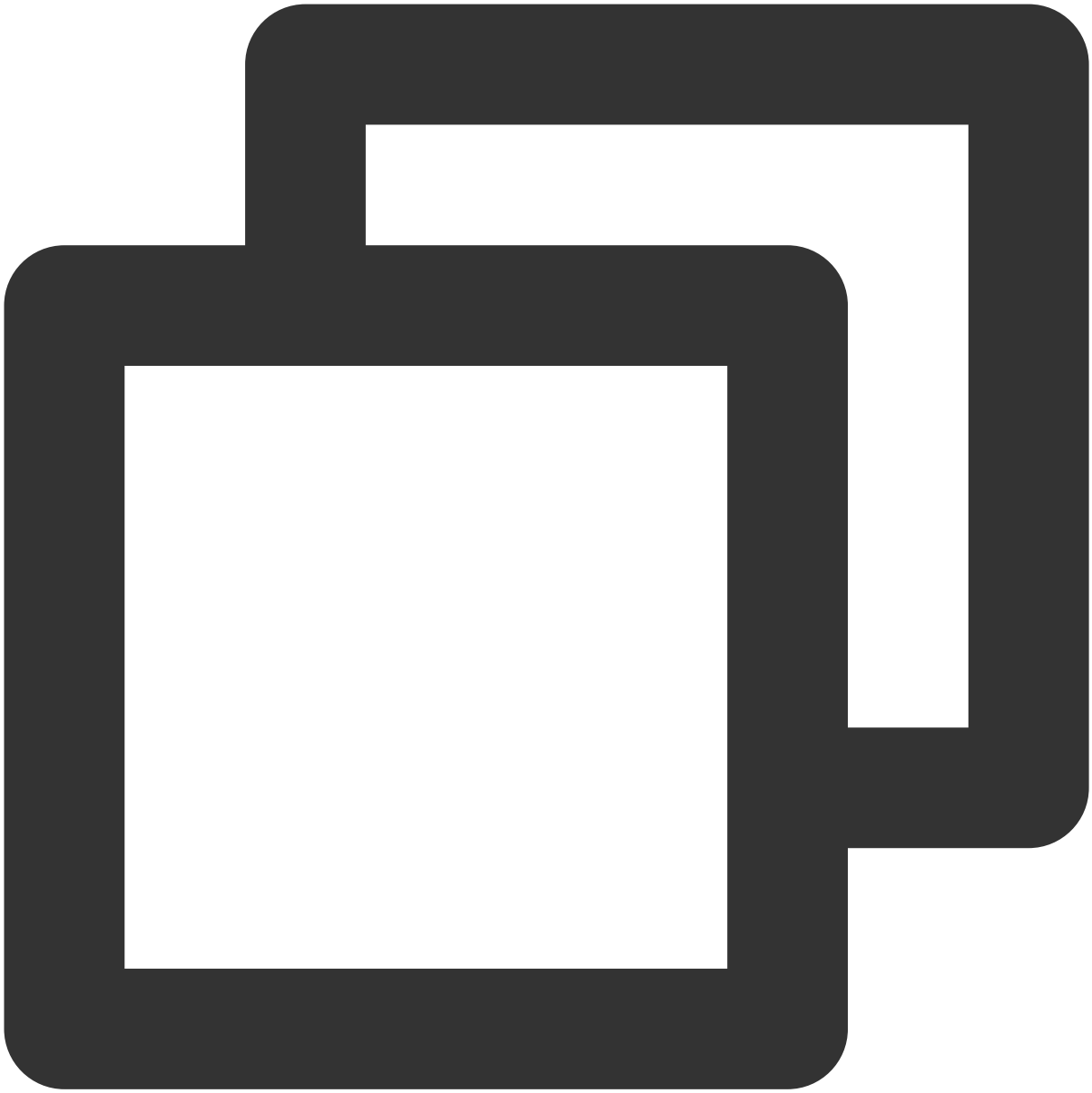
初始化

注意

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

配置文件各字段介绍：

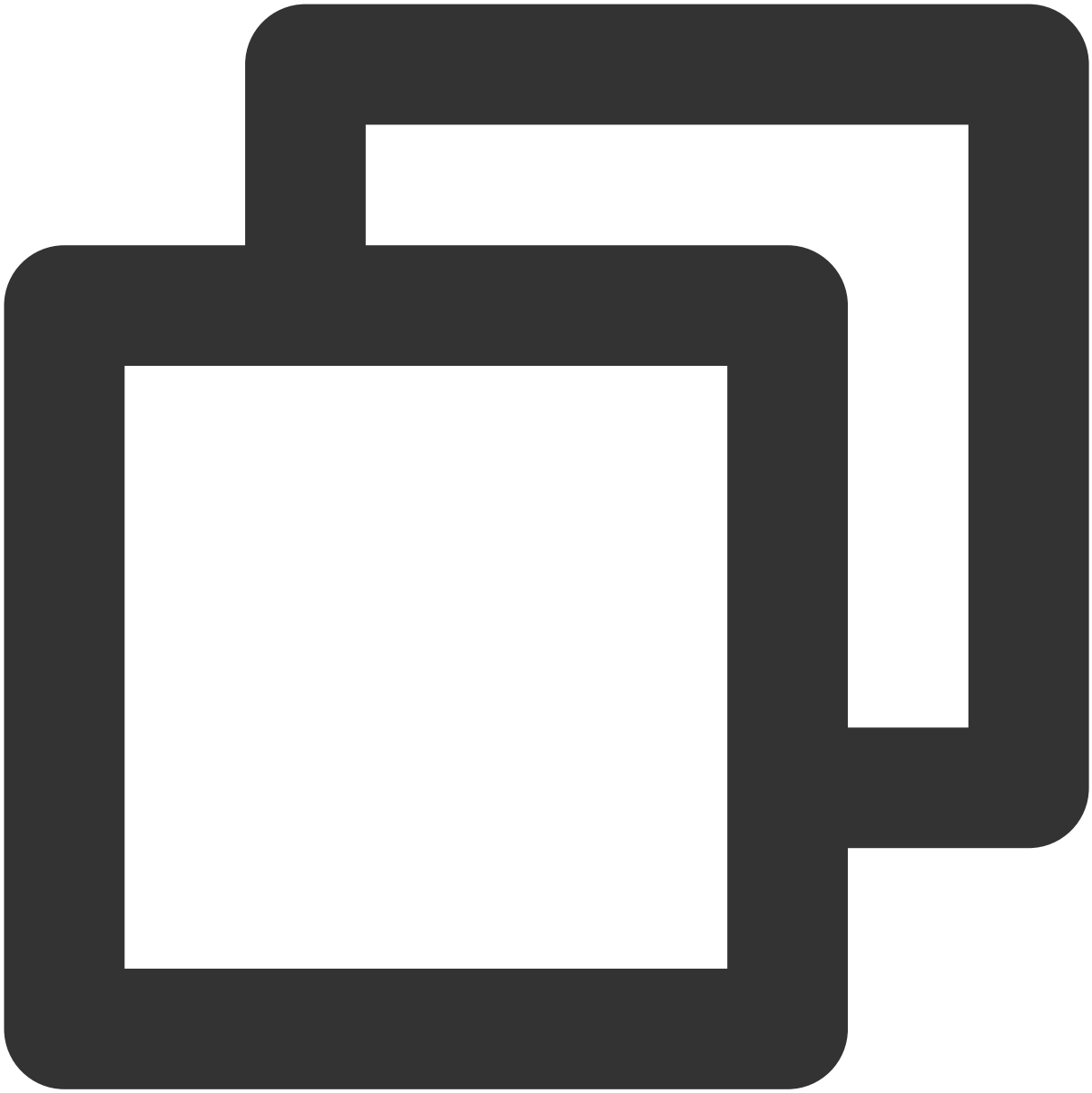


```
{
"SecretId":"********************************", // sercret_id 替换为用户的 SecretId，登
"SecretKey":"********************************", // sercret_key 替换为用户的 SecretKey，
"Region":"ap-guangzhou",                    // 存储桶地域，替换为客户存储桶所在地域，可以在COS
"SignExpiredTime":360,                 // 签名超时时间，单位s
"ConnectTimeoutInms":6000,             // connect 超时时间，单位ms
"ReceiveTimeoutInms":60000,            // recv 超时时间，单位ms
"UploadPartSize":10485760,             // 上传文件分块大小，1M~5G，默认为10M
"UploadCopyPartSize":20971520,         // 上传复制文件分块大小，5M~5G，默认为20M
"UploadThreadPoolSize":5,              // 单文件分块上传线程池大小
"DownloadSliceSize":4194304,           // 下载文件分块大小
```

```
"DownloadThreadPoolSize":5,          // 单文件下载线程池大小
"AsynThreadPoolSize":2,              // 异步上传下载线程池大小
"LogoutType":1,                      // 日志输出类型,0:不输出,1:输出到屏幕,2输出到syslog
"LogLevel":3,                        // 日志级别:1: ERR, 2: WARN, 3:INFO, 4:DBG
"IsDomainSameToHost":false,          // 是否使用专有的 host
"DestDomain":"",                     // 特定 host
"IsUseIntranet":false,               // 是否使用特定ip和端口号
"IntranetAddr":""                    // 特定 ip 和端口号,例如“127.0.0.1:80”
}
```

使用临时密钥访问 COS

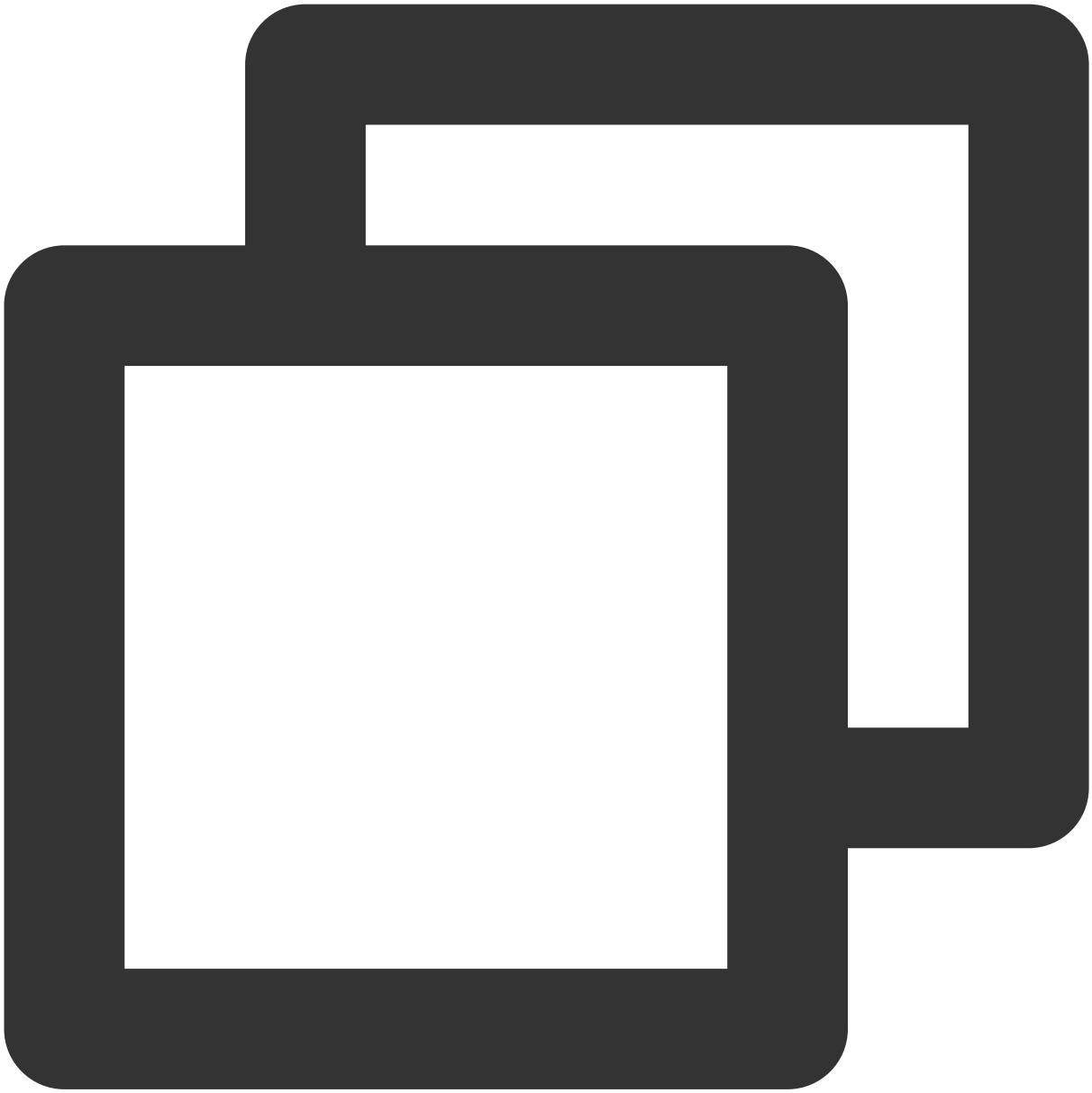
如果您需要使用临时密钥访问 COS，请参见如下代码：



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"
int main(int argc, char *argv[]) {
    qcloud_cos::CosConfig config("./config.json");
    // 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 htt
    config.SetTmpToken("xxx");
    qcloud_cos::CosAPI cos(config);
}
```

将 SDK 内部日志打印到自定义日志文件

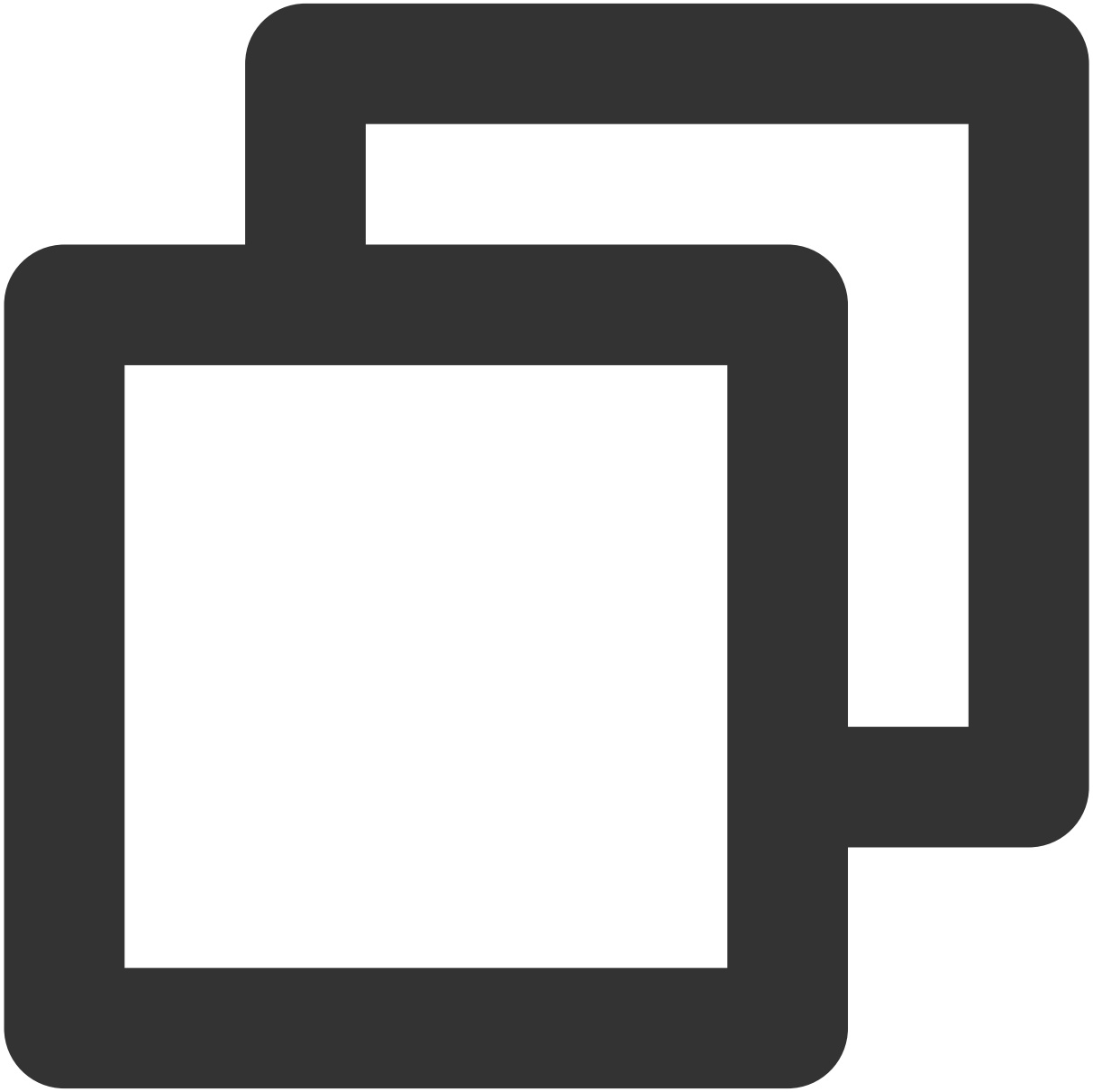
如果您需要将 SDK 内部日志打印到自定义日志文件，特别是 Windows 系统，则参考如下代码：



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"
void TestLogCallback(const std::string& log) {
    std::ofstream ofs;
    ofs.open("test.log", std::ios_base::app);
    ofs << log;
    ofs.close();
```

```
}
int main(int argc, char** argv) {
    qcloud_cos::CosConfig config("./config.json");
    config.SetLogCallback(&TestLogCallback);
    qcloud_cos::CosAPI cos(config);
}
```

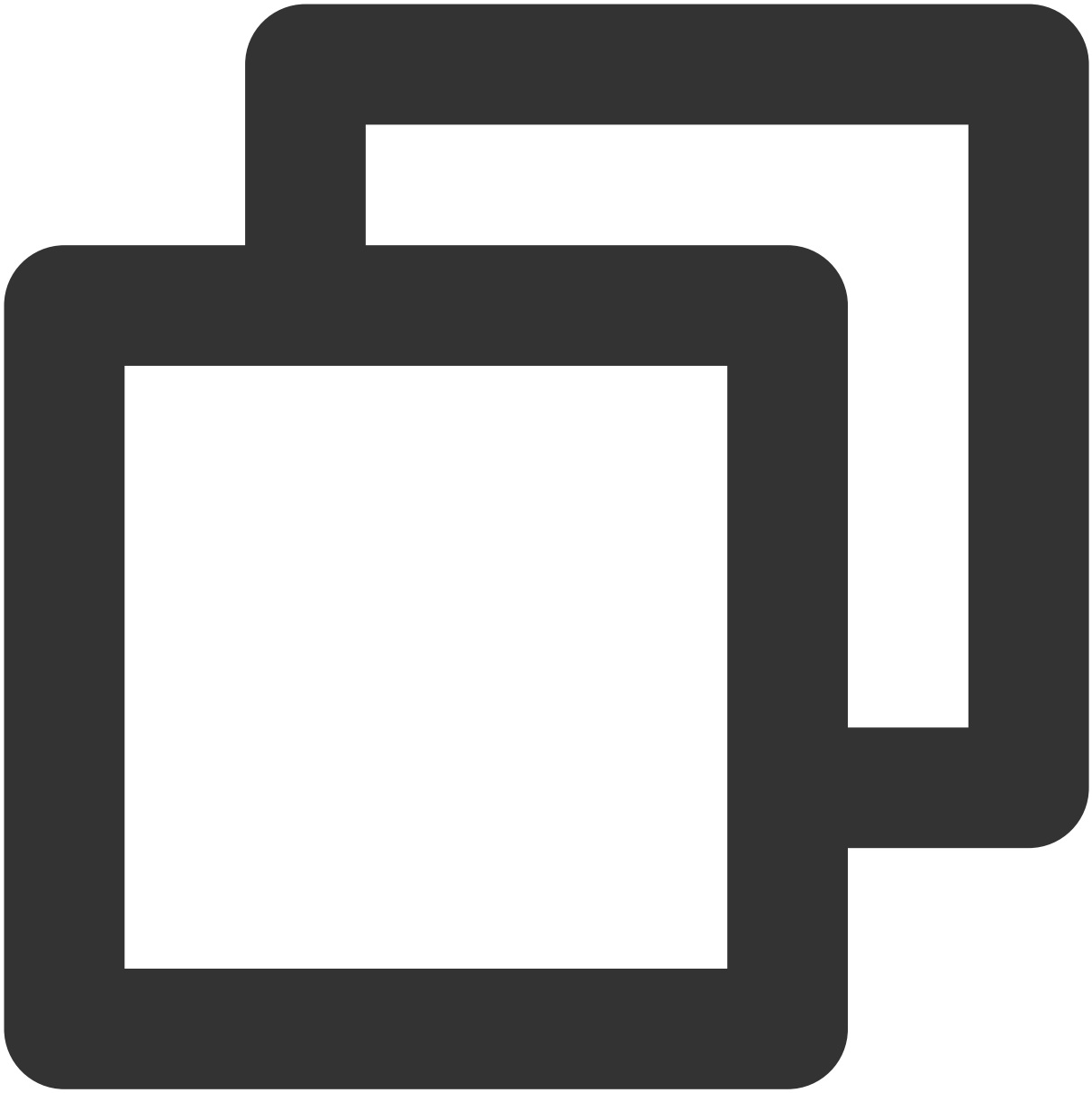
创建存储桶



```
#include "cos_api.h"
```

```
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造创建存储桶的请求
    std::string bucket_name = "examplebucket-1250000000"; // 替换为用户的存储桶名，由bu
    qcloud_cos::PutBucketReq req(bucket_name);
    qcloud_cos::PutBucketResp resp;

    // 3. 调用创建存储桶接口
    qcloud_cos::CosResult result = cos.PutBucket(req, &resp);

    // 4. 处理调用结果
    if (result.IsSucc()) {
        // 创建成功
    } else {
        // 创建存储桶失败，可以调用 CosResult 的成员函数输出错误信息，例如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

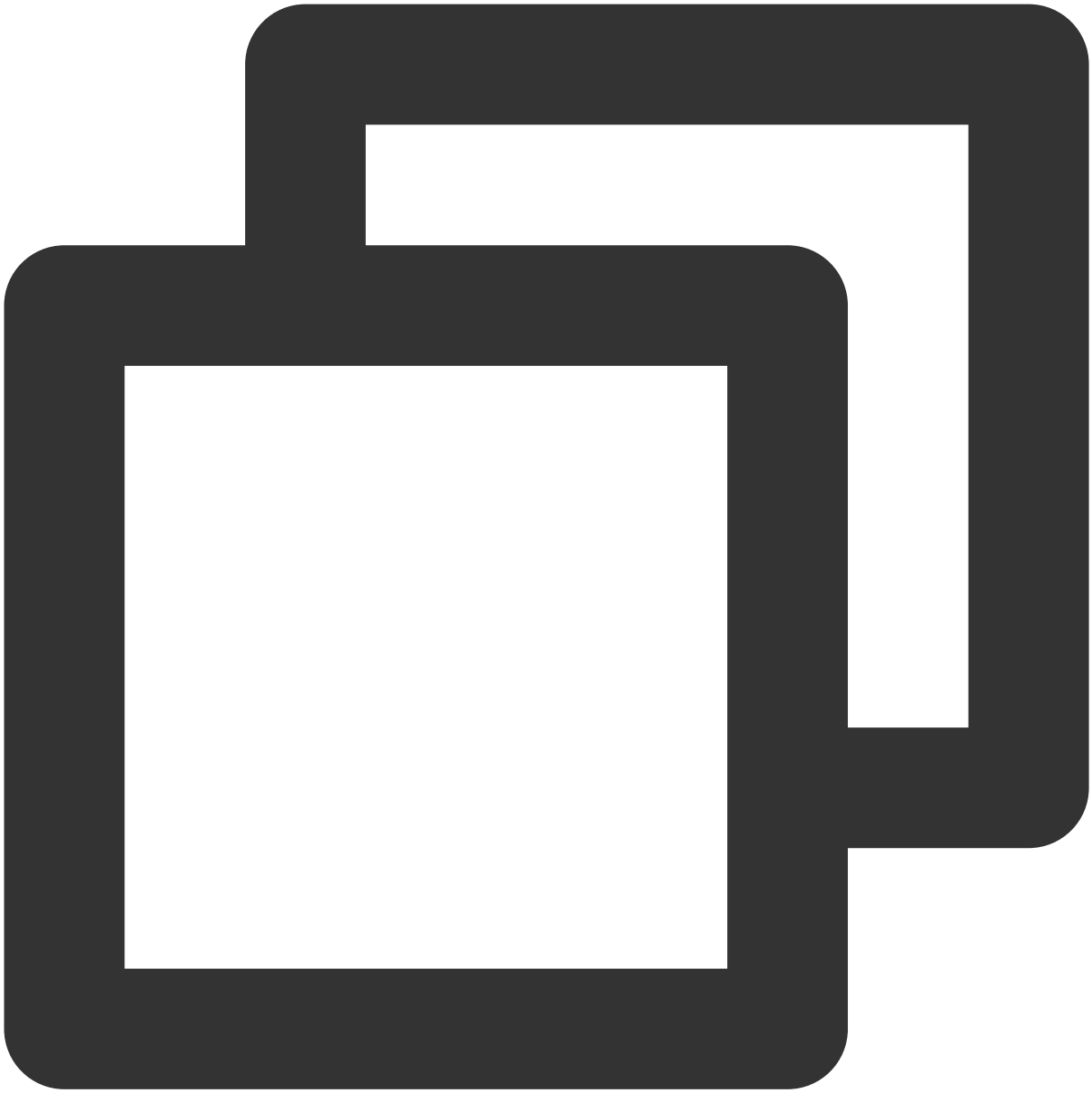
查询存储桶列表



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造查询存储桶列表的请求
    qcloud_cos::GetServiceReq req;
```

```
    qcloud_cos::GetServiceResp resp;
    qcloud_cos::CosResult result = cos.GetService(req, &resp);

    // 3. 获取响应信息
    const qcloud_cos::Owner& owner = resp.GetOwner();
    const std::vector<qcloud_cos::Bucket>& buckets = resp.GetBuckets();
    std::cout << "owner.m_id=" << owner.m_id << ", owner.display_name=" << owner.m_

    for (std::vector<qcloud_cos::Bucket>::const_iterator itr = buckets.begin(); itr
        const qcloud_cos::Bucket& bucket = *itr;
        std::cout << "Bucket name=" << bucket.m_name << ", location="
            << bucket.m_location << ", create_date=" << bucket.m_create_date << std
    }

    // 4. 处理调用结果
    if (result.IsSucc()) {
        // 查询存储桶列表成功
    } else {
        // 查询存储桶列表失败，可以调用 CosResult 的成员函数输出错误信息，比如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

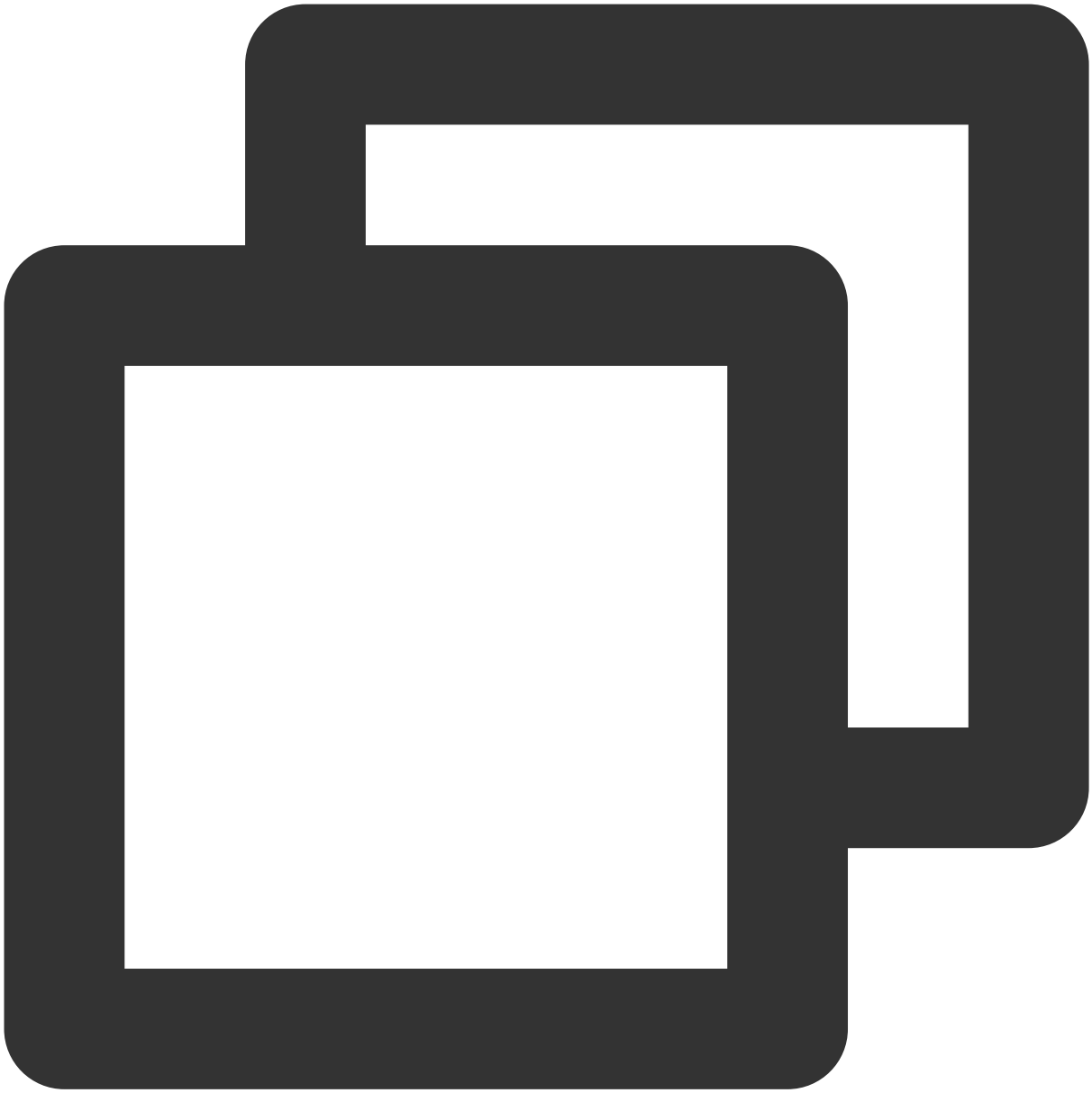
上传对象



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造上传文件的请求
    std::string bucket_name = "examplebucket-1250000000"; // 替换为用户的存储桶名，由bu
```

```
    std::string object_name = "exampleobject"; //exampleobject 即为对象键（Key），是对
    qcloud_cos::PutObjectByFileReq req(bucket_name, object_name, "/path/to/local/fi
    //req.SetXCosStorageClass("STANDARD_IA"); // 默认为 STANDARD，可以调用 Set 方法设置
    qcloud_cos::PutObjectByFileResp resp;

    // 3. 调用上传文件接口
    qcloud_cos::CosResult result = cos.PutObject(req, &resp);

    // 4. 处理调用结果
    if (result.IsSucc()) {
        // 上传文件成功
    } else {
        // 上传文件失败，可以调用 CosResult 的成员函数输出错误信息，比如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

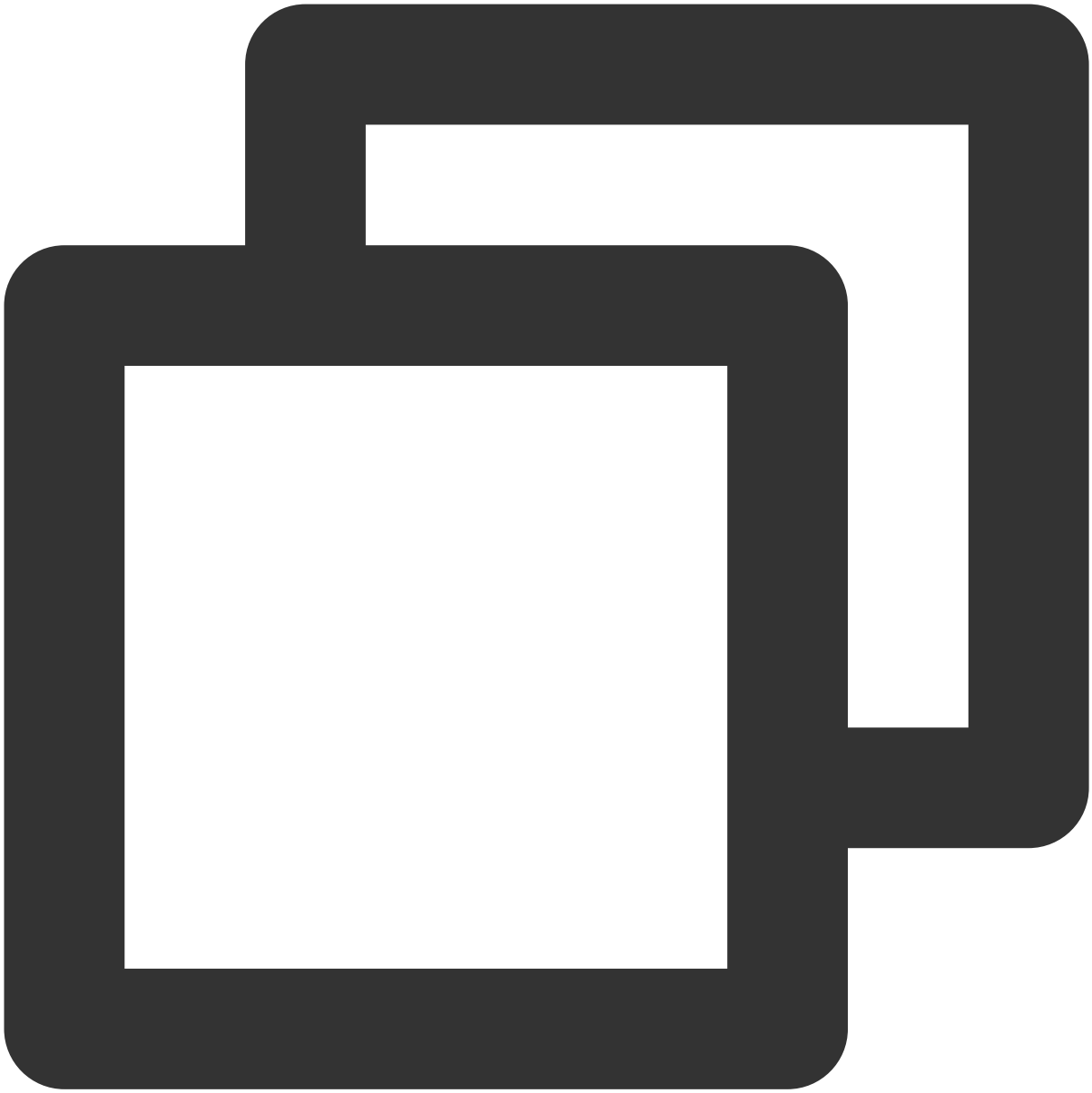
查询对象列表



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造查询对象列表的请求
    std::string bucket_name = "examplebucket-1250000000"; // 替换为用户的存储桶名，由bu
```

```
    qcloud_cos::GetBucketReq req(bucket_name);
    qcloud_cos::GetBucketResp resp;
    qcloud_cos::CosResult result = cos.GetBucket(req, &resp);

    std::vector<qcloud_cos::Content> cotents = resp.GetContents();
    for (std::vector<qcloud_cos::Content>::const_iterator itr = cotents.begin(); it
        const qcloud_cos::Content& content = *itr;
        std::cout << "key name=" << content.m_key << ", lastmodified ="
            << content.m_last_modified << ", size=" << content.m_size << std::endl;
    }

    // 3. 处理调用结果
    if (result.IsSucc()) {
        // 查询对象列表成功
    } else {
        // 查询对象列表失败，可以调用 CosResult 的成员函数输出错误信息，例如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

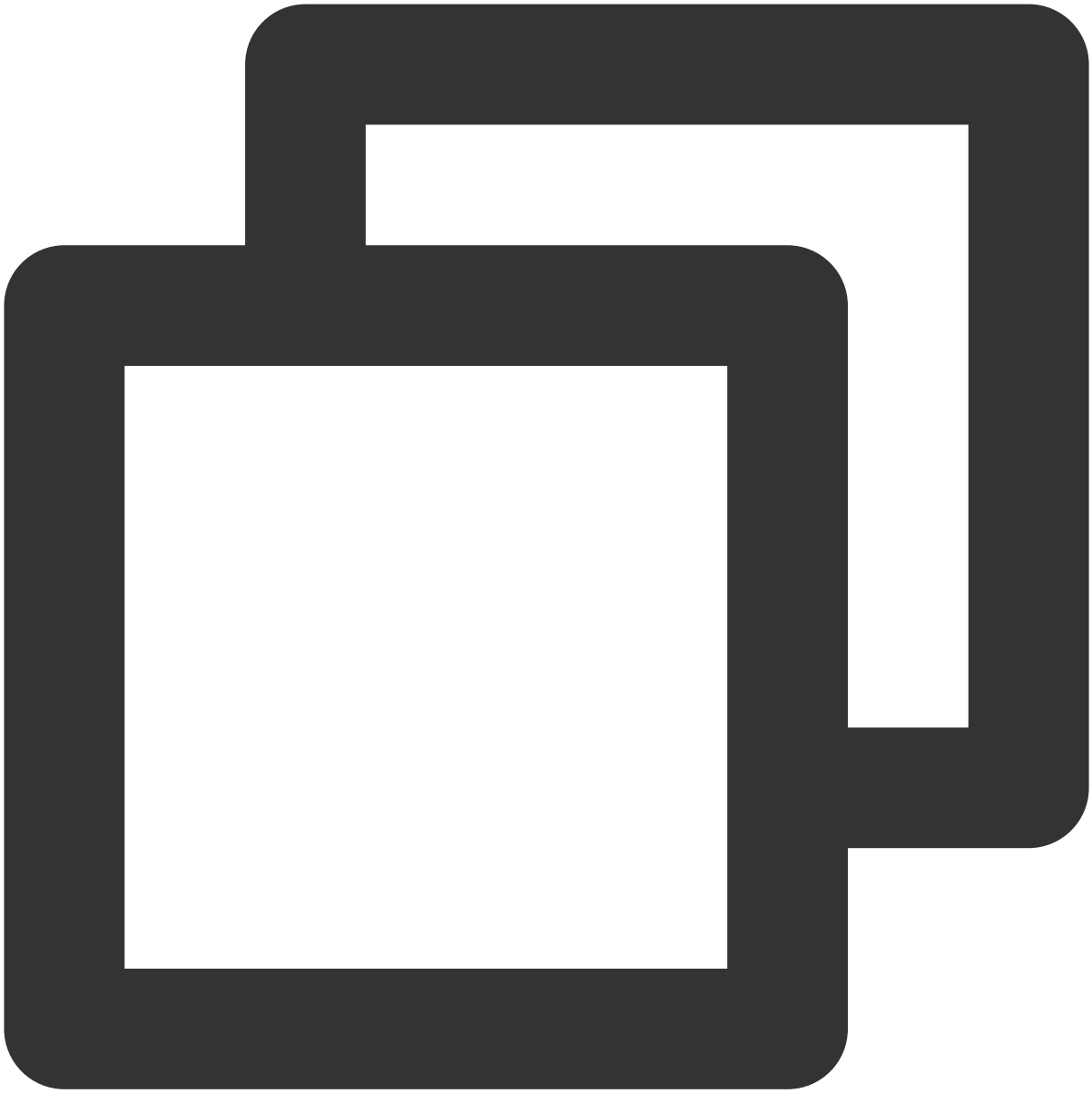
下载对象



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造下载对象的请求
    std::string bucket_name = "examplebucket-1250000000"; // 替换为用户的存储桶名，由bu
```

```
    std::string object_name = "exampleobject"; // exampleobject 即为对象键（Key），是对
    std::string local_path = "/tmp/exampleobject";
    // request 需要提供 appid、bucketname、object,以及本地的路径（包含文件名）
    qcloud_cos::GetObjectByFileReq req(bucket_name, object_name, local_path);
    qcloud_cos::GetObjectByFileResp resp;

    // 3. 调用下载对象接口
    qcloud_cos::CosResult result = cos.GetObject(req, &resp);

    // 4. 处理调用结果
    if (result.IsSucc()) {
        // 下载文件成功
    } else {
        // 下载文件失败，可以调用 CosResult 的成员函数输出错误信息，例如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

删除对象



```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
    // 1. 指定配置文件路径，初始化 CosConfig
    qcloud_cos::CosConfig config("./config.json");
    qcloud_cos::CosAPI cos(config);

    // 2. 构造删除对象的请求
    std::string bucket_name = "examplebucket-1250000000"; // 替换为用户的存储桶名，由bu
```

```
    std::string object_name = "exampleobject"; // exampleobject 即为对象键（Key），是对
    // 3. 调用删除对象接口
    qcloud_cos::DeleteObjectReq req(bucket_name, object_name);
    qcloud_cos::DeleteObjectResp resp;
    qcloud_cos::CosResult result = cos.DeleteObject(req, &resp);

    // 4. 处理调用结果
    if (result.IsSucc()) {
        // 删除对象成功
    } else {
        // 删除对象失败，可以调用 CosResult 的成员函数输出错误信息，例如 requestID 等
        std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
        std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
        std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
        std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
        std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
        std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
    }
}
```

图片处理
图片持久化处理

最近更新时间：2024-03-13 15:17:11

简介

本文档提供关于图片持久化处理的 API 概览以及 SDK 示例代码。

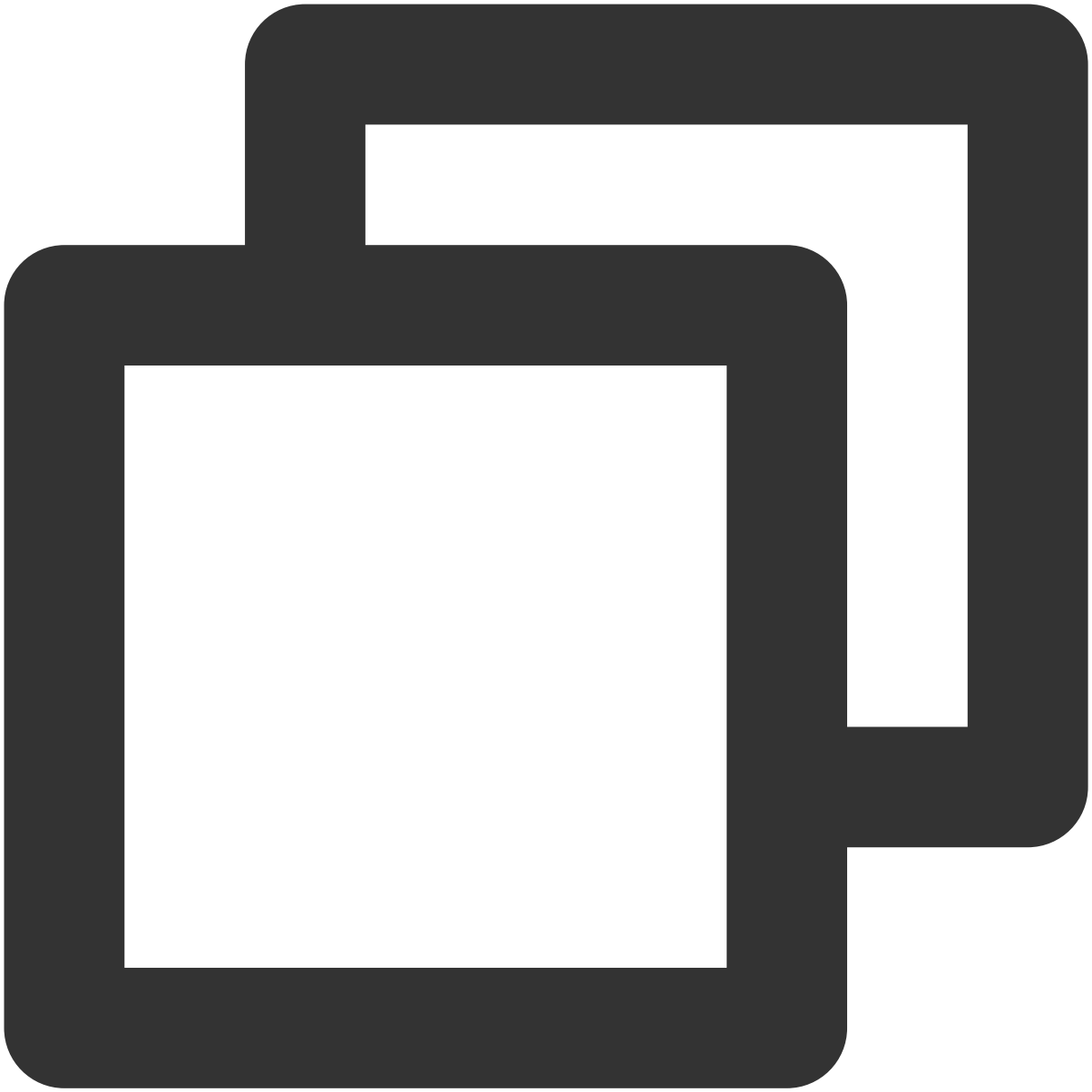
API	说明
图片持久化处理	用于图片持久化处理

上传时处理

功能说明

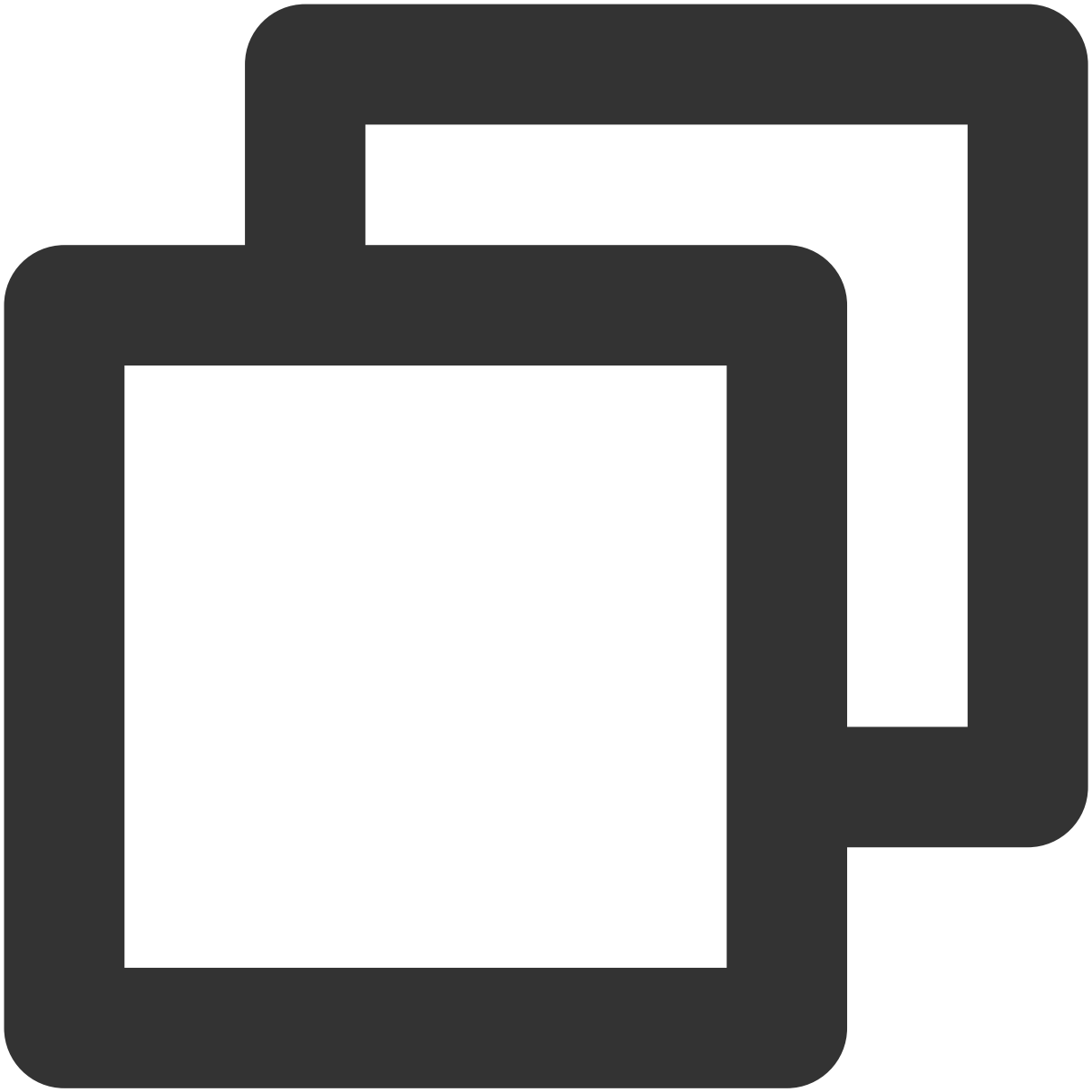
在图片上传时实现相应的图片处理，并可将原图和处理结果存入到 COS。目前支持20M以内文件处理 。

方法原型



```
CosResult PutImage(const PutImageByFileReq& request, PutImageByFileResp* response);
```

请求示例



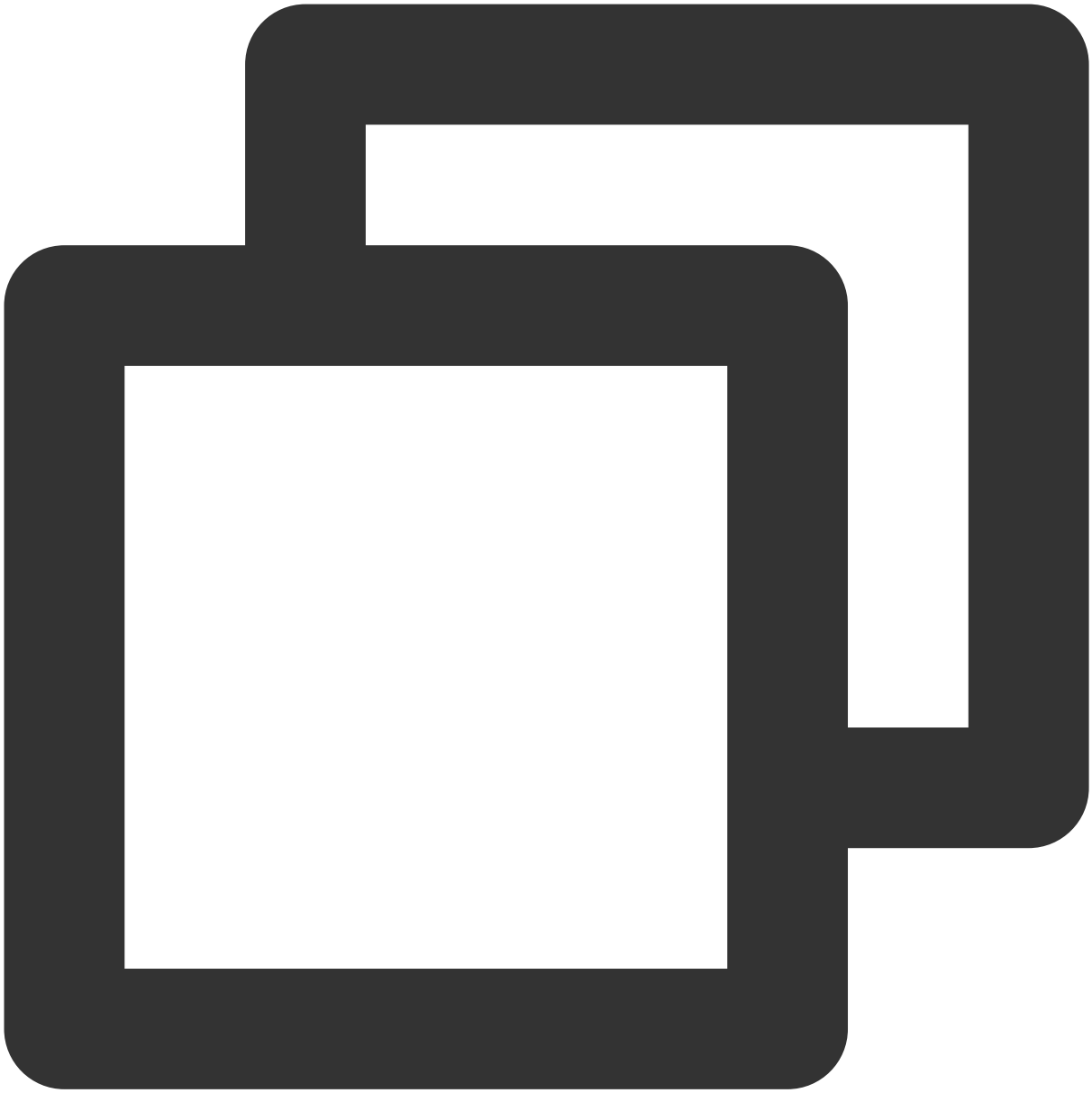
```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";

PutImageByFileReq req(bucket_name, object_name, local_file);
PutImageByFileResp resp;

PicOperation pic_operation;

PicRules rule;
// 处理结果的文件路径名称，如以/开头
```

```
rule.fileid = "/" + object_name + "_sharpen";
// 锐化处理参数
rule.rule = "imageMogr2/sharpen/70";
pic_operation.AddRule(rule);

rule.fileid = "/" + object_name + "_rotate";
// 旋转处理参数
rule.rule = "imageMogr2/rotate/90";
pic_operation.AddRule(rule);

CosResult result = cos.PutImage(req, &resp);
if (result.IsSucc()) {
   // 调用成功，调用 resp 的成员函数获取返回内容
   std::cout << "ProcessResult: " << resp.GetUploadResult().to_string() << std::end
} else {
   // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	PutImage 操作的请求	PutImageByFileReq	是
resp	PutImage 操作的响应	PutImageByFileResp	是

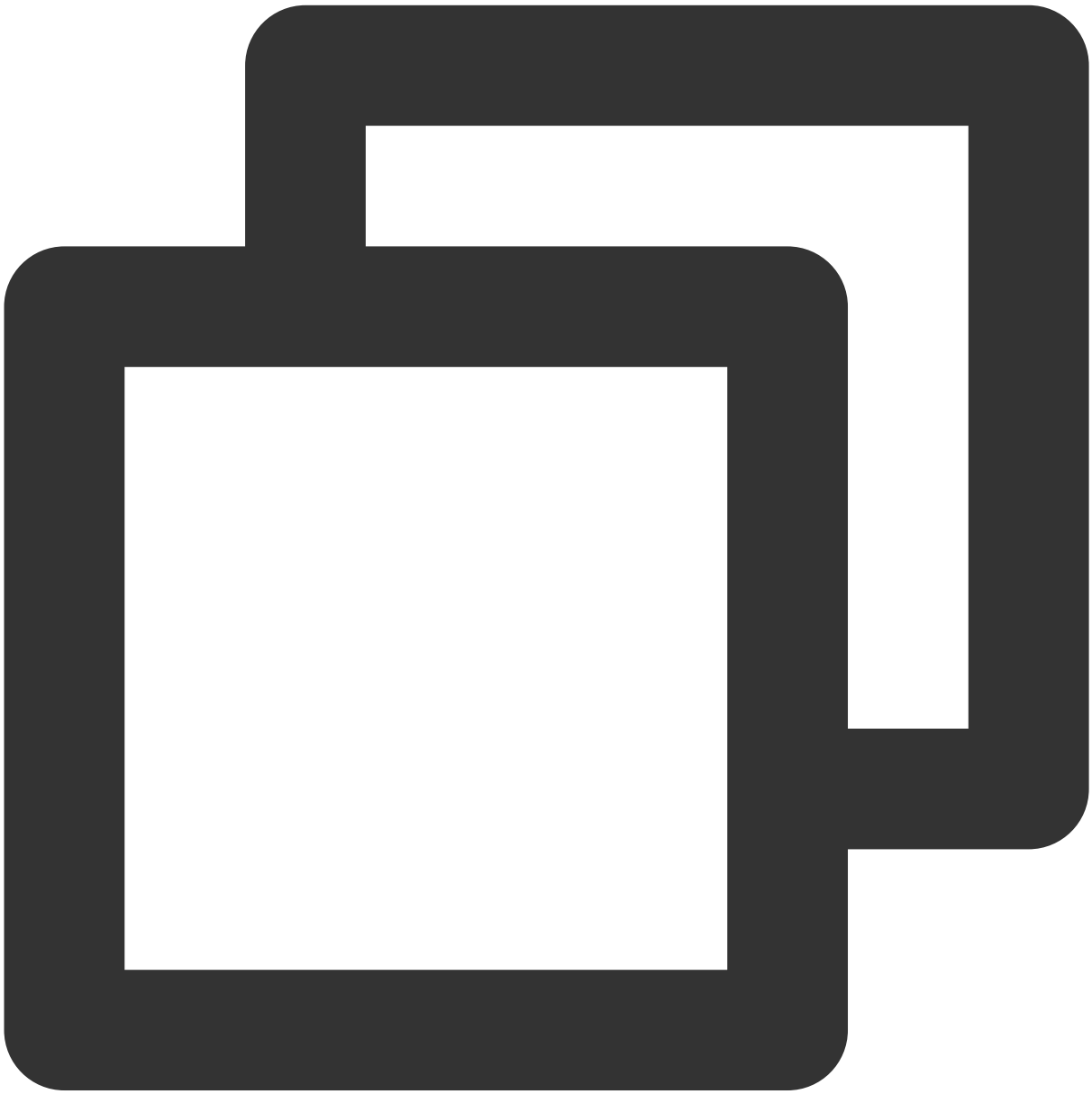
req 涉及到如下结构:



```
struct PicRules {
  std::string bucket;  // 存储结果的目标存储桶名称，格式为 BucketName-APPID，如果不指定的话
  std::string fileid;  // 处理结果的文件路径名称，如以/开头，则存入指定文件夹中，否则，存入原图
  std::string rule;  // 处理参数，参见对象存储图片处理API。若按指定样式处理，则以style/开头，
};

class PicOperation {
 public:
  PicOperation() : is_pic_info(true) {}
  virtual ~PicOperation() {}
  void AddRule(const PicRules& rule) { rules.push_back(rule); }
```

```
  void TurnOffPicInfo() { is_pic_info = false; }
 private:
  boolis_pic_info;  // 是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
  std::vector<PicRules> rules;  //处理规则，一条规则对应一个处理结果（目前支持五条规则），不
};
```

resp 涉及到如下结构：



```
struct CodeLocation {
  std::vector<std::string> points;  // 二维码坐标点
};
```

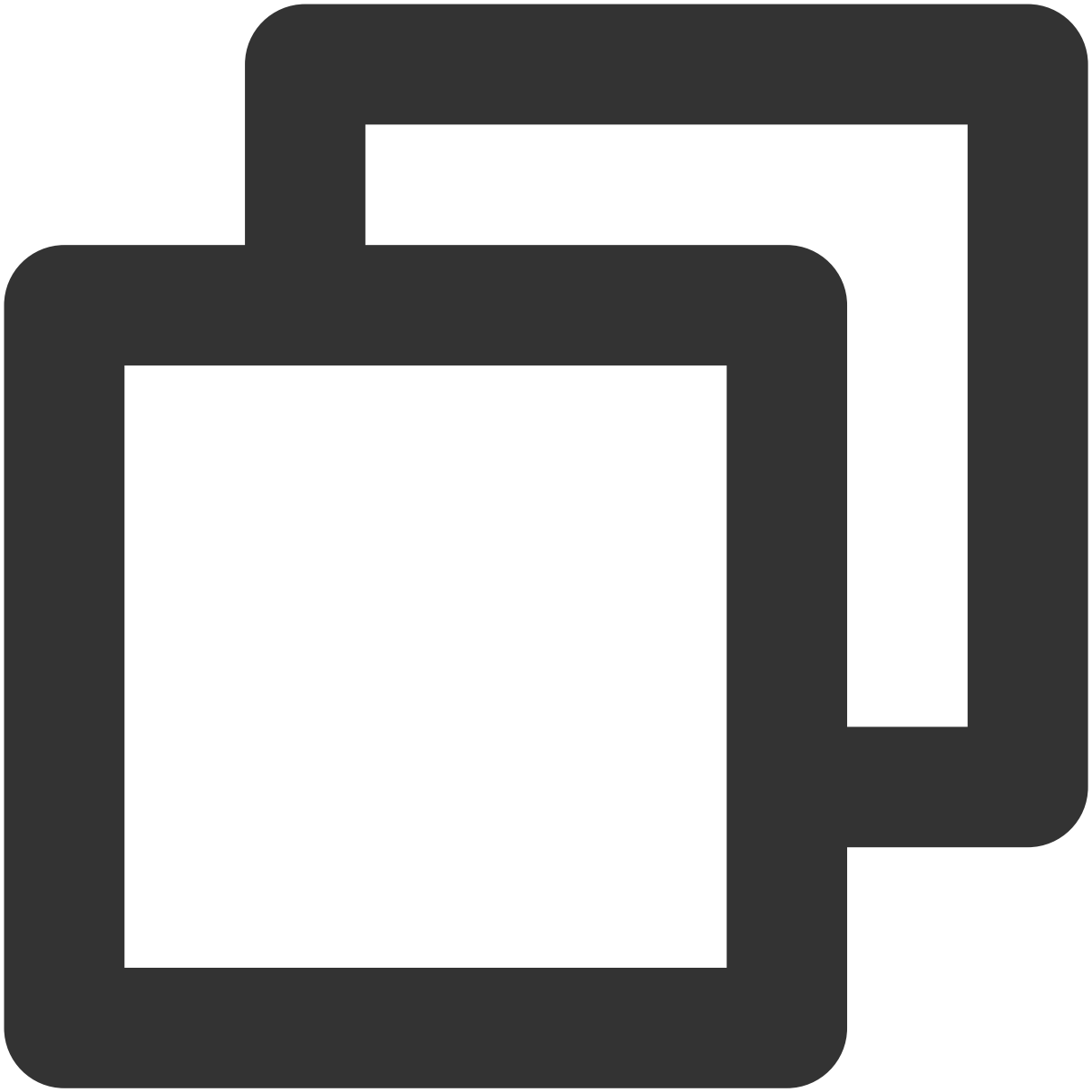
```
struct QRcodeInfo {
  std::string code_url;         // 二维码的内容。可能识别不出
  CodeLocation code_location;   // 图中识别到的二维码位置坐标

};

struct Object {
  std::string key;        // 文件名
  std::string location;   // 图片路径
  std::string format;     // 图片格式
  int width;              // 图片宽度
  int height;             // 图片高度
  int size;               // 图片大小
  int quality;            // 图片质量
  std::string etag;       // 处理结果图 ETag 信息
  int code_status;  // 二维码识别结果。0表示未识别到二维码，1表示识别到二维码
  int watermark_status;   // 当 type
                          // 为2时返回该字段，表示提取到全盲水印的可信度。
                          //   具体为0-100的数字，75分以上表示确定有盲水印，60-75表示疑似有盲
  std::vector<QRcodeInfo> qr_code_info;   // 二维码识别结果，可能有多个
};

struct ProcessResults {
  std::vector<Object> objects;   // 可能有多个对象
};

struct ImageInfo {
  std::string format;   // 格式
  int width;            // 图片宽度
  int height;           // 图片高度
  int quality;          // 图片质量
  std::string ave;      // 图片主色调
  int orientation;      // 图片旋转角度
};

struct UploadResult {
  OriginalInfo original_info;   // 原图信息
  ProcessResults process_result;   // 图片处理结果
};
```

云上图片处理

功能说明

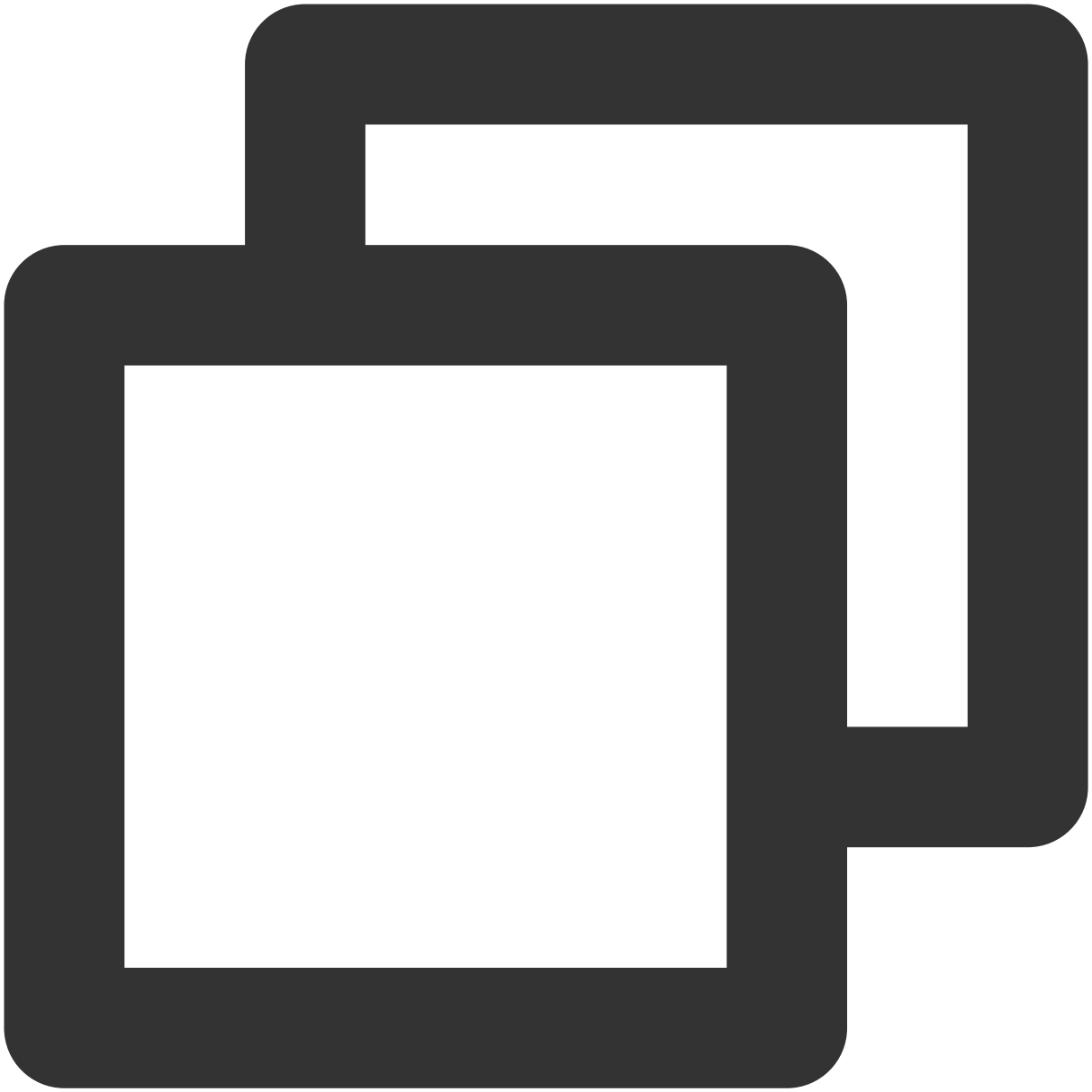
能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

方法原型



```
CosResult CloudImageProcess(const CloudImageProcessReq& request, CloudImageProcessR
```

请求示例



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";

CloudImageProcessReq req(bucket_name, object_name);
CloudImageProcessResp resp;

PicOperation pic_operation;
PicRules rule;
// 处理结果的文件路径名称，如以/开头
rule.fileid = "/" + object_name + "_thumbnail";
```

```
// 缩放处理参数
rule.rule = "imageMogr2/thumbnail/!30p";
pic_operation.AddRule(rule);

rule.fileid = "/" + object_name + "_cut";
// 裁剪处理参数
rule.rule = "imageMogr2/cut/300x300";
pic_operation.AddRule(rule);
req.SetPicOperation(pic_operation);

CosResult result = cos.CloudImageProcess(req, &resp);
if (result.IsSucc()) {
    // 调用成功，调用 resp 的成员函数获取返回内容
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	PutImage操作的请求	PutImageByFileReq	是
resp	PutImage操作的响应	PutImageByFileResp	是

基础图片处理

最近更新时间：2024-03-13 15:17:16

简介

腾讯云对象存储 COS 集成了 数据万象（Cloud Infinite，CI）专业的一体化多媒体解决方案，涵盖以下图片处理功能，详情可见 图片处理概述。

服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图
	样式设置	设置图片的样式，方便管理不同需求的图片

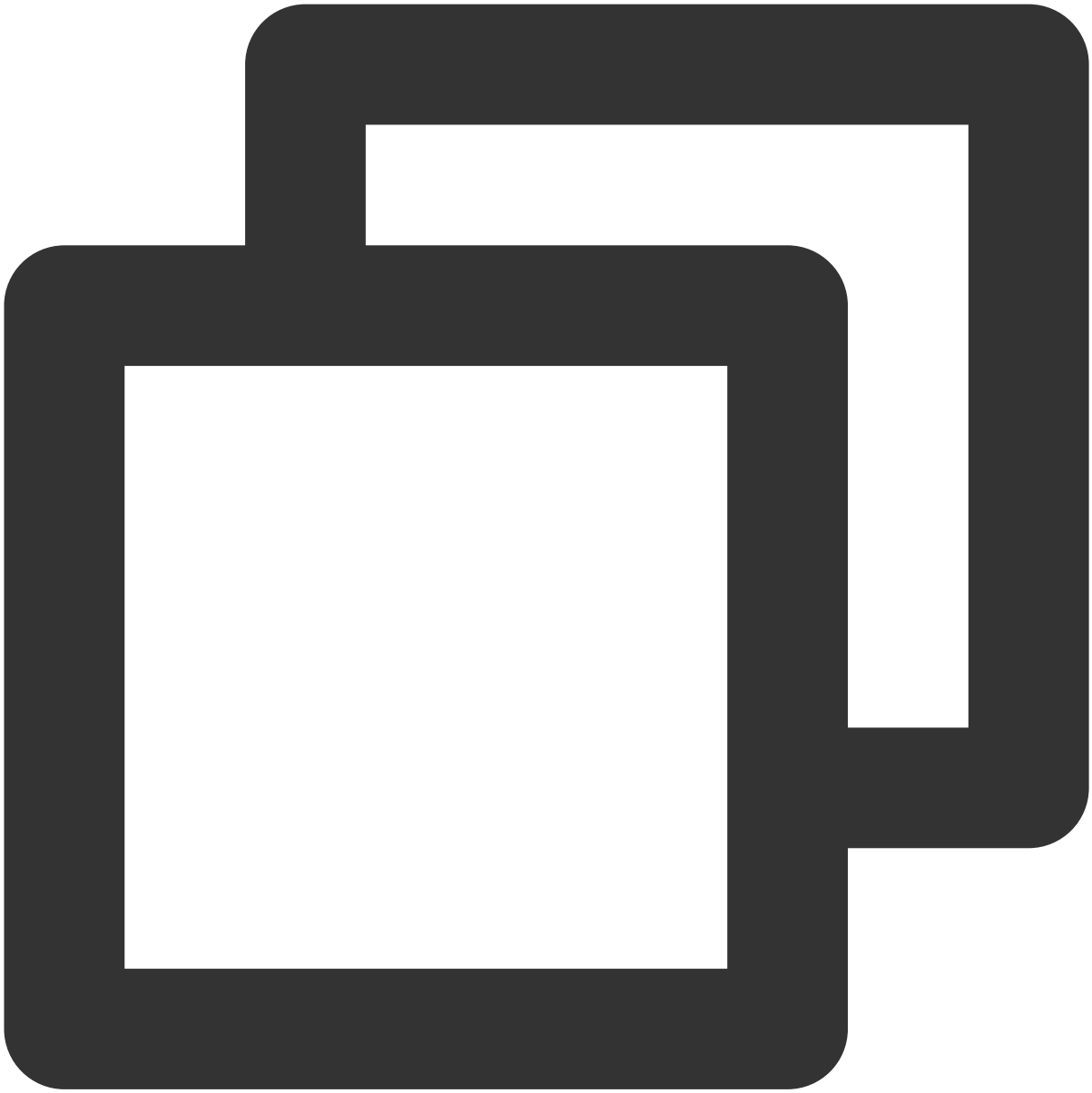
基础图片处理

缩放

功能说明

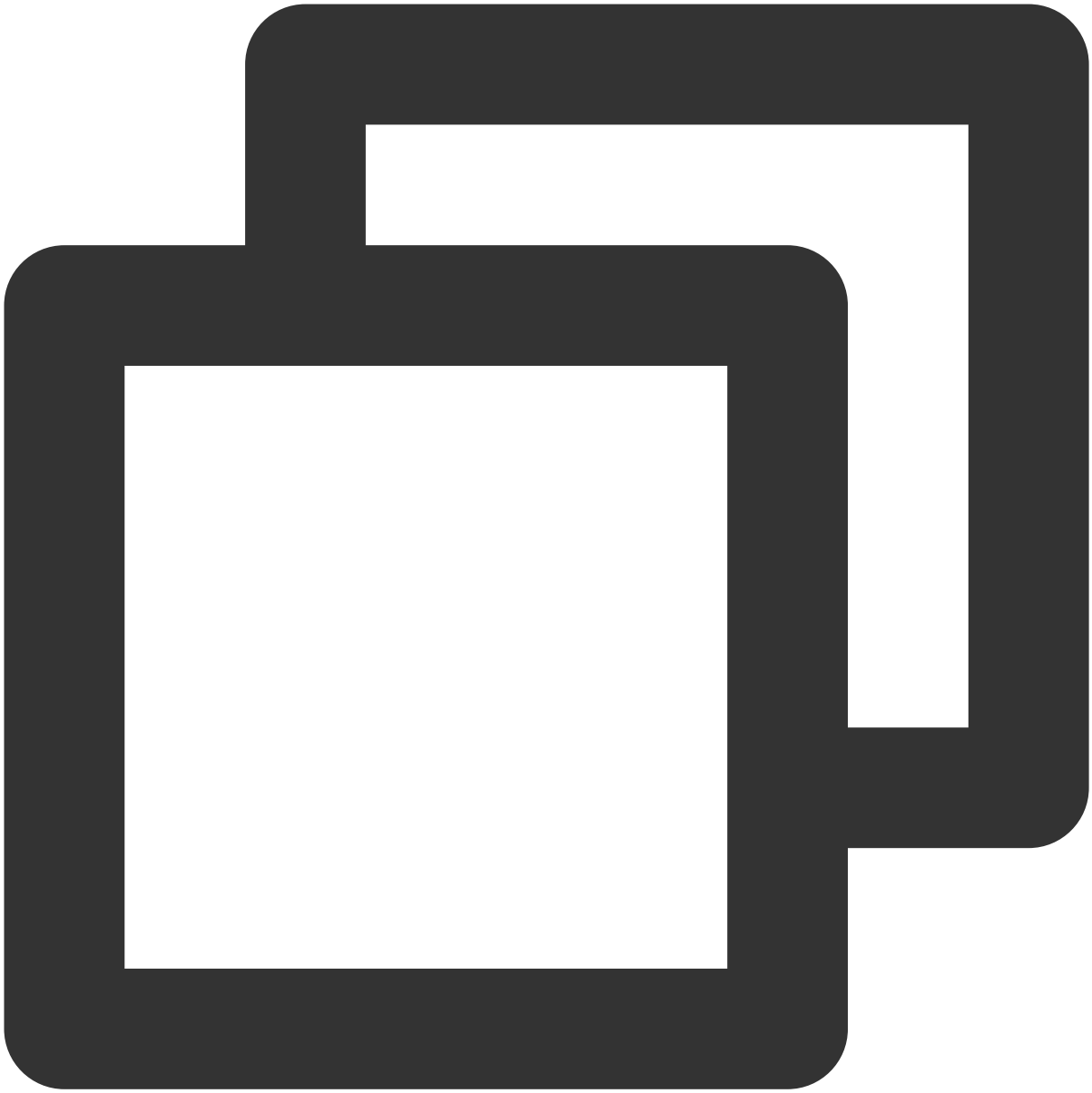
等比缩放、设定目标宽高缩放等多种方式 。

方法原型



```
CosResult GetObject(const GetObjectByFileReq& request, GetObjectByFileResp* respons
```

请求示例



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";

qcloud_cos::GetObjectByFileReq req(bucket_name, object_name, local_file);
// 指定图片的宽高为原图的50%
req.AddParam("imageMogr2/thumbnail/!50p", "");
qcloud_cos::GetObjectByFileResp resp;
qcloud_cos::CosResult result = cos.GetObject(req, &resp);
if (result.IsSucc()) {
   // 调用成功，调用 resp 的成员函数获取返回内容
```

```
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	GetObject操作的请求	GetObjectByFileReq	是
resp	GetObject操作的响应	GetObjectByFileResp	是

裁剪

功能说明

跟缩放操作方法类似。

盲水印

最近更新时间：2024-03-13 15:17:16

简介

本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。

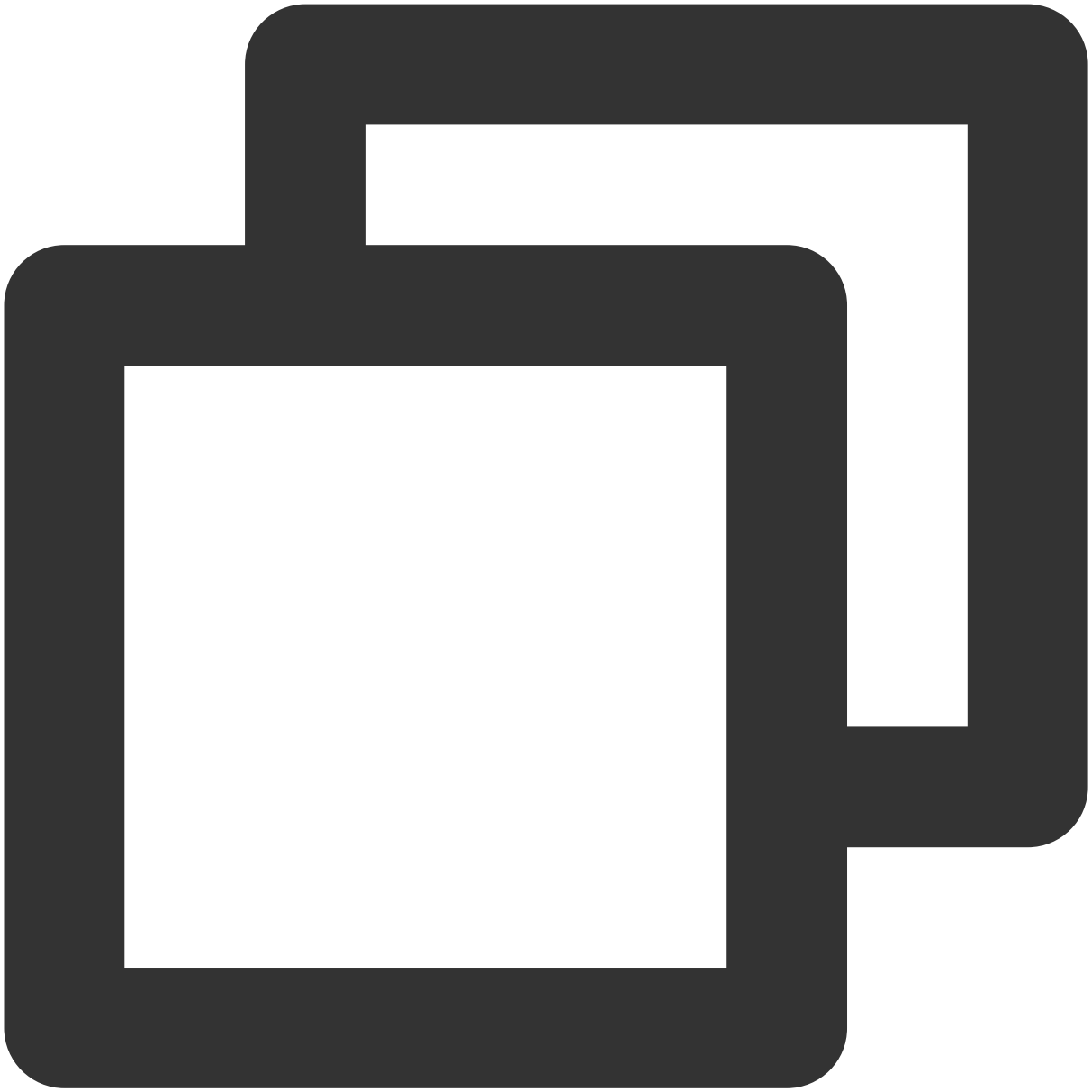
API	操作描述
盲水印	对本地图片添加或提取盲水印并上传至存储桶

上传时添加盲水印

功能说明

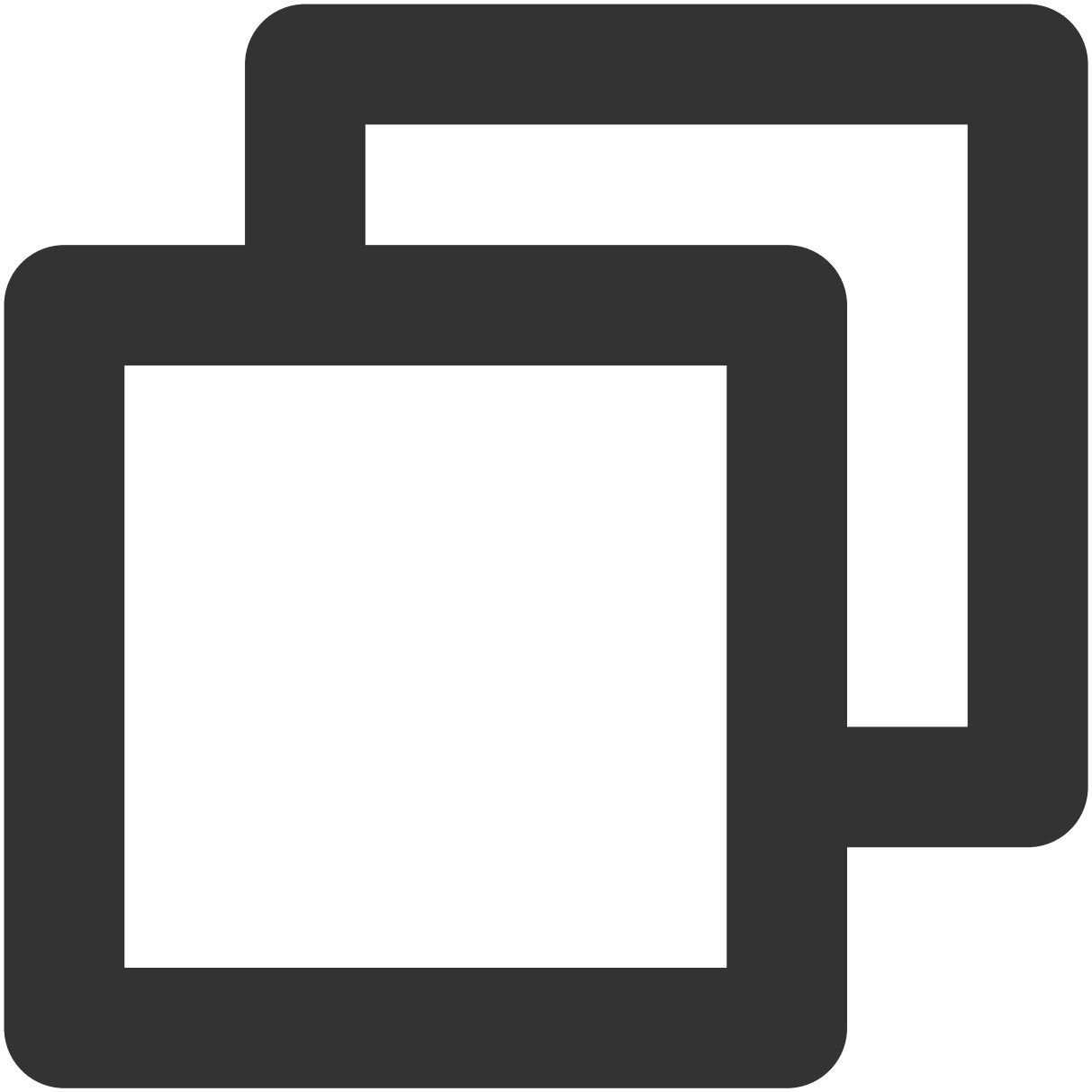
在上传图片时添加盲水印。

方法原型



```
CosResult PutImage(const PutImageByFileReq& request, PutImageByFileResp* response);
```

请求示例



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";

PutImageByFileReq req(bucket_name, object_name, local_file);
PutImageByFileResp resp;

PicOperation pic_operation;
PicRules rule;
// 存储文件路径
rule.fileid = "/" + object_name + "_watermark";
```

```
// 盲水印图片地址，需要经过 URL 安全的 Base64 编码
std::string image = "http://" + bucket_name + ".cos.ap-guangzhou.myqcloud.com/" + o
// 盲水印处理参数
rule.rule = "watermark/3/type/1/image/" + CodecUtil::Base64Encode(image);
pic_operation.AddRule(rule);

CosResult result = cos.PutImage(req, &resp);
if (result.IsSucc()) {
    // 调用成功，调用 resp 的成员函数获取返回内容
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

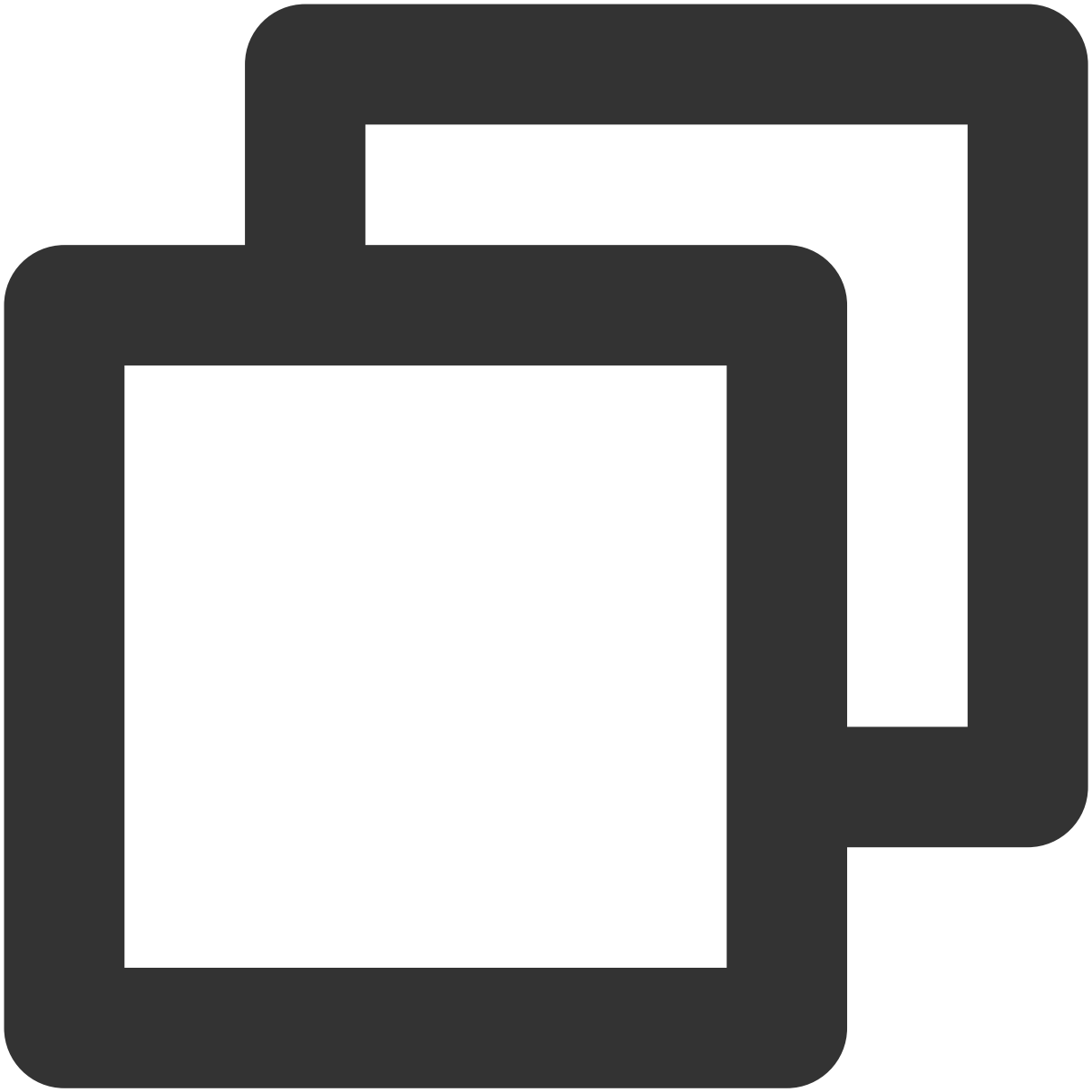
参数	参数描述	类型	是否必填
req	PutImage操作的请求	PutImageByFileReq	是
resp	PutImage操作的响应	PutImageByFileResp	是

下载时添加盲水印

功能说明

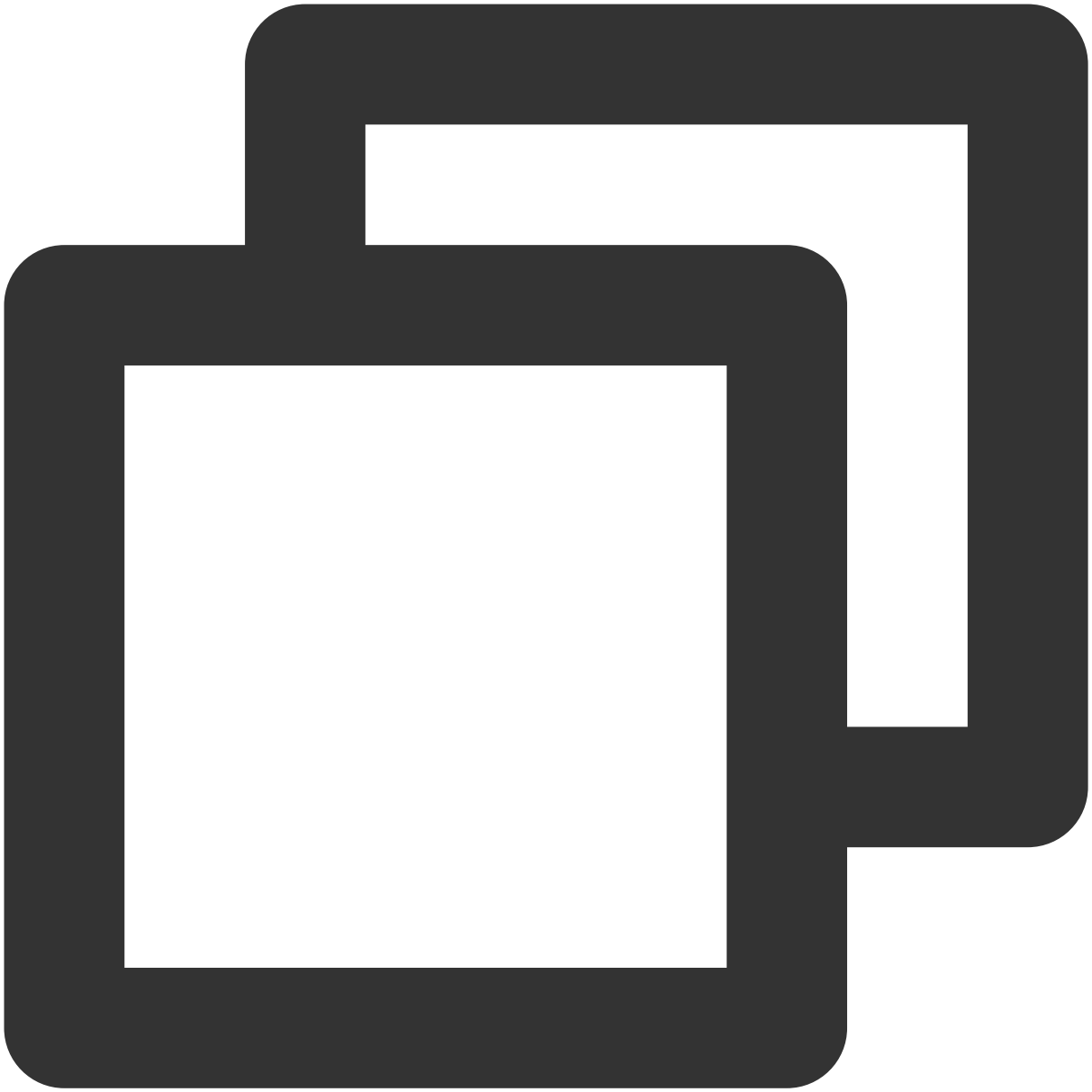
在图片下载时添加盲水印。

方法原型



```
CosResult GetObject(const GetObjectByFileReq& request, GetObjectByFileResp* respons
```

请求示例



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";

GetObjectByFileReq req(bucket_name, object_name, local_file);
GetObjectByFileResp resp;

// 盲水印文本
std::string watermark_text = "test";
// 盲水印参数，需要经过 URL 安全的 Base64 编码
std::string watermark_param = "watermark/3/type/3/text/" + CodecUtil::Base64Encode(
```

```
req.AddParam(watermark_param, "");

CosResult result = cos.GetObject(req, &resp);
if (result.IsSucc()) {
   // 调用成功，调用 resp 的成员函数获取返回内容
} else {
   // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	PutImage操作的请求	PutImageByFileReq	是
resp	PutImage操作的响应	PutImageByFileResp	是

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 15:17:11

简介

本文档提供关于媒体 bucket 的 API 概览和 SDK 示例代码。

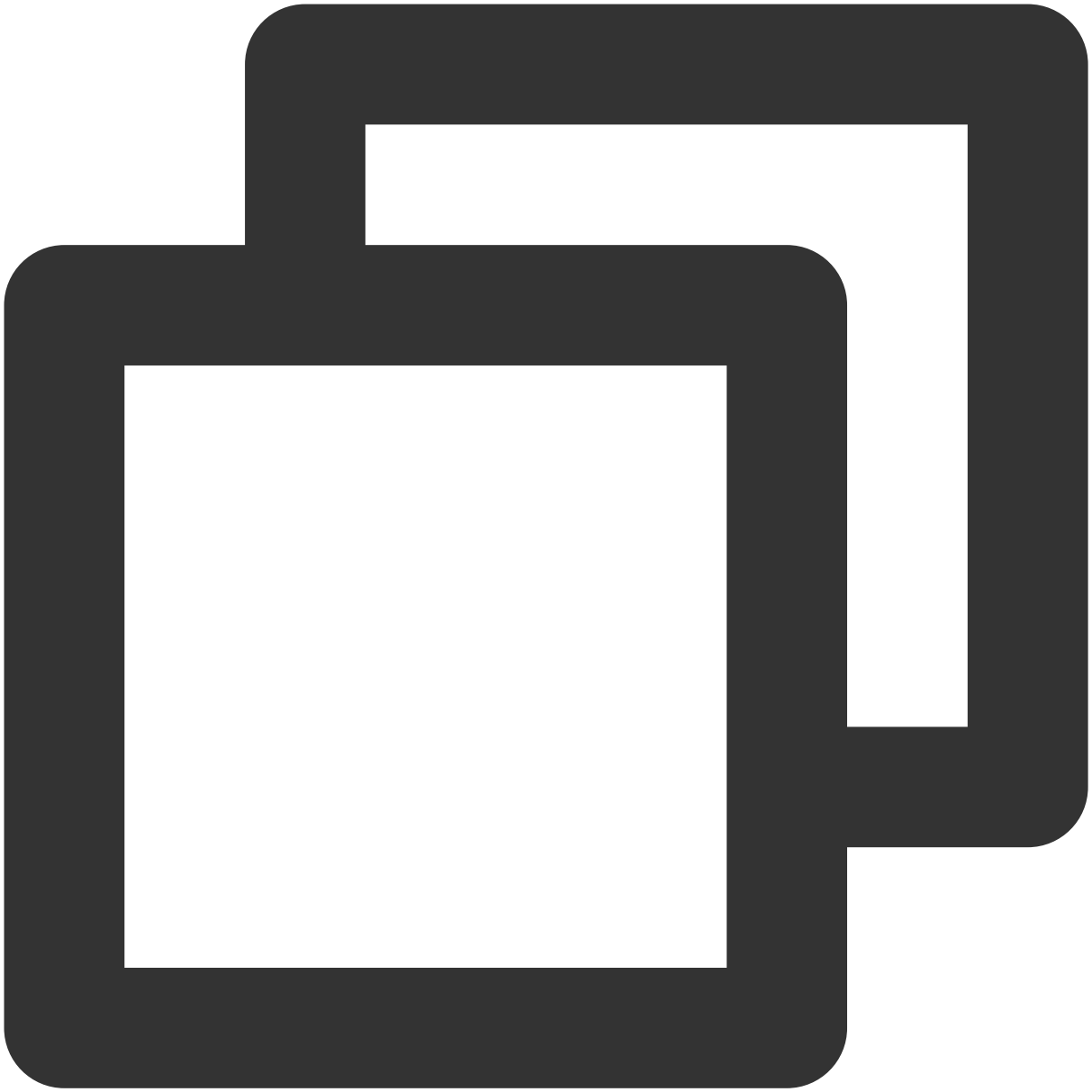
API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

功能说明

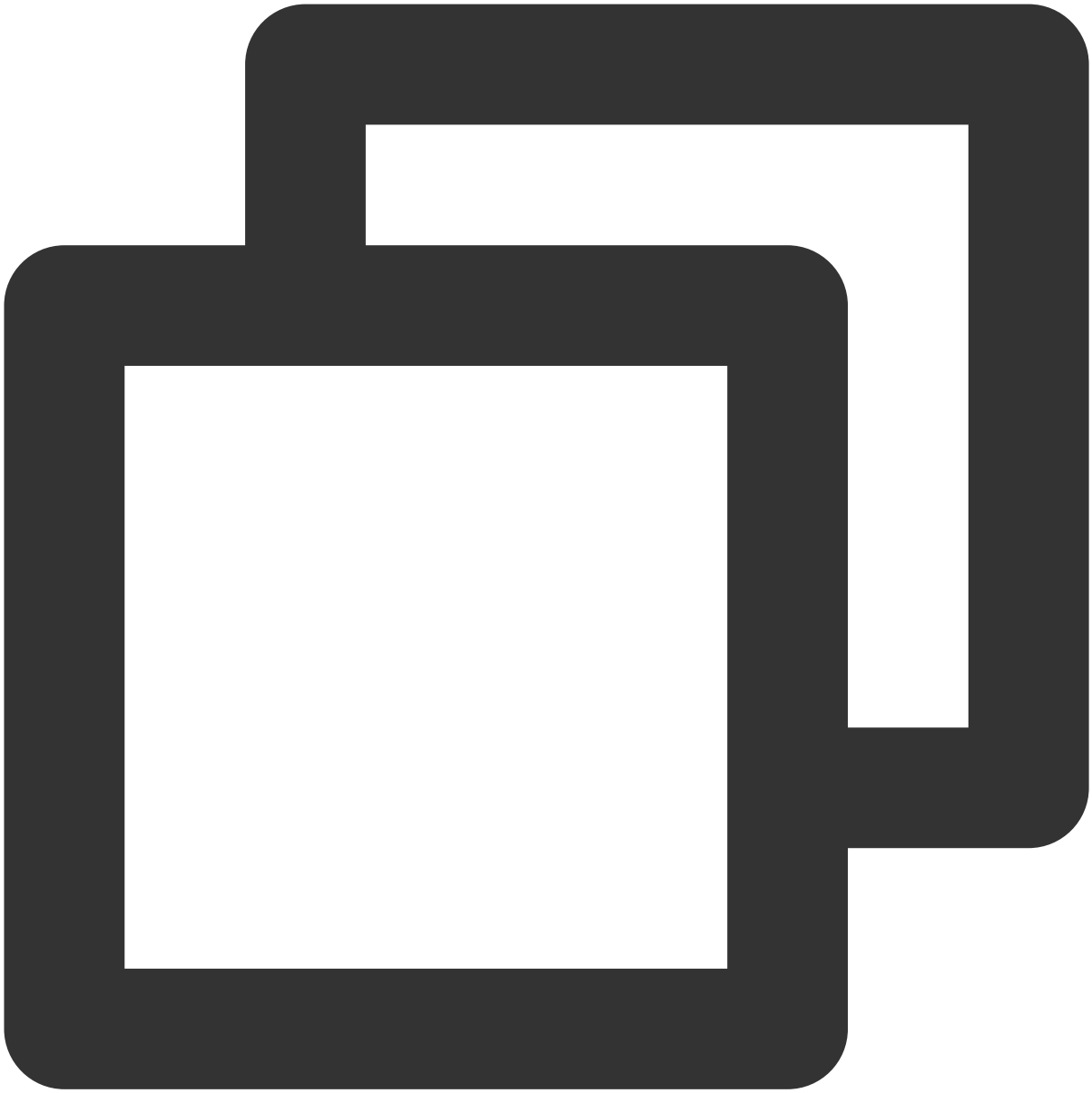
用于查询已经开通媒体处理功能的存储桶。

方法原型



```
CosResult DescribeMediaBuckets(const DescribeMediaBucketsReq& request, DescribeMedi
```

示例代码



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
DescribeMediaBucketsReq req;
DescribeMediaBucketsResp resp;
req.SetRegions("ap-guangzhou");
// 设置存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
// req.SetBucketNames("xxx");
// 设置存储桶名称前缀，前缀搜索
// req.SetBucketName("xxx");
// 设置第几页
// req.SetPageNumber();
```

```
// 设置每页个数
// req.SetPageSize();
CosResult result = cos.DescribeMediaBuckets(req, &resp);
qcloud_cos::CosResult result = cos.GetObject(req, &resp);
if (result.IsSucc()) {
    // 调用成功，调用 resp 的成员函数获取返回内容
    std::cout << "Result: " << resp.GetResult().to_string() << std::endl;
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数名称	参数描述	类型	是否必填
request	操作的请求	DescribeMediaBucketsReq	是
response	操作的响应	DescribeMediaBucketsResp	是

媒体信息接口

最近更新时间：2024-03-13 15:17:11

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

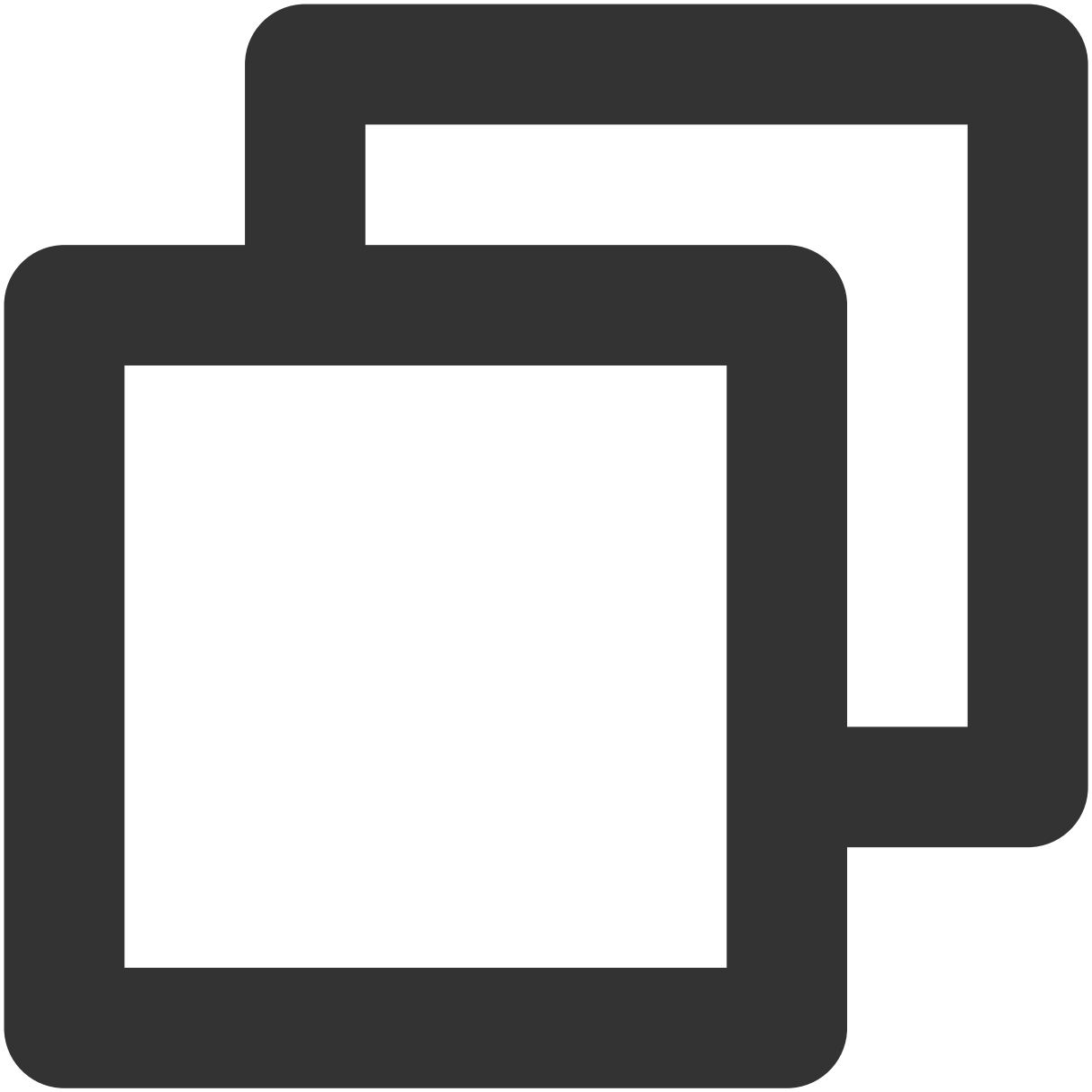
API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

功能说明

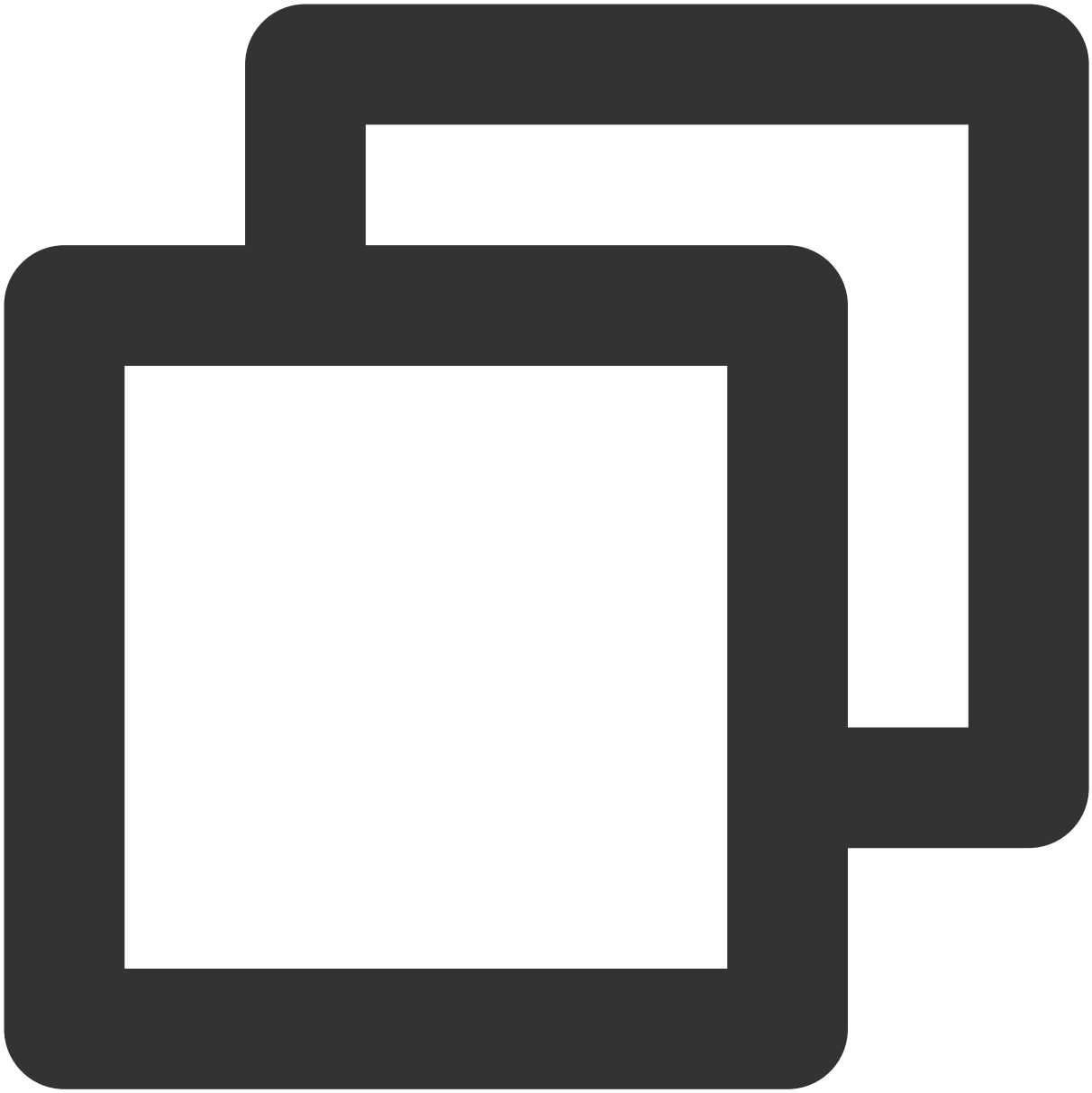
用于获取媒体文件的信息。

方法原型



```
CosResult GetMediaInfo(const GetMediaInfoReq& request, GetMediaInfoResp* response);
```

示例代码



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
GetMediaInfoReq req(bucket_name, object_name);
GetMediaInfoResp resp;
CosResult result = cos.GetMediaInfo(req, &resp);
if (result.IsSucc()) {
    // 调用成功，调用 resp 的成员函数获取返回内容
    std::cout << "Result: " << resp.GetResult().to_string() << std::endl;
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数名称	参数描述	类型	是否必填
request	操作的请求	GetMediaInfoReq	是
response	操作的响应	GetMediaInfoResp	是

媒体截图接口

最近更新时间：2024-03-13 15:17:16

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

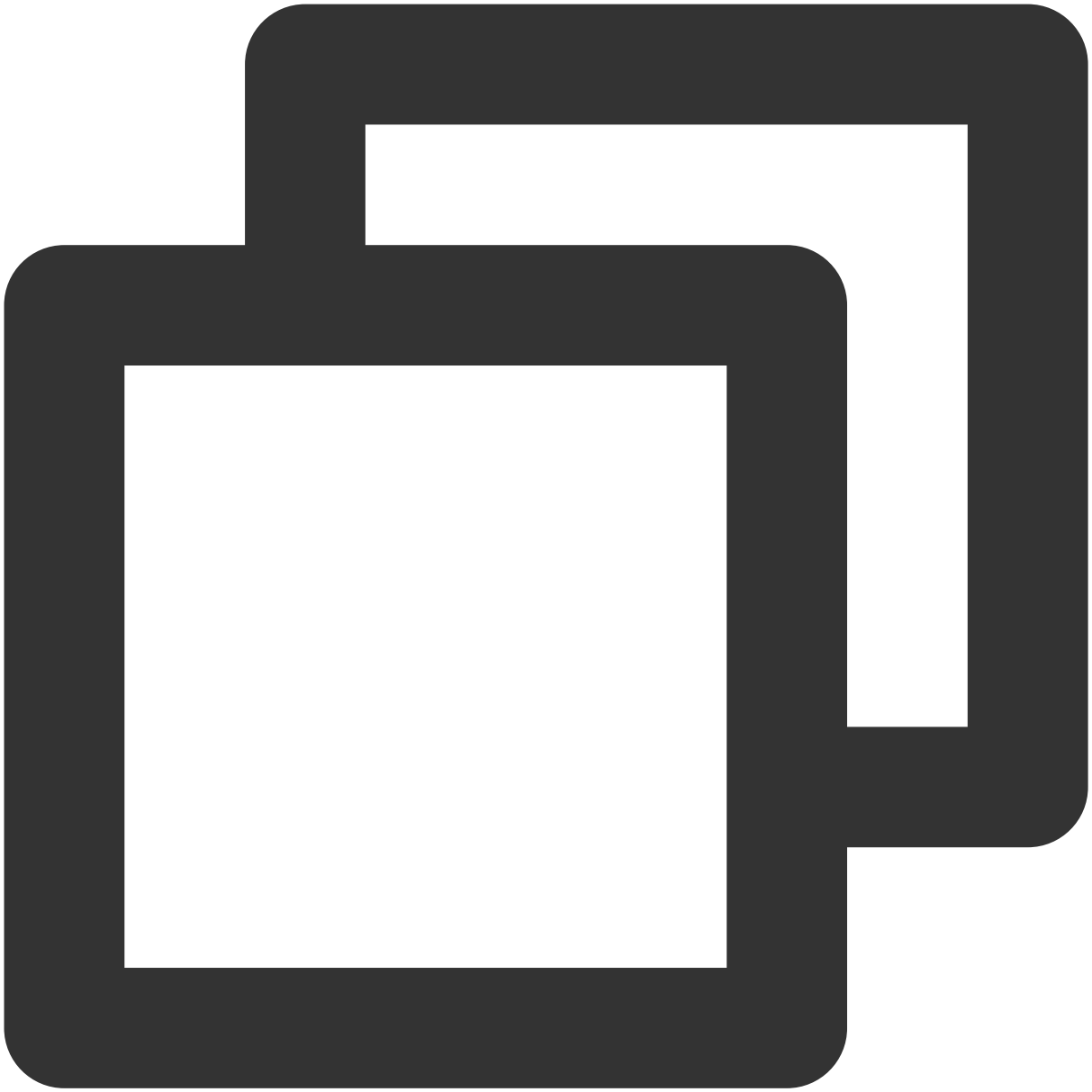
API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

功能说明

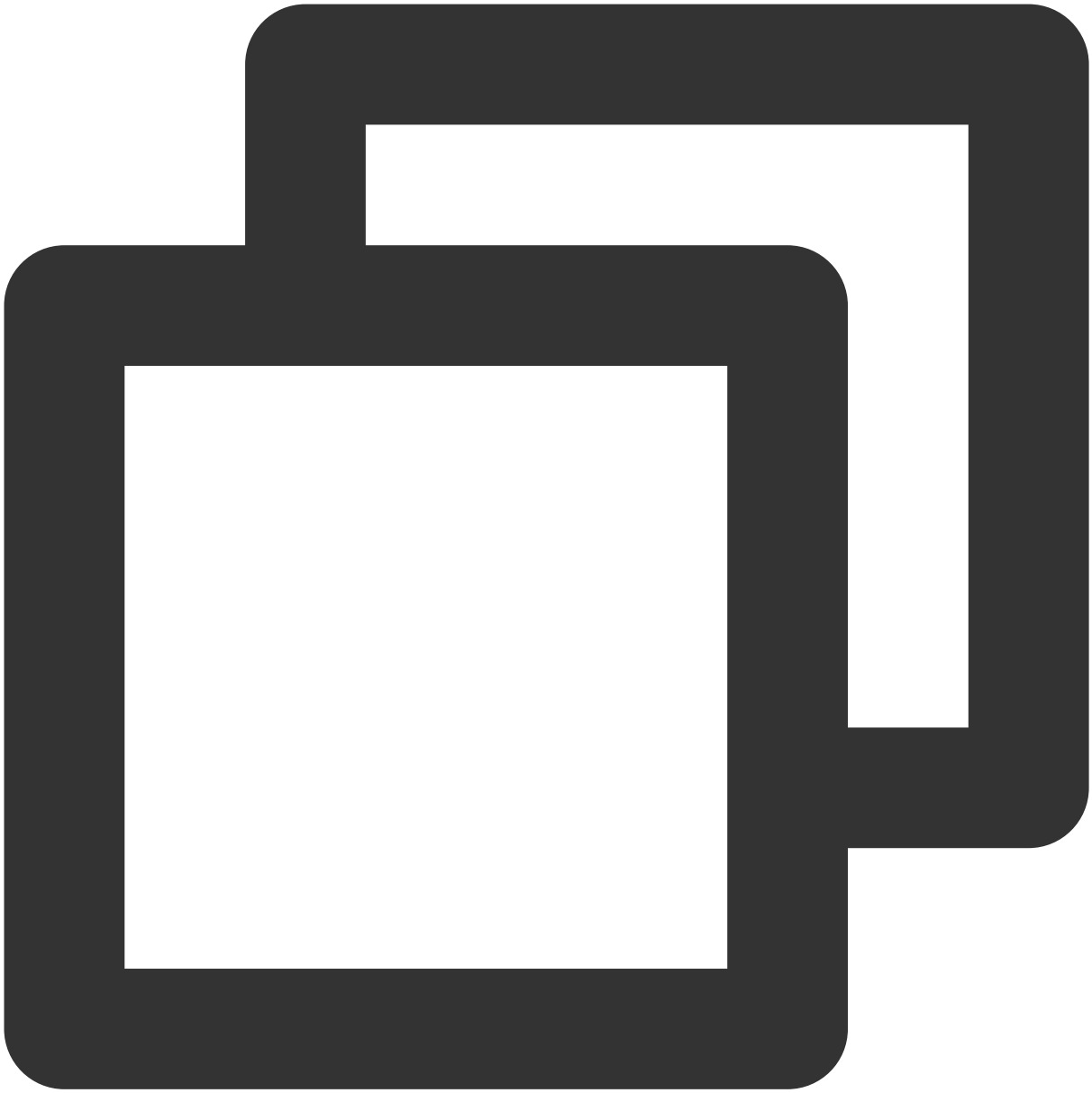
用于获取媒体文件某个时间的截图，截图信息存到本地文件。

方法原型



```
CosResult GetSnapshot(const GetSnapshotReq& request, GetSnapshotResp* response);
```

示例代码



```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
GetSnapshotReq req(bucket_name, object_name, local_file);
GetSnapshotResp resp;
req.SetTime(100);
// 设置截图的宽
// req.SetWitdh(xxx);
// 设置截图的高
// req.SetHeight(xxx);
// 设置格式
// req.SetFormat(xxx);
```

```
CosResult result = cos.GetSnapshot(req, &resp);
if (result.IsSucc()) {
    // 调用成功，调用 resp 的成员函数获取返回内容
} else {
    // 调用失败，调用 result 的成员函数获取错误信息
}
```

参数说明

参数名称	参数描述	类型	是否必填
request	操作的请求	GetSnapshotReq	是
response	操作的响应	GetSnapshotResp	是

.NET(C#) SDK 快速入门

最近更新时间：2024-03-04 16:55:55

相关资源

SDK 源码下载请参见：XML .NET SDK。
SDK 快速下载地址：XML .NET SDK。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：.NET（C#）SDK 常见问题。
说明：
如果您在使用 SDK 时遇到函数或方法不存在等错误，请先将 SDK 升级到最新版再重试。

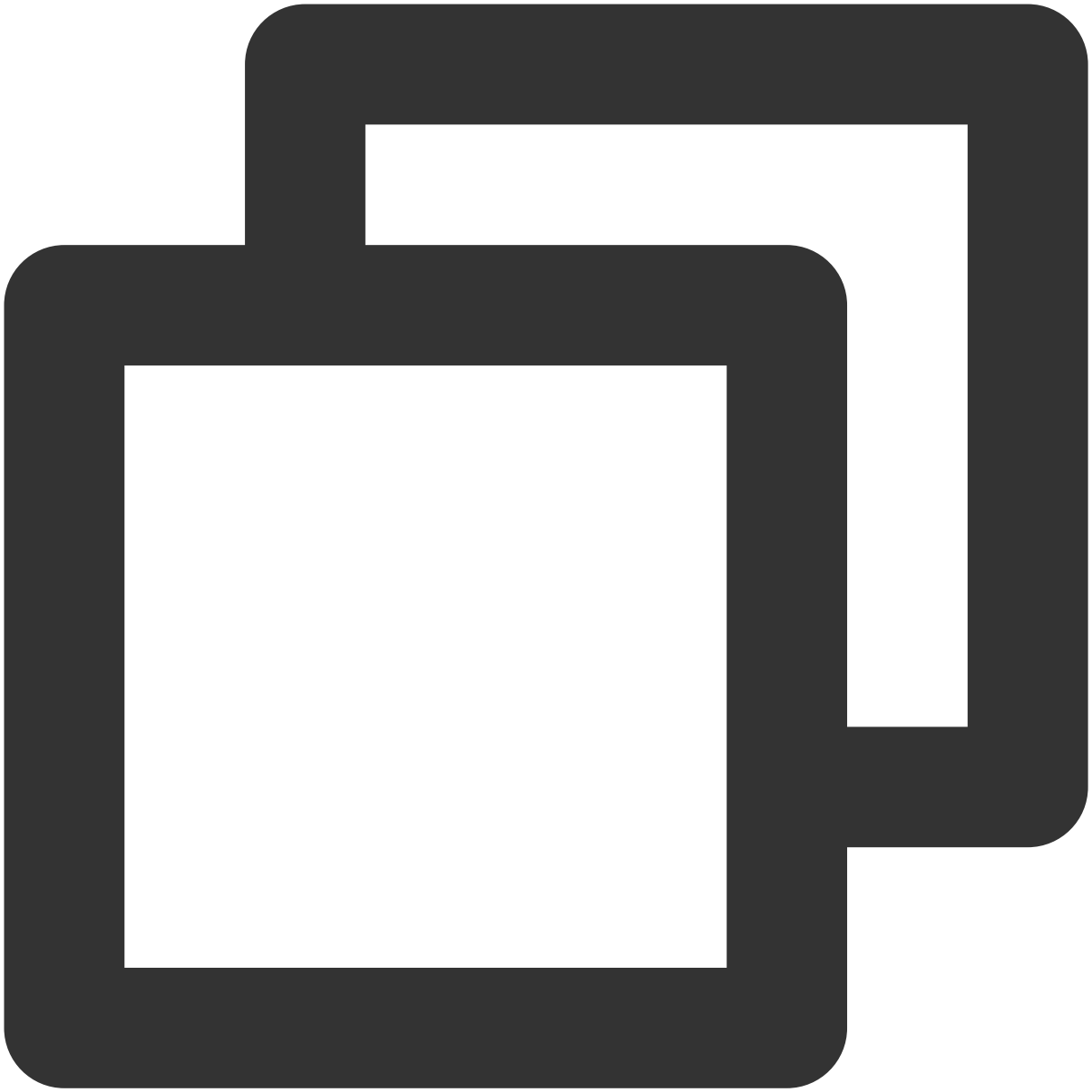
第一步：集成 SDK

环境依赖

.NET SDK 基于 .NET Standard 2.0 开发。
Windows：安装 .NET Core 2.0 及以上版本，或者 .NET Framework 2.0 及以上版本。
Linux/Mac：安装 .NET Core 2.0 及以上版本。

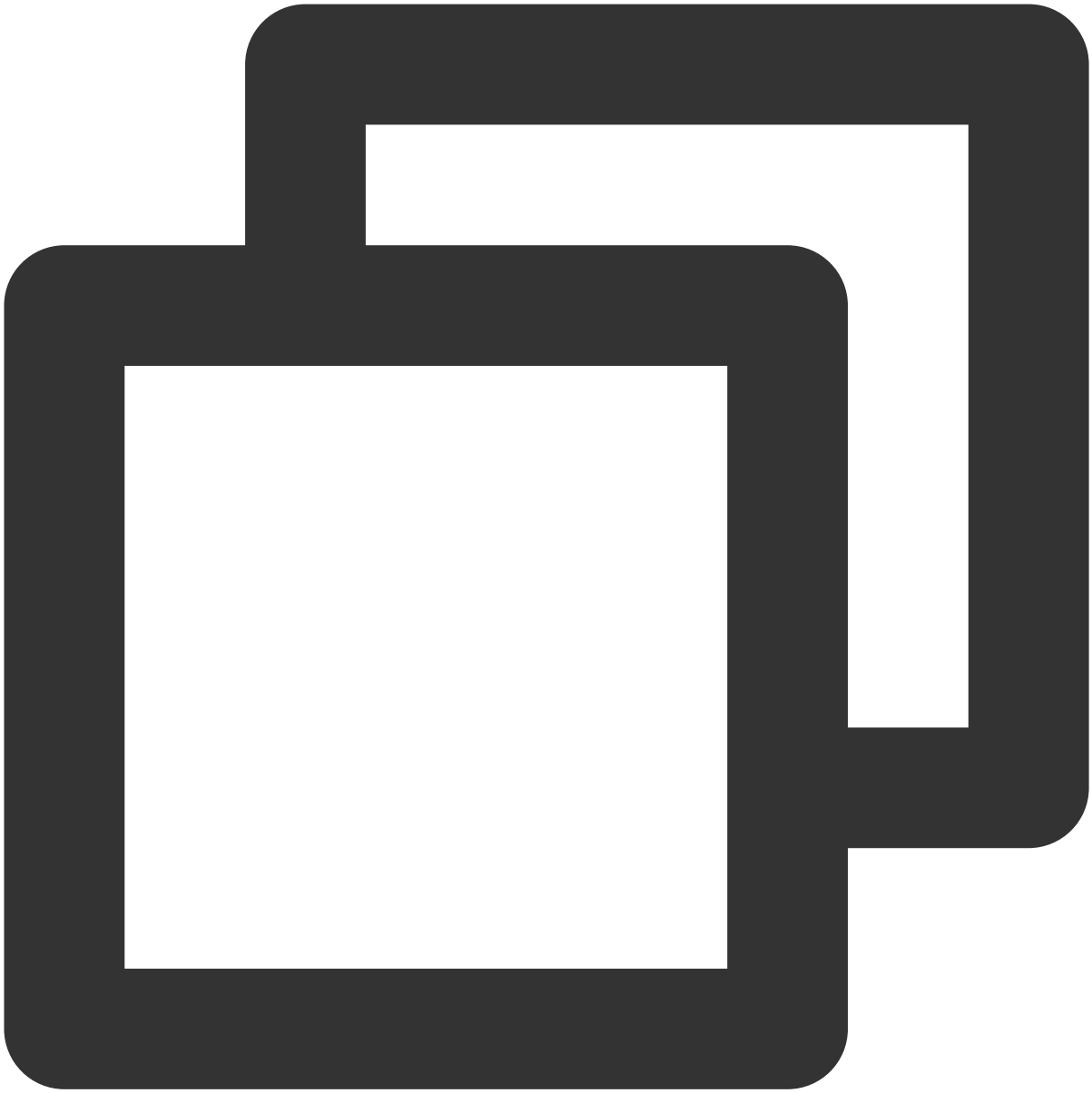
添加 SDK

我们提供 `Nuget` 的集成方式，您可以在工程的 `csproj` 文件里添加：



```
<PackageReference Include="Tencent.QCloud.Cos.Sdk" Version="5.4.*" />
```

如果您是使用 .NET CLI，请使用如下命令安装：



```
dotnet add package Tencent.QCloud.Cos.Sdk
```

如果您是用于目标框架 .Net Framework 4.0 及以下版本的开发，请下载 Releases 并使用 COSXML-Compatible.dll 文件。

在 Visual Studio 项目中选择**项目** > **添加引用** > **浏览** > **COSXML-Compatible.dll** 的方式添加 .NET(C#) SDK。

说明：

兼容包中不包含高级上传、下载等功能的支持，详情参见 向下兼容指南。

第二步：初始化 COS 服务

注意：

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

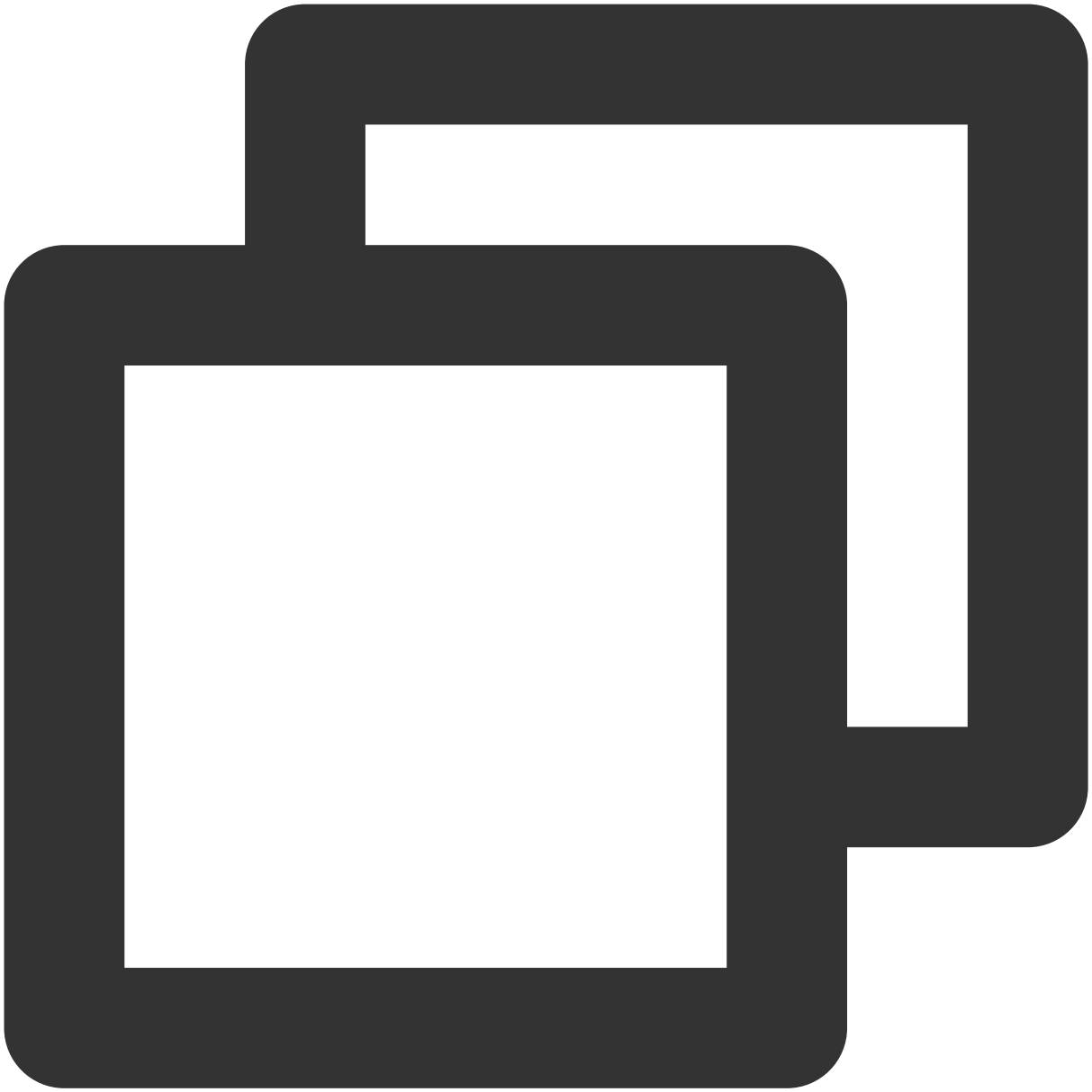
如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

下面为您介绍如何使用 COS .NET SDK 完成一个基础操作，例如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。

说明：

关于文章中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。

SDK 中常用的命名空间有：



```
using COSXML;
using COSXML.Auth;
using COSXML.Model.Object;
using COSXML.Model.Bucket;
using COSXML.CosException;
```

在执行任何和 COS 服务相关请求之前，都需要先实例化 `CosXmlConfig` ，`QCloudCredentialProvider` ，`CosXmlServer` 3个对象。其中：

`CosXmlConfig` 提供配置 SDK 接口。

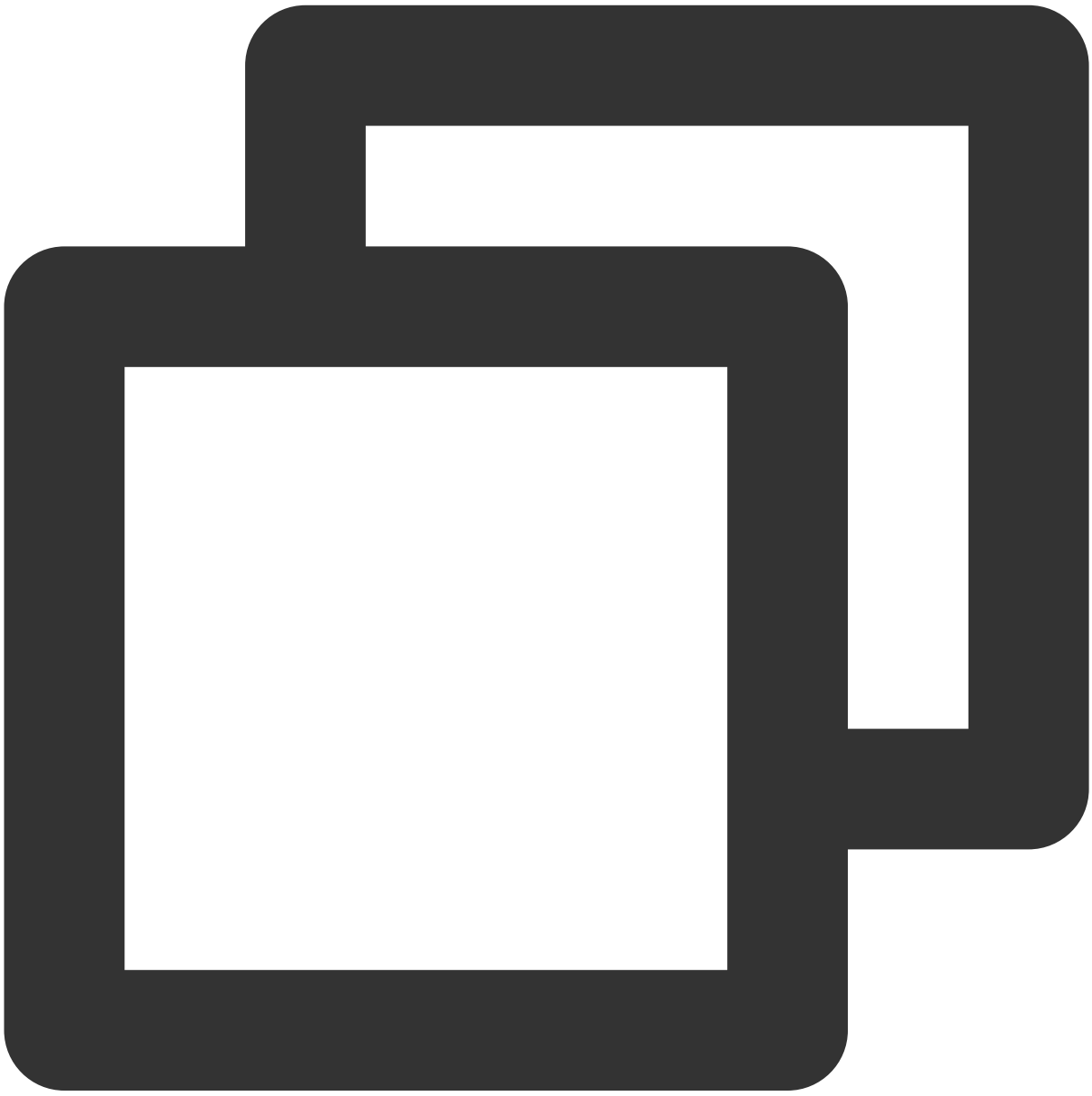
`QCloudCredentialProvider` 提供设置密钥信息接口。

`CosXmlServer` 提供各种 COS API 服务接口。

说明：

下述初始化示例中使用的临时密钥，其生成和使用可参见 临时密钥生成及使用指引。

1. 初始化服务设置



```
//初始化 CosXmlConfig
string appid = "1250000000";//设置腾讯云账户的账户标识 APPID
string region = "COS_REGION"; //设置一个默认的存储桶地域
CosXmlConfig config = new CosXmlConfig.Builder()
```

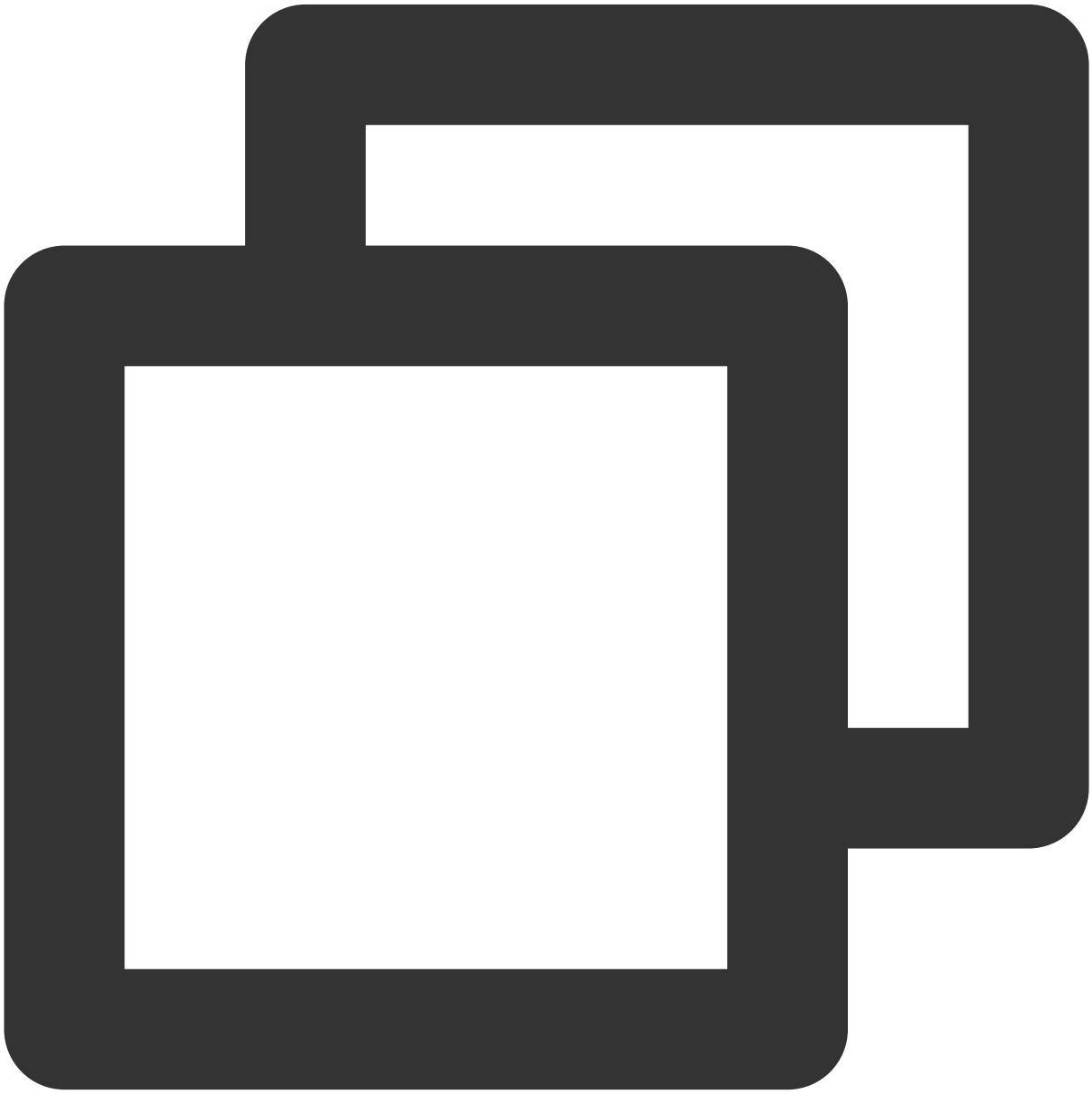
```
    .IsHttps(true)  //设置默认 HTTPS 请求
    .SetRegion(region)  //设置一个默认的存储桶地域
    .SetDebugLog(true)  //显示日志
    .Build();  //创建 CosXmlConfig 对象
```

2. 提供访问凭证

SDK 中提供了3种方式：持续更新的临时密钥、不变的临时密钥、永久密钥。

方式1：持续更新的临时密钥

因为临时密钥在一定时效后过期，以下方式保证可以在过期后自动获取到新的临时密钥。

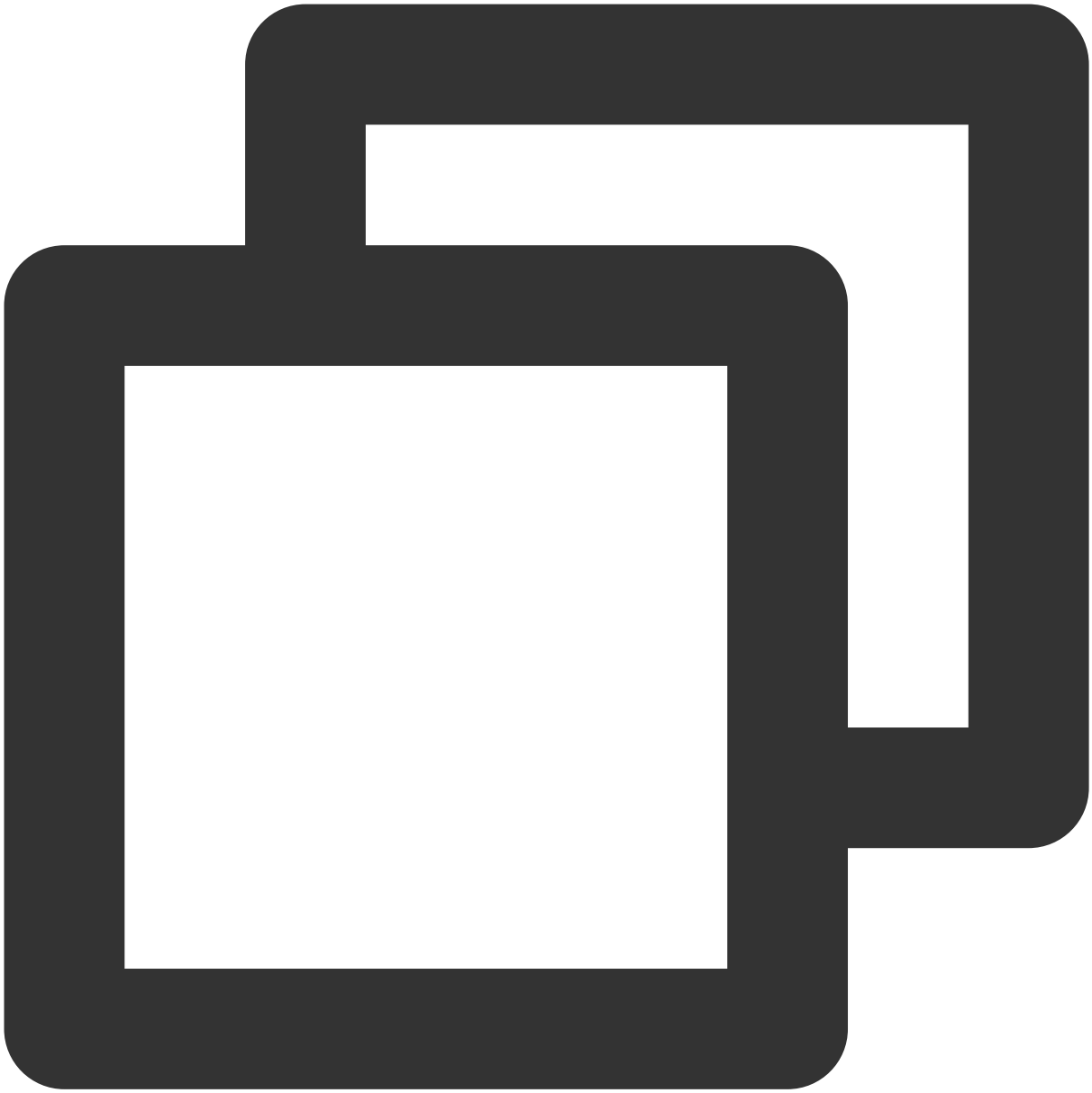


```
public class CustomQCloudCredentialProvider : DefaultSessionQCloudCredentialProvide
{

  // 这里假设开始没有密钥，也可以用初始的临时密钥来初始化
  public CustomQCloudCredentialProvider(): base(null, null, 0L, null) {
    ;
  }

  public override void Refresh()
  {
    //... 首先通过腾讯云请求临时密钥
    string tmpSecretId = "SECRET_ID"; //"临时密钥 SecretId", 临时密钥生成和使用指引参见 h
    string tmpSecretKey = "SECRET_KEY"; //"临时密钥 SecretKey", 临时密钥生成和使用指引参
    string tmpToken = "COS_TOKEN"; //"临时密钥 token", 临时密钥生成和使用指引参见 https:/
    long tmpStartTime = 1546860702;//临时密钥有效开始时间，精确到秒
    long tmpExpiredTime = 1546862502;//临时密钥有效截止时间，精确到秒
    // 调用接口更新密钥
    SetQCloudCredential(tmpSecretId, tmpSecretKey,
      String.Format("{0};{1}", tmpStartTime, tmpExpiredTime), tmpToken);
  }
}

QCloudCredentialProvider cosCredentialProvider = new CustomQCloudCredentialProvider
```

方式2：不变的临时密钥（不建议）

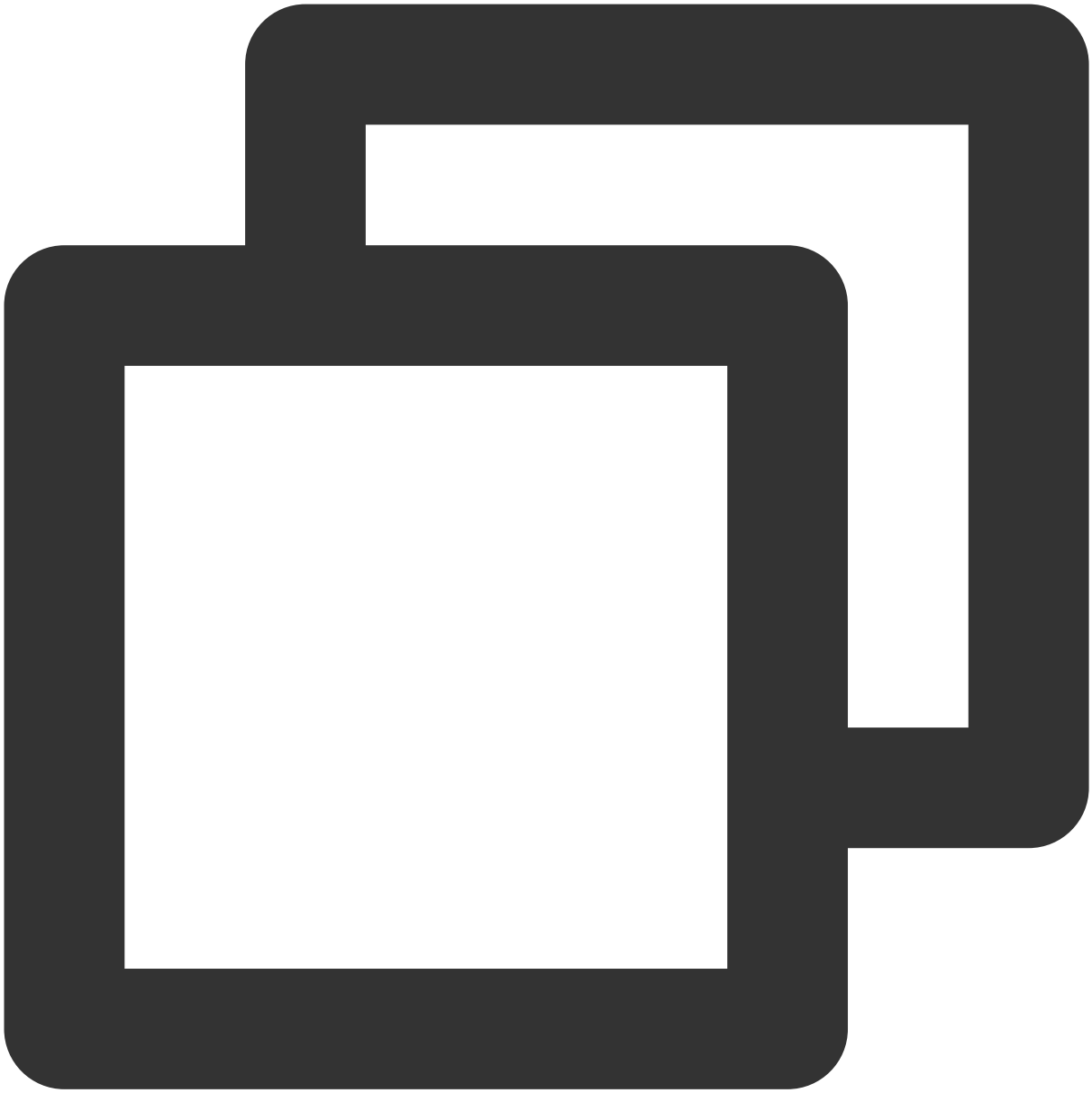
注意：

由于临时密钥在一定时效后过期，这种方式在过期后会请求失败，不建议使用。



```
string tmpSecretId = "SECRET_ID"; //"临时密钥 SecretId";
string tmpSecretKey = "SECRET_KEY"; //"临时密钥 SecretKey";
string tmpToken = "COS_TOKEN"; //"临时密钥 token";
long tmpExpireTime = 1546862502;//临时密钥有效截止时间，精确到秒
QCloudCredentialProvider cosCredentialProvider = new DefaultSessionQCloudCredential
  tmpSecretId, tmpSecretKey, tmpExpireTime, tmpToken);
```

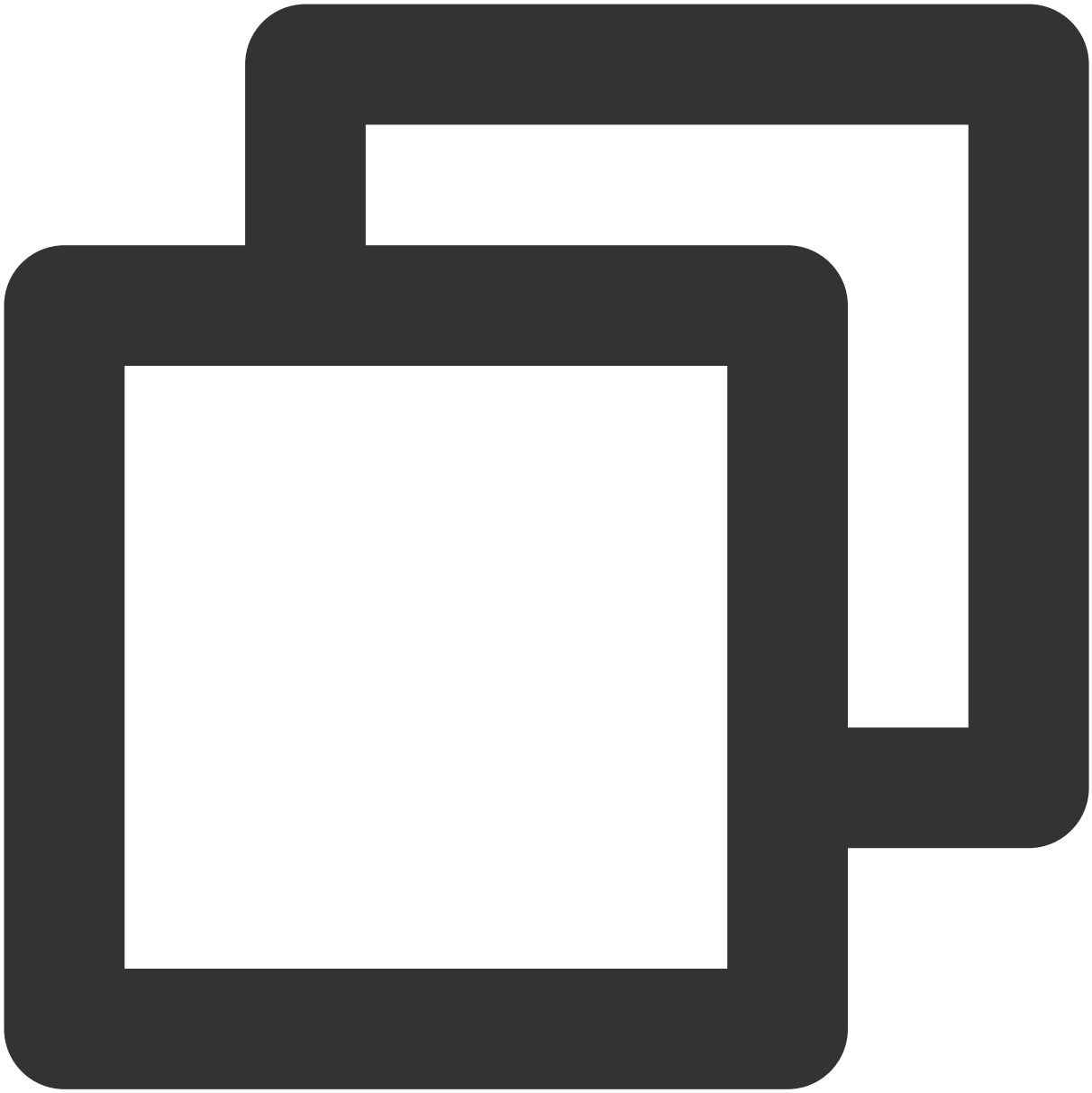
方式3：永久密钥（不建议）



```
string secretId = Environment.GetEnvironmentVariable("SECRET_ID"); //用户的 SecretId
string secretKey = Environment.GetEnvironmentVariable("SECRET_KEY"); //用户的 Secret
long durationSecond = 600;  //每次请求签名有效时长，单位为秒
QCloudCredentialProvider cosCredentialProvider = new DefaultQCloudCredentialProvide
  secretId, secretKey, durationSecond);
```

3. 初始化 CosXmlServer

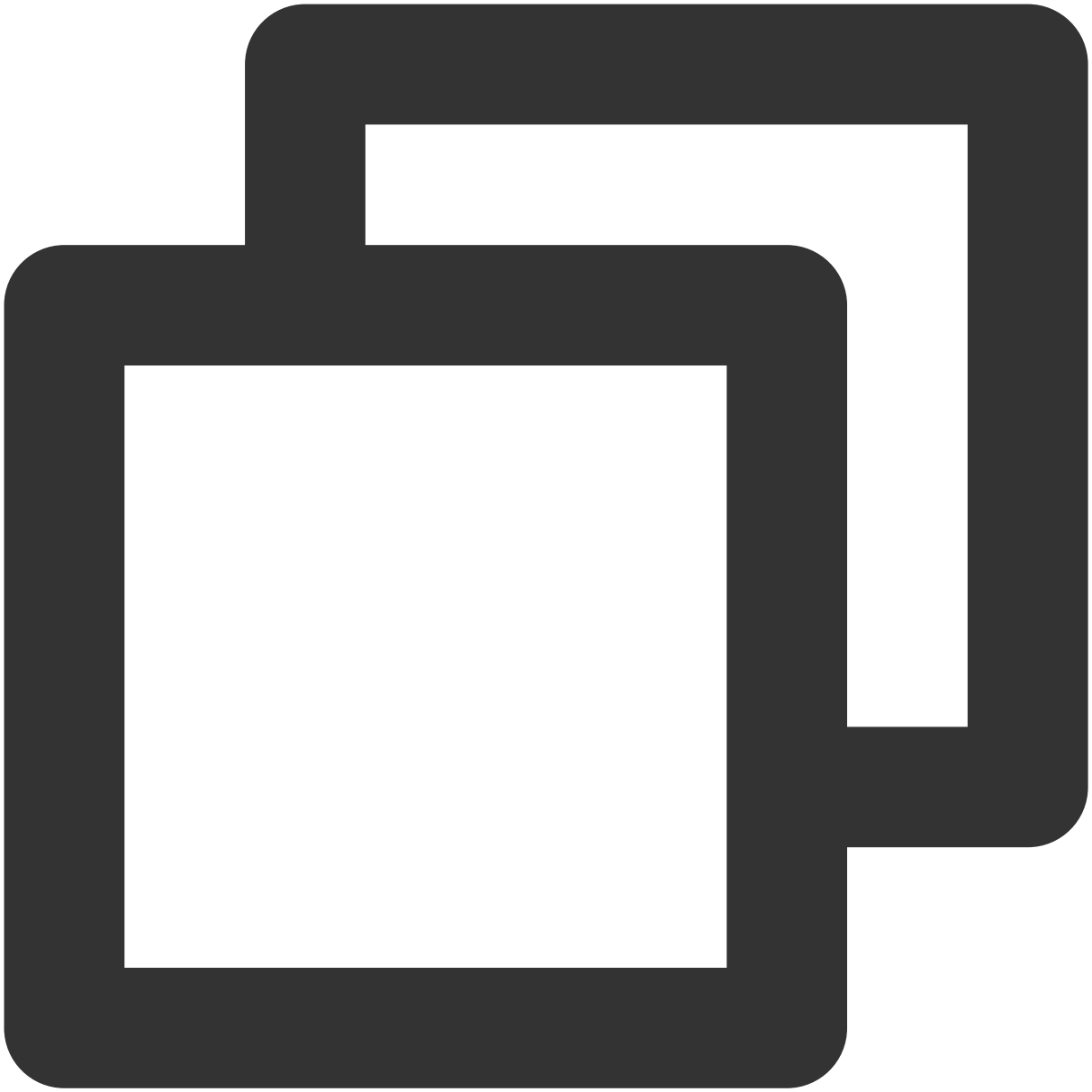
使用 `CosXmlConfig` 与 `QCloudCredentialProvider` 初始化 `CosXmlServer` 服务类。服务类建议在程序中作为**单例**使用。



```
CosXml cosXml = new CosXmlServer(config, cosCredentialProvider);
```

第三步：访问 COS 服务

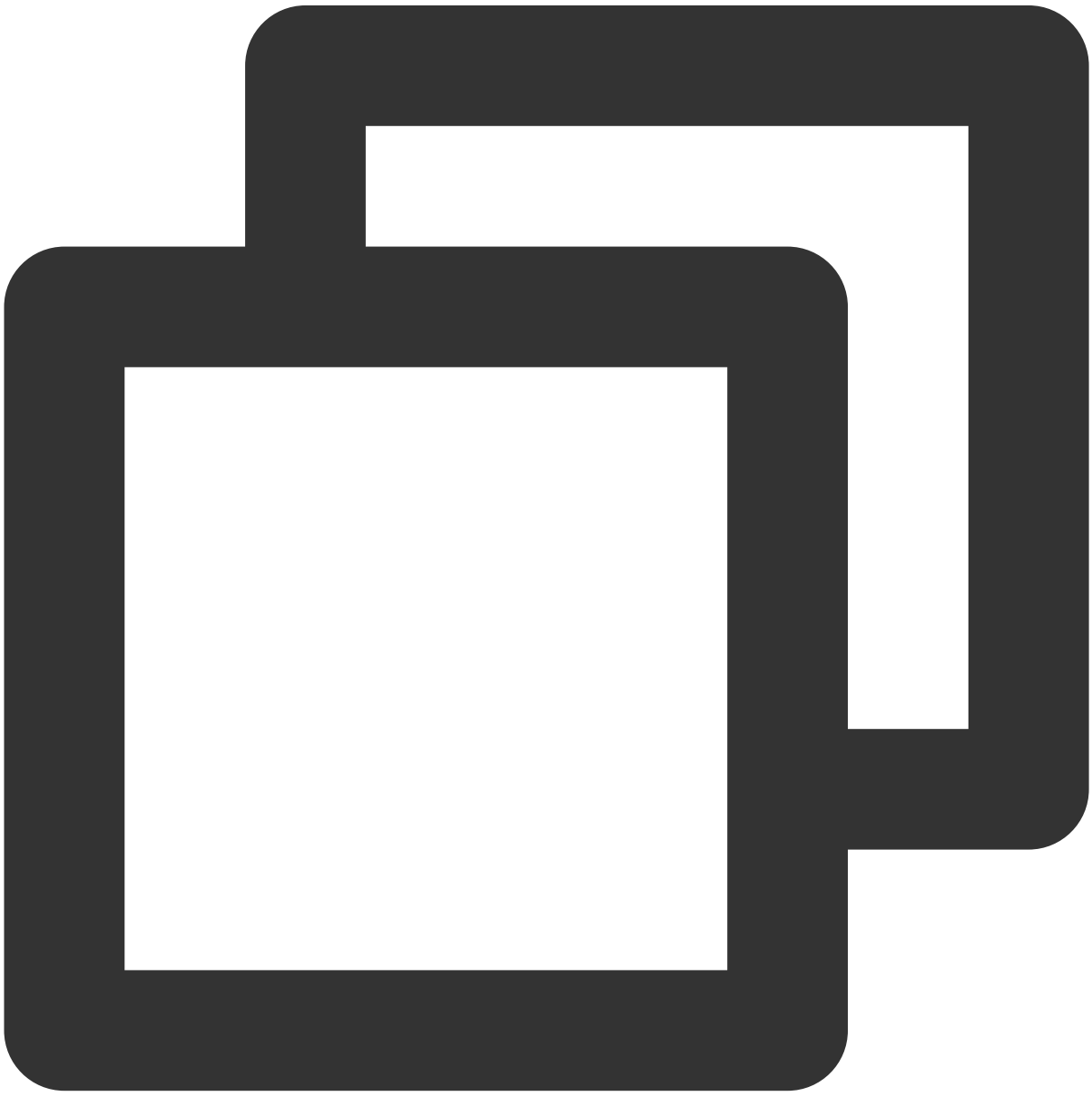
创建存储桶



```
try
{
  string bucket = "examplebucket-1250000000"; //格式：BucketName-APPID
  PutBucketRequest request = new PutBucketRequest(bucket);
  //执行请求
  PutBucketResult result = cosXml.PutBucket(request);
  //请求成功
  Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
```

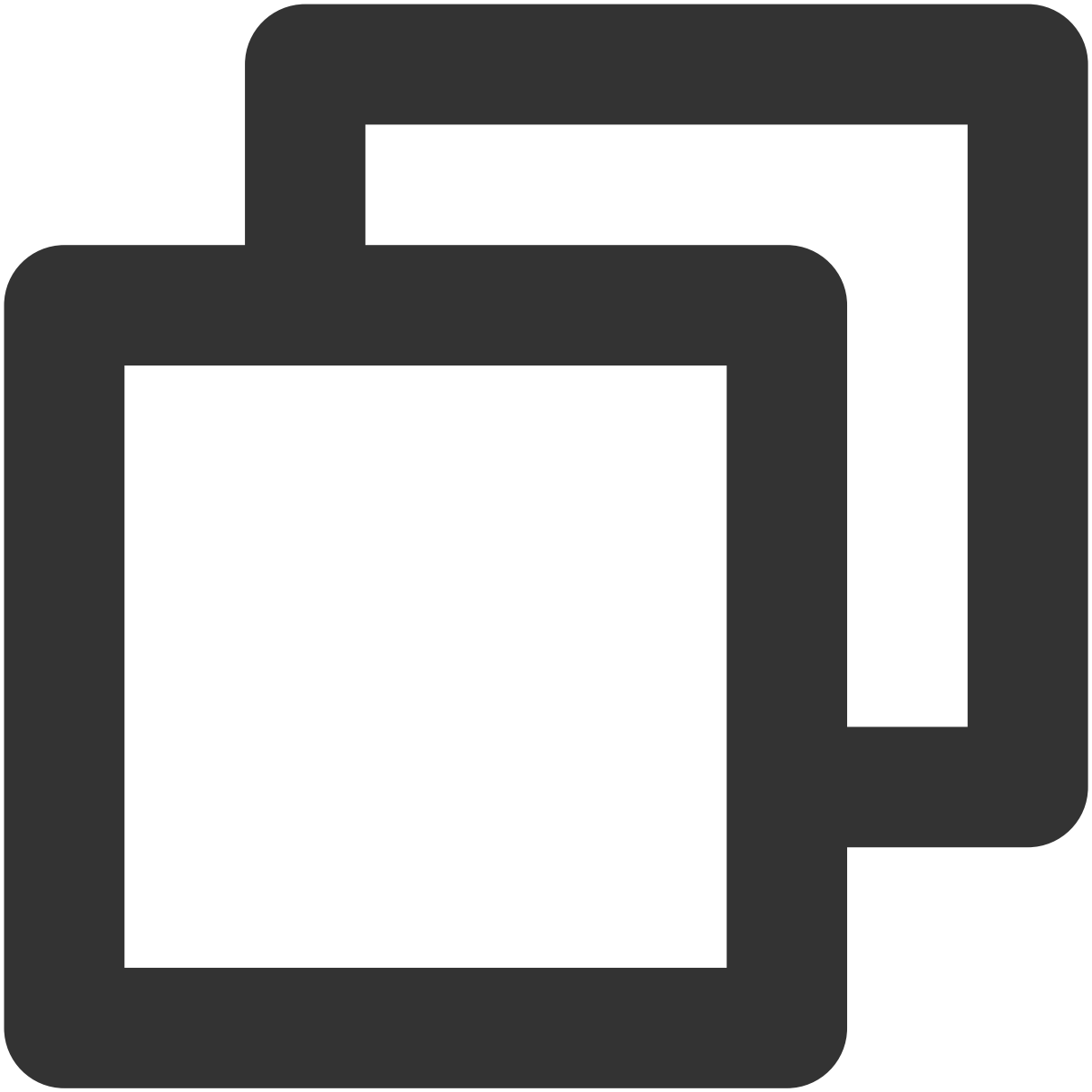
```
    //请求失败
    Console.WriteLine("CosClientException: " + clientEx);
}
catch (COSXML.CosException.CosServerException serverEx)
{
    //请求失败
    Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

查询存储桶列表



```
try
{
  GetServiceRequest request = new GetServiceRequest();
  //执行请求
  GetServiceResult result = cosXml.GetService(request);
  //得到所有的 buckets
  List<ListAllMyBuckets.Bucket> allBuckets = result.listAllMyBuckets.buckets;
}
catch (COSXML.CosException.CosClientException clientEx)
{
  //请求失败
  Console.WriteLine("CosClientException: " + clientEx);
}
catch (COSXML.CosException.CosServerException serverEx)
{
  //请求失败
  Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

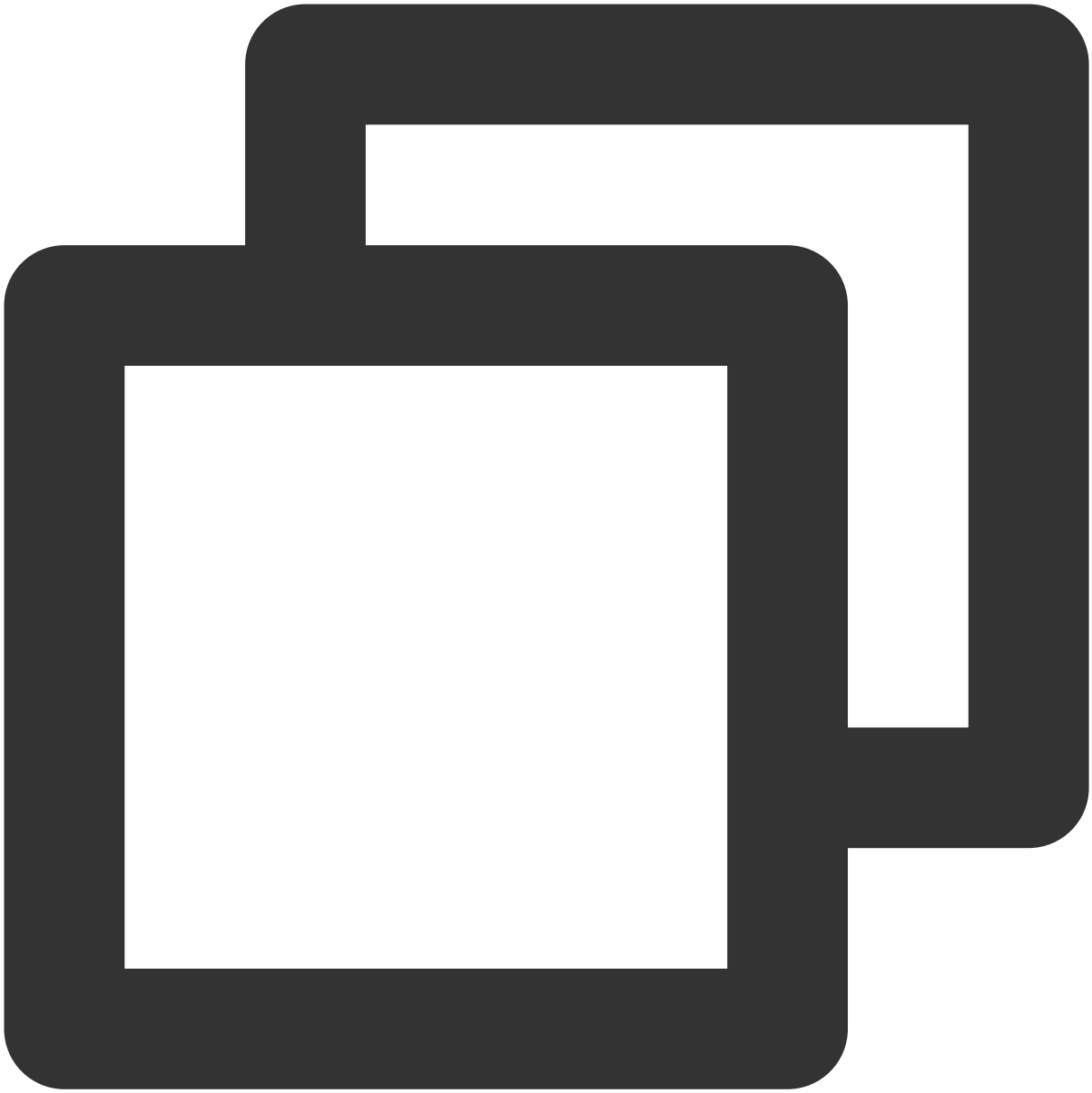
上传对象



```
// 初始化 TransferConfig
TransferConfig transferConfig = new TransferConfig();

// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXml, transferConfig);

String bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = @"temp-source-file";//本地文件绝对路径

// 上传对象
```

```
COSXMLUploadTask uploadTask = new COSXMLUploadTask(bucket, cosPath);
uploadTask.SetSrcPath(srcPath);

uploadTask.progressCallback = delegate (long completed, long total)
{
    Console.WriteLine(String.Format("progress = {0:##.##}%", completed * 100.0 / to
};

try {
  COSXML.Transfer.COSXMLUploadTask.UploadTaskResult result = await
    transferManager.UploadAsync(uploadTask);
  Console.WriteLine(result.GetResultInfo());
  string eTag = result.eTag;
} catch (Exception e) {
    Console.WriteLine("CosException: " + e);
}
```

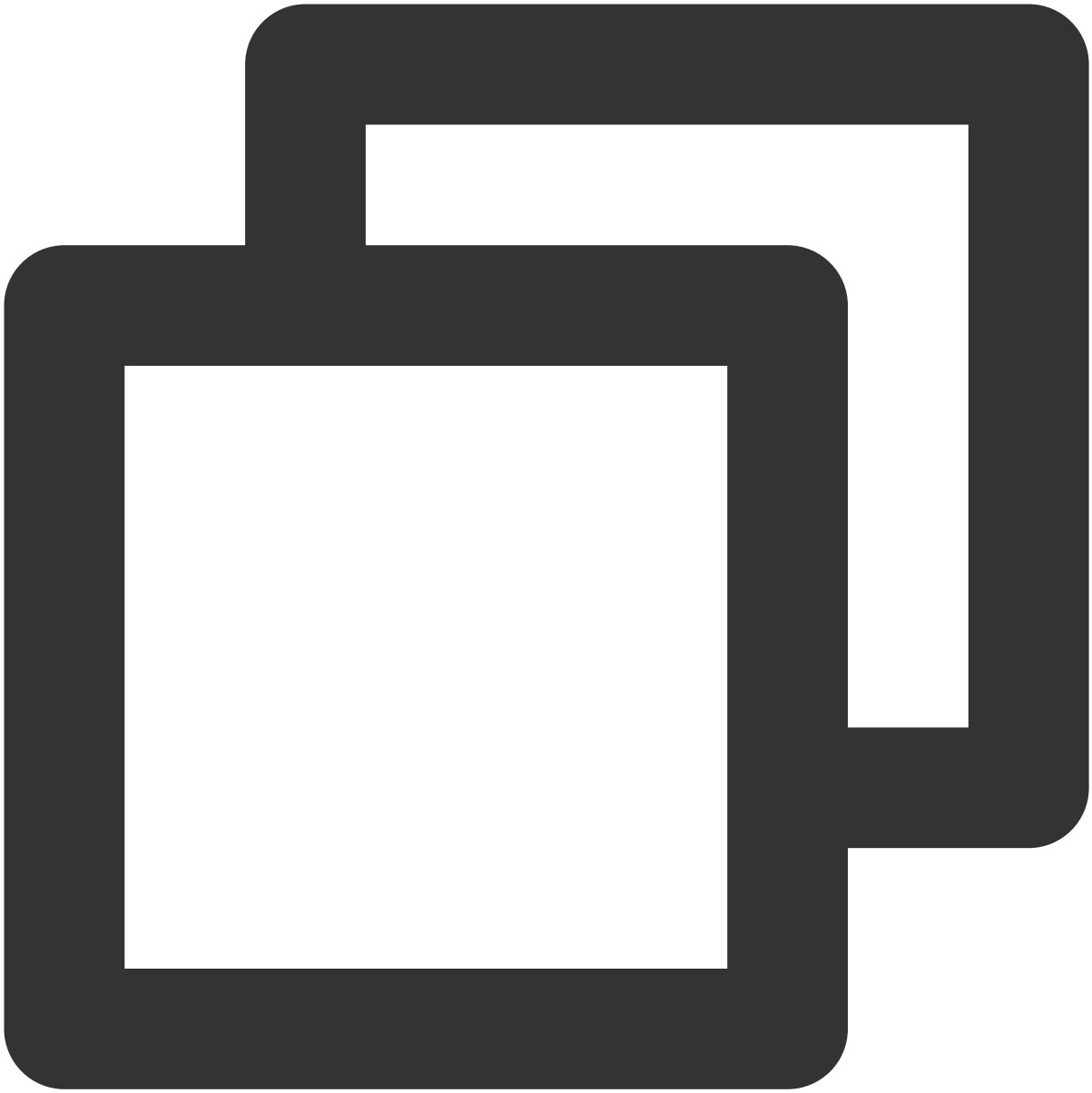
查询对象列表



```
try
{
  string bucket = "examplebucket-1250000000"; //格式：BucketName-APPID
  GetBucketRequest request = new GetBucketRequest(bucket);
  //执行请求
  GetBucketResult result = cosXml.GetBucket(request);
  //bucket的相关信息
  ListBucket info = result.listBucket;
  if (info.isTruncated) {
    // 数据被截断，记录下数据下标
    this.nextMarker = info.nextMarker;
```

```
    }
}
catch (COSXML.CosException.CosClientException clientEx)
{
    //请求失败
    Console.WriteLine("CosClientException: " + clientEx);
}
catch (COSXML.CosException.CosServerException serverEx)
{
    //请求失败
    Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

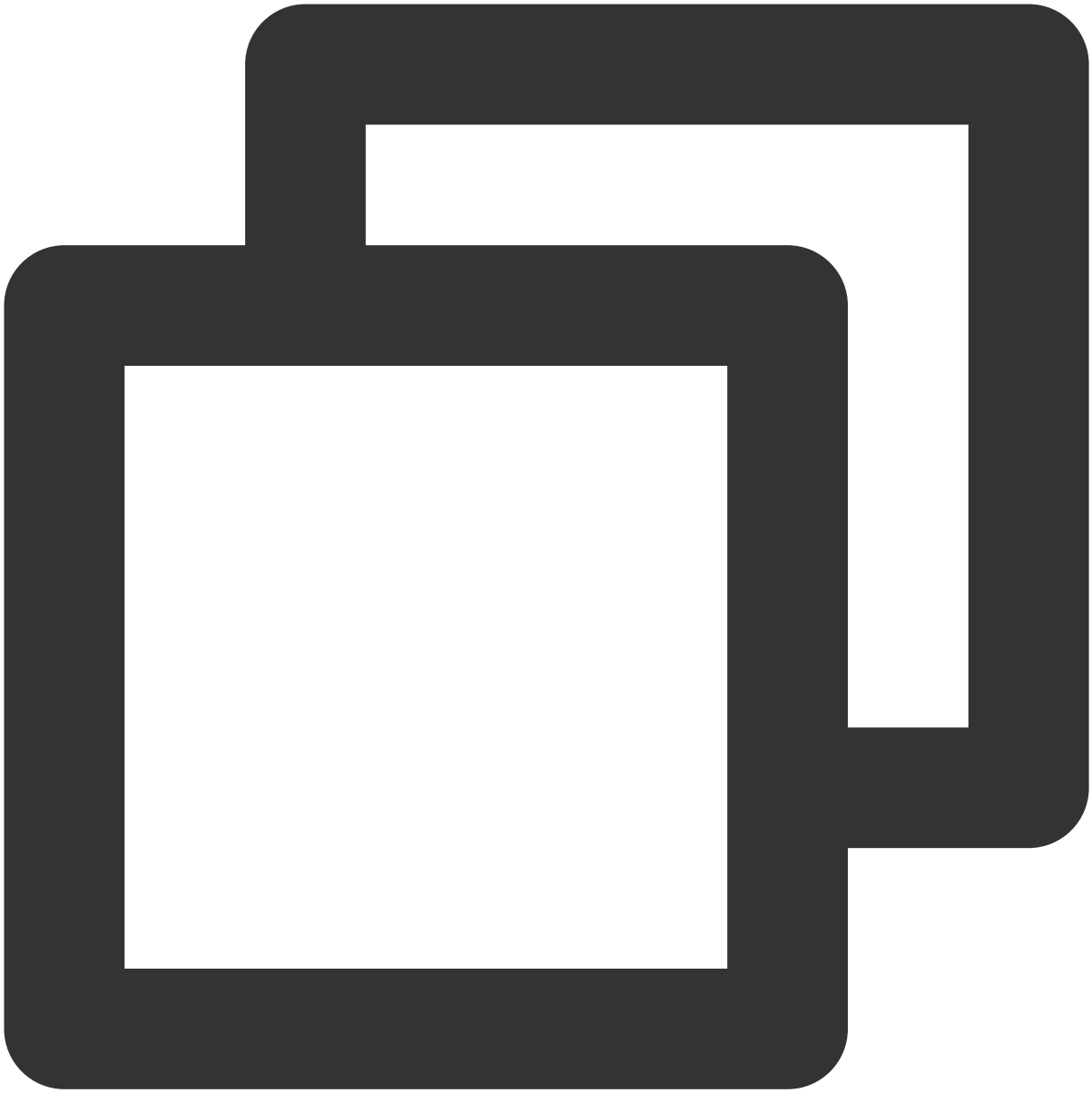
下载对象



```
// 初始化 TransferConfig
TransferConfig transferConfig = new TransferConfig();

// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXml, transferConfig);

String bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
string localDir = System.IO.Path.GetTempPath();//本地文件夹
string localFileName = "my-local-temp-file"; //指定本地保存的文件名
```

```
// 下载对象
COSXMLDownloadTask downloadTask = new COSXMLDownloadTask(bucket, cosPath,
  localDir, localFileName);

downloadTask.progressCallback = delegate (long completed, long total)
{
    Console.WriteLine(String.Format("progress = {0:##.##}%", completed * 100.0 / to
};

try {
  COSXML.Transfer.COSXMLDownloadTask.DownloadTaskResult result = await
    transferManager.DownloadAsync(downloadTask);
  Console.WriteLine(result.GetResultInfo());
  string eTag = result.eTag;
} catch (Exception e) {
    Console.WriteLine("CosException: " + e);
}
```

删除对象



```
try
{
  string bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
  string key = "exampleobject"; //对象键
  DeleteObjectRequest request = new DeleteObjectRequest(bucket, key);
  //执行请求
  DeleteObjectResult result = cosXml.DeleteObject(request);
  //请求成功
  Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
```

```
{
  //请求失败
  Console.WriteLine("CosClientException: " + clientEx);
}
catch (COSXML.CosException.CosServerException serverEx)
{
  //请求失败
  Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

图片处理
基础图片处理

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于基础图片处理的相关的 API 概览以及 SDK 示例代码。

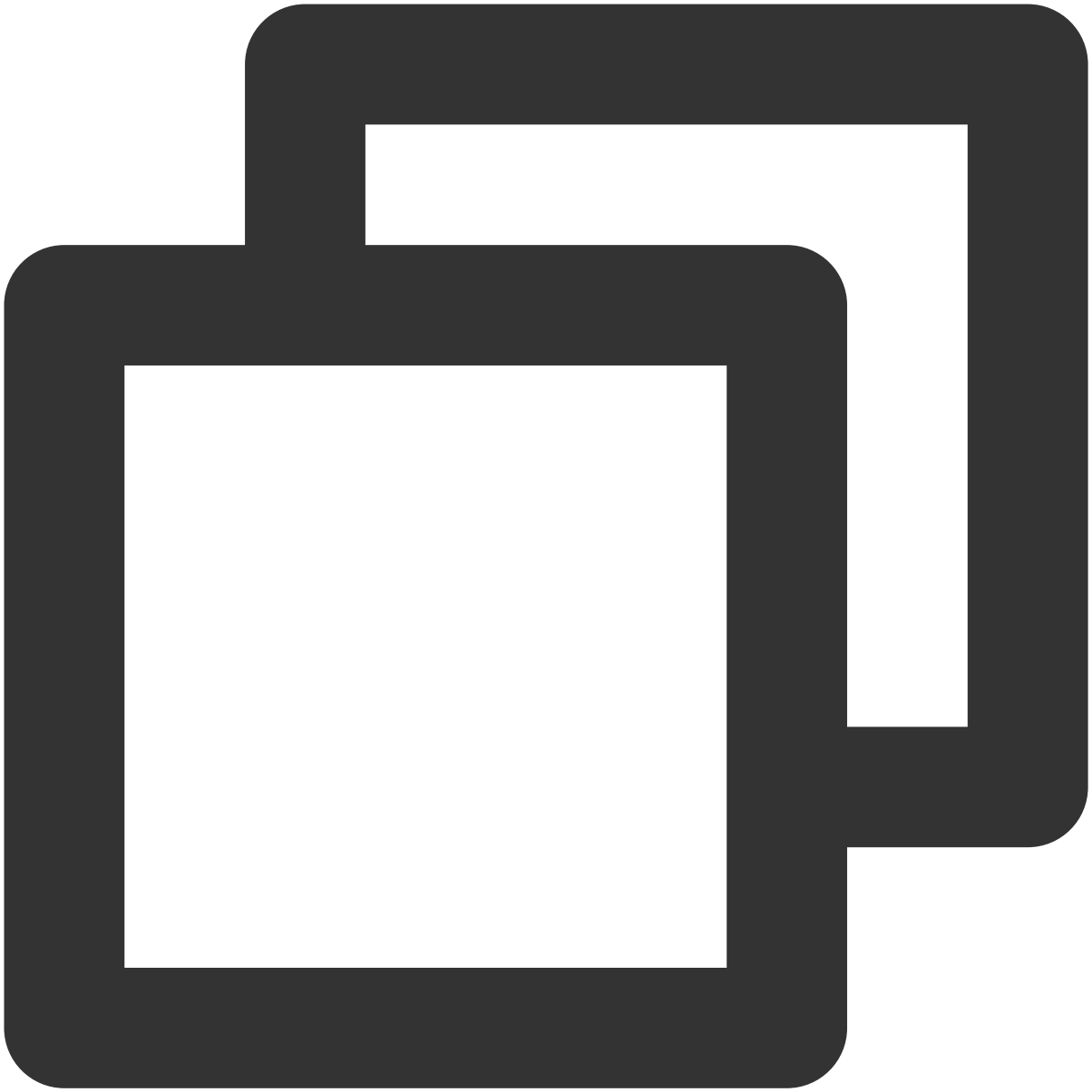
服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

上传时使用图片处理

下面示例展示了如何在上传图片时自动实现图片处理。
图片上传完成后，COS 会存储原始图片和已处理过的图片。后续用户可以通过普通的下载请求获取处理结果。

示例代码

版权所有：腾讯云计算（北京）有限责任公司



```
PutObjectRequest request = new PutObjectRequest(bucket, key, srcPath);

JObject o = new JObject();
// 不返回原图
o["is_pic_info"] = 0;
JArray rules = new JArray();
JObject rule = new JObject();
rule["bucket"] = bucket;
rule["fileid"] = "desample_photo.jpg";
//处理参数，规则参见：https://cloud.tencent.com/document/product/436/44879
rule["rule"] = "imageMogr2/thumbnail/400x400";
```

```
rules.Add(rule);
o["rules"] = rules;

string ruleString = o.ToString(Formatting.None);
request.SetRequestHeader("Pic-Operations", ruleString);
//执行请求
PutObjectResult result = cosXml.PutObject(request);
```

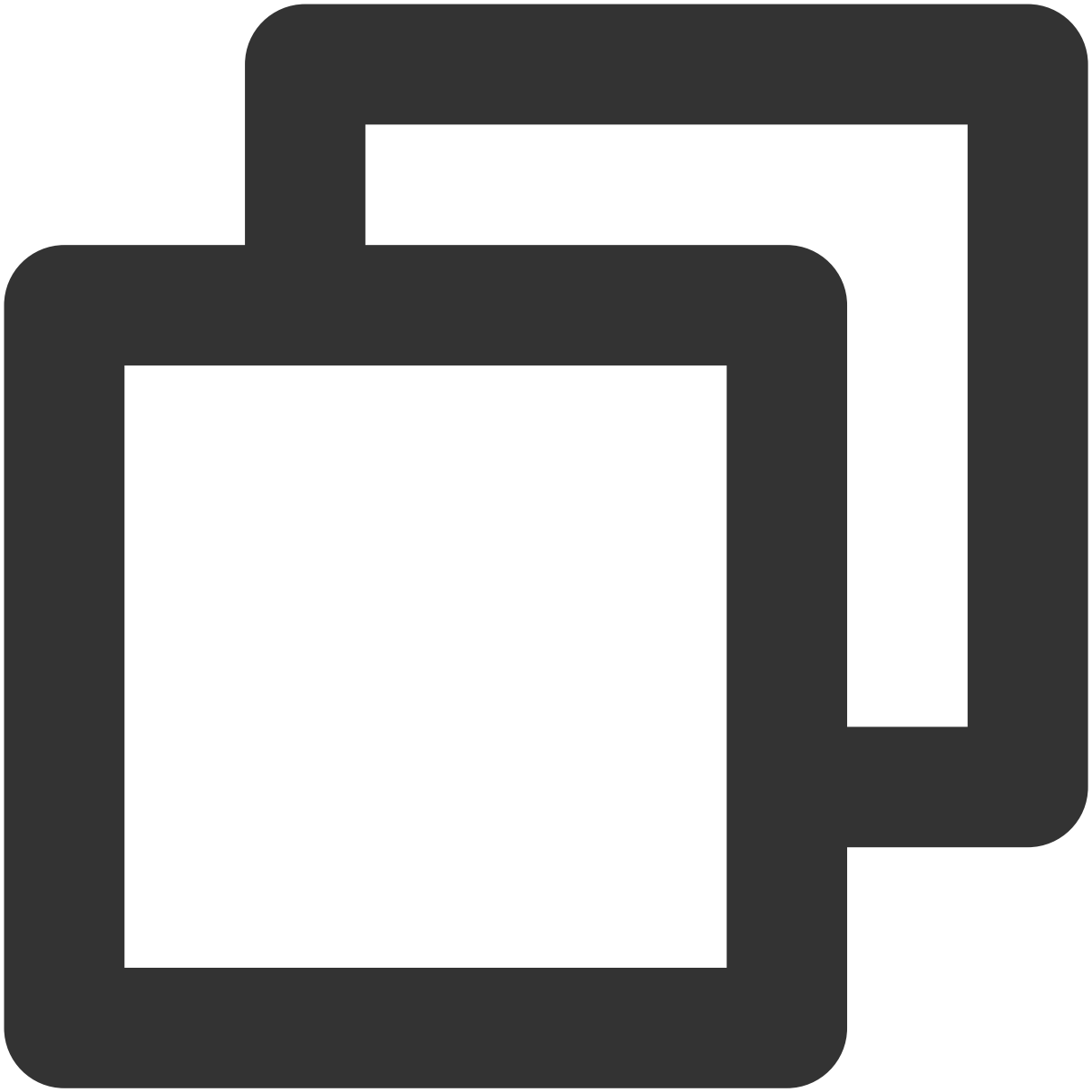
说明：

更多完整示例，请前往 GitHub 查看。

对云上数据进行图片处理

下面示例展示了如何在对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

示例代码



```
JObject o = new JObject();
// 不返回原图
o["is_pic_info"] = 0;
JArray rules = new JArray();
JObject rule = new JObject();
rule["bucket"] = bucket;
rule["fileid"] = "desample_photo.jpg";
//处理参数，规则参见：https://cloud.tencent.com/document/product/436/44879
rule["rule"] = "imageMogr2/thumbnail/400x400";
rules.Add(rule);
o["rules"] = rules;
```

版权所有：腾讯云计算（北京）有限责任公司

```
string ruleString = o.ToString(Formatting.None);

ImageProcessRequest request = new ImageProcessRequest(bucket, key, ruleString);
ImageProcessResult result = cosXml.ImageProcess(request);
```

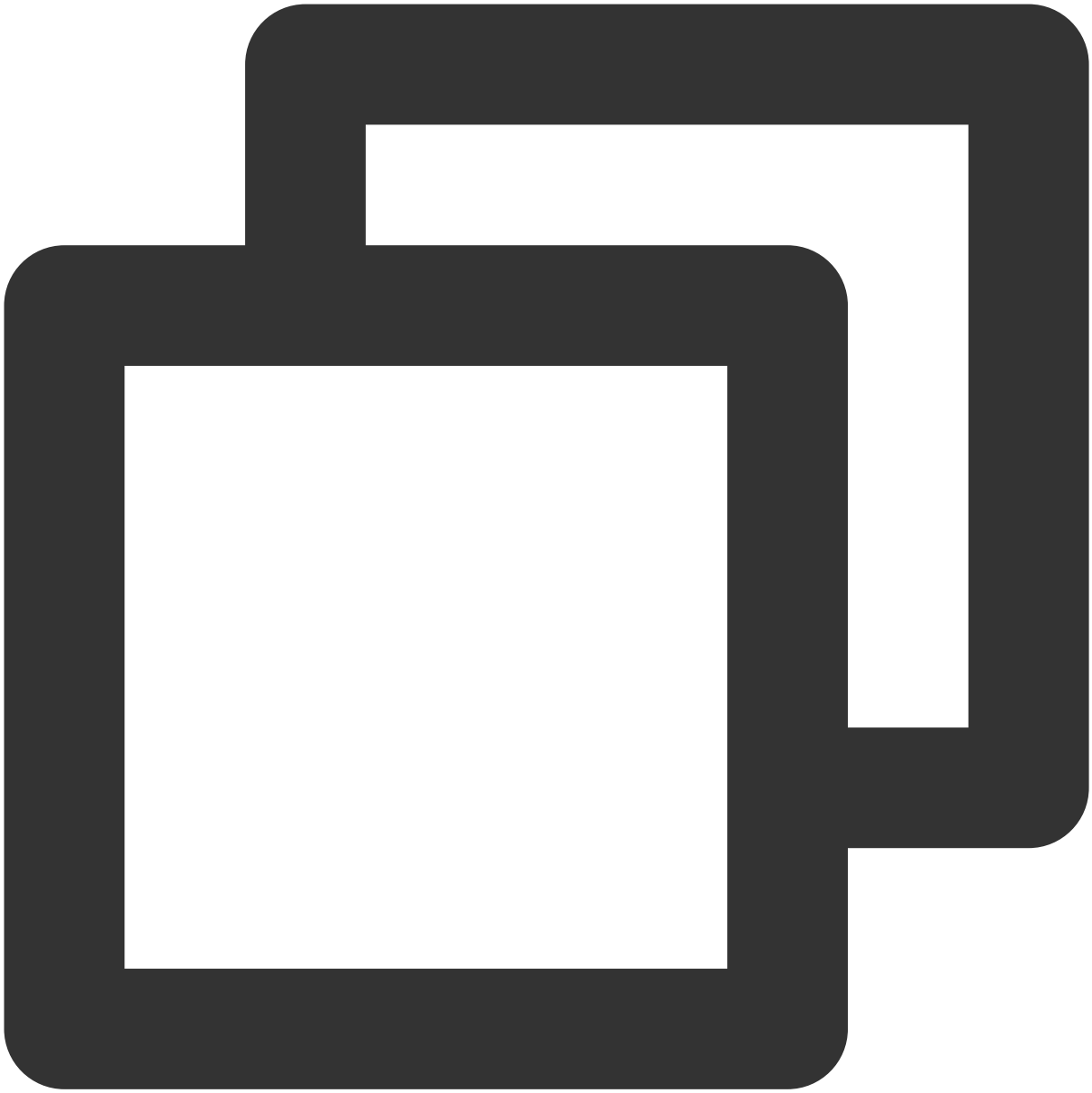
说明：

更多完整示例，请前往 GitHub 查看。

下载时进行图片处理

下面示例展示了如何在对已存储在 COS 的图片在下载时进行处理操作。

示例代码



```
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, key, localDir, loc
//处理参数，这里的实例是格式转换为 TPG 图片，规则参见：https://cloud.tencent.com/document/p
getObjectRequest.SetQueryParameter("imageMogr2/format/tpg", null);
```

说明：

更多完整示例，请前往 GitHub 查看。

盲水印

最近更新时间：2024-03-04 16:55:55

简介

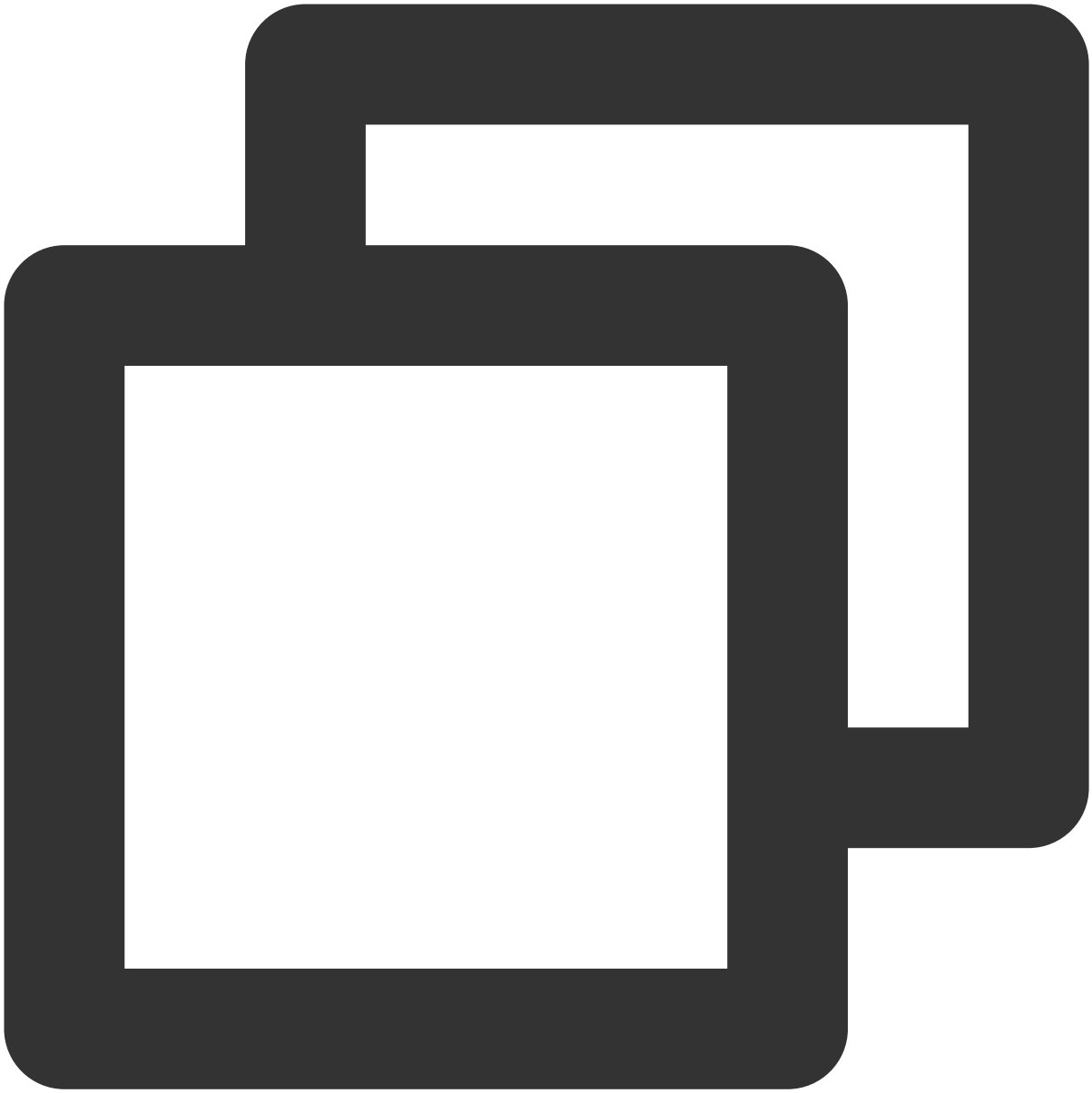
本文档提供关于盲水印相关的 API 概览以及 SDK 示例代码。

添加盲水印

功能说明

盲水印支持在上传时添加以及下载时添加。

示例代码一：上传时添加盲水印



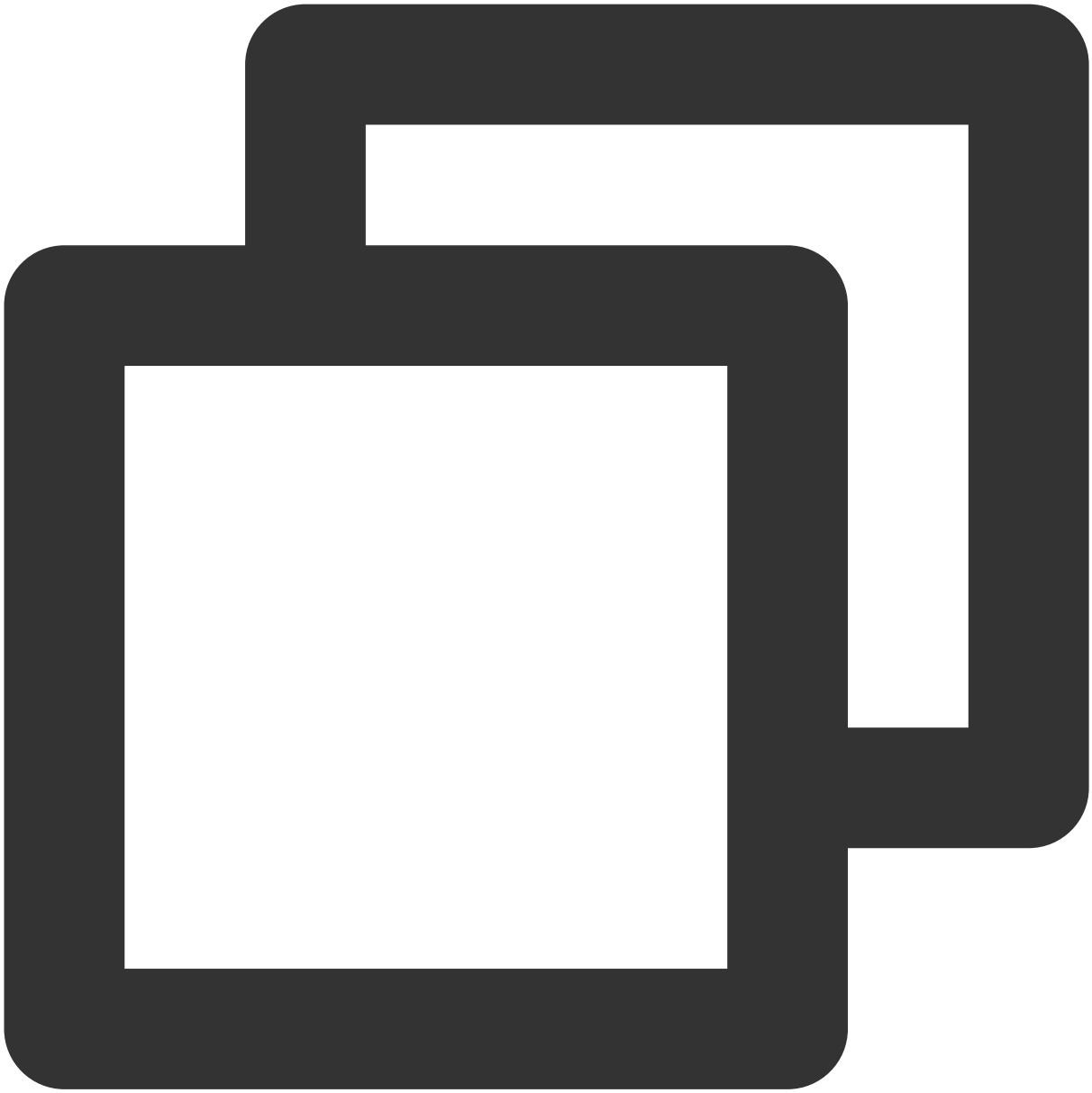
```
PutObjectRequest request = new PutObjectRequest(bucket, key, srcPath);

JObject o = new JObject();
// 不返回原图
o["is_pic_info"] = 0;
JArray rules = new JArray();
JObject rule = new JObject();
rule["bucket"] = bucket;
rule["fileid"] = key;
//处理参数，规则参见：https://cloud.tencent.com/document/product/436/46782
rule["rule"] = "watermark/3/type/<type>/image/<imageUrl>/text/<text>/level/<level>"
```

```
rules.Add(rule);
o["rules"] = rules;

string ruleString = o.ToString(Formatting.None);
request.SetRequestHeader("Pic-Operations", ruleString);
//执行请求
PutObjectResult result = cosXml.PutObject(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

示例代码二：下载时添加盲水印



```
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, key, localDir, loc
//处理参数，规则参见：https://cloud.tencent.com/document/product/436/46782
getObjectRequest.SetQueryParameter("watermark/3/type/<type>/image/<imageUrl>/text/<

GetObjectResult result = cosXml.GetObject(getObjectRequest);
```

说明：

更多完整示例，请前往 GitHub 查看。

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体 bucket 接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

功能说明

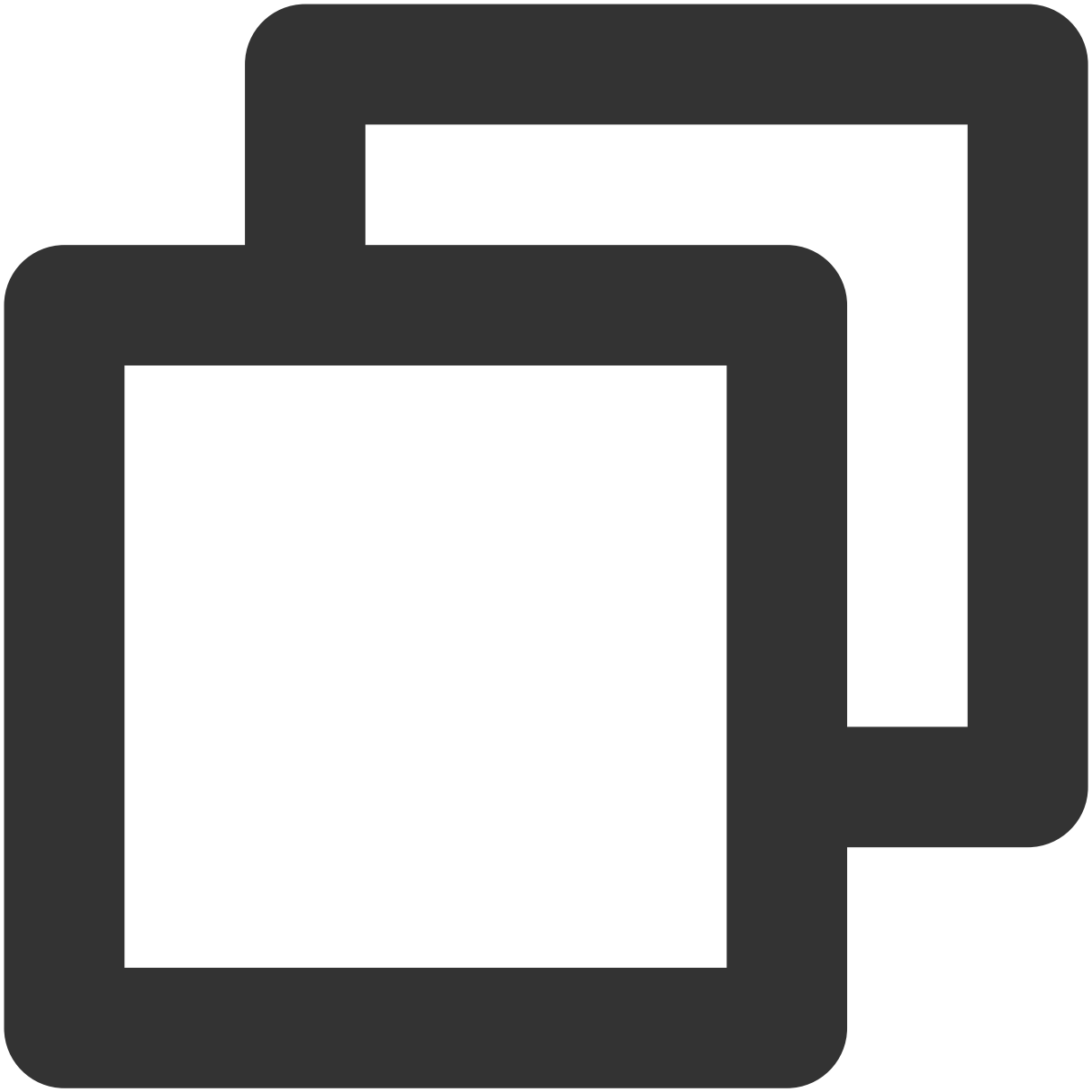
DescribeMediaBuckets 接口用于查询已经开通媒体处理功能的存储桶。

说明：

DescribeMediaBuckets 接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本更新日志，请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using COSXML;

namespace COSSnippet
{
    public class DescribeMediaBucketModel {

      private CosXml cosXml;
```

```
        DescribeMediaBucketModel() {
          CosXmlConfig config = new CosXmlConfig.Builder()
            .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://www.t
            .Build();

          string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
          string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
          long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
          QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
            secretKey, durationSecond);

          this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
        }

        // 获取开通了万象功能的 Buckets 列表
        public void DescribeMediaBucket()
        {
          //.cssg-snippet-body-start:[DescribeMediaBucket]
          DescribeMediaBucketsRequest request = new DescribeMediaBucketsRequest();
          // 执行请求
          DescribeMediaBucketsResult result = cosXml.DescribeMediaBuckets(request);
          Console.WriteLine(result.GetResultInfo());
          // 遍历Bucket列表
          for (int i = 0; i < result.mediaBuckets.MediaBucketList.Count; i++)
          {
            Console.WriteLine(result.mediaBuckets.MediaBucketList[i].BucketId);
            Console.WriteLine(result.mediaBuckets.MediaBucketList[i].Region);
            Console.WriteLine(result.mediaBuckets.MediaBucketList[i].CreateTime);
          }
          //.cssg-snippet-body-end
        }

        static void Main(string[] args)
        {
          DescribeMediaBucketModel m = new DescribeMediaBucketModel();
          // 获取媒体Buckets列表
          m.DescribeMediaBucket();
          // .cssg-methods-pragma
        }
      }
  }
```

说明：

更多完整示例，请前往 GitHub 查看。

媒体信息接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

功能说明

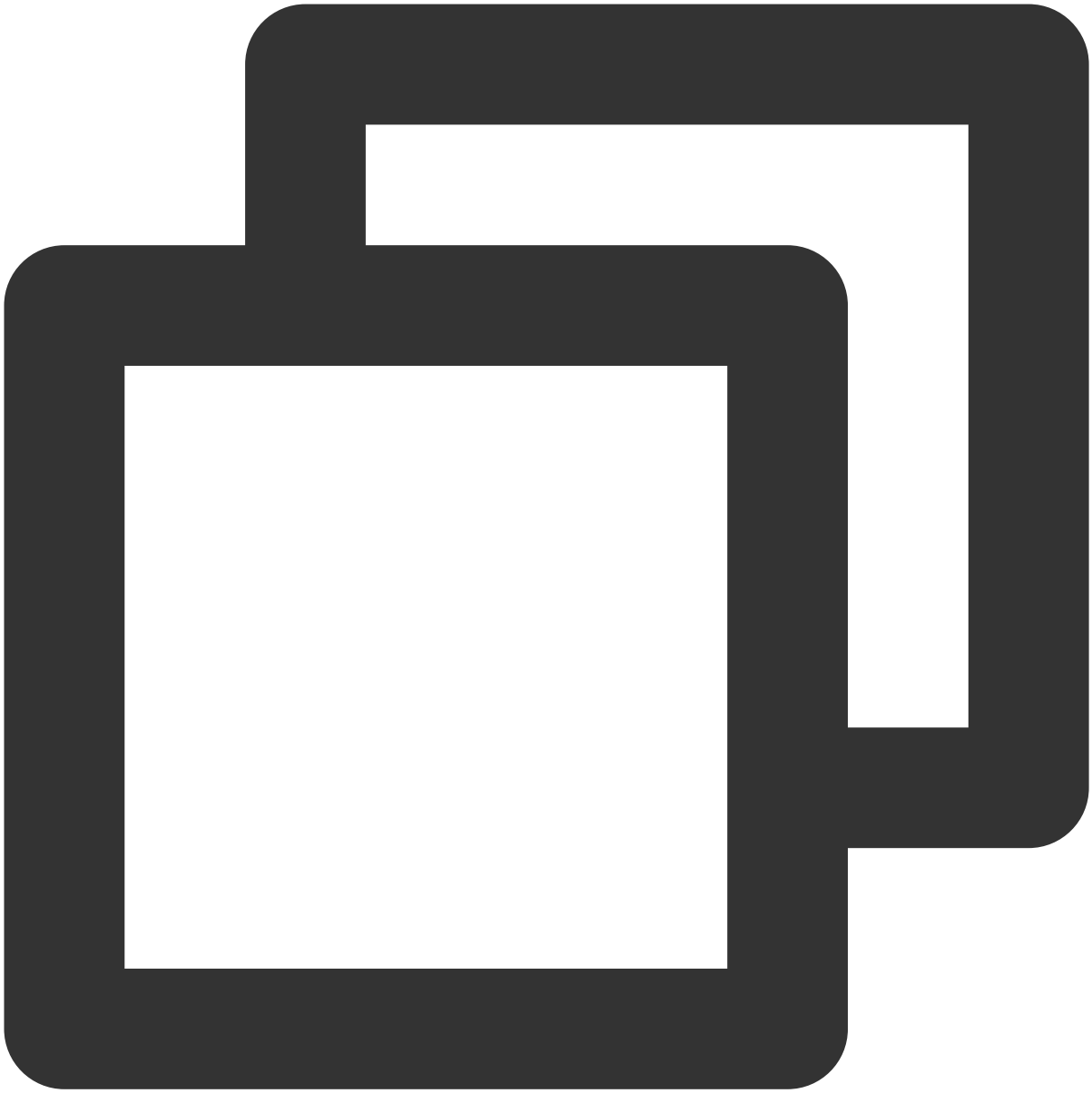
用于查询媒体文件的信息。使用此接口需要在控制台开通存储桶的媒体处理功能。

说明：

GetMediainfo 接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本更新日志，请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using COSXML;

namespace COSSnippet
{
    public class GetMediaInfoModel {

      private CosXml cosXml;
```

```
GetMediaInfoModel() {
  CosXmlConfig config = new CosXmlConfig.Builder()
    .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://www.t
    .Build();

  string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
  string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
  long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
  QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
    secretKey, durationSecond);

  this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
}

/// 获取媒体文件信息
public void GetMediaInfo()
{
  //.cssg-snippet-body-start:[GetMediaInfo]
  // 存储桶名称, 此处填入格式必须为 BucketName-APPID, 其中 APPID 获取参考 https://co
  string bucket = "examplebucket-1250000000";
  string key = "mediafile"; // 媒体文件的对象键, 需要替换成桶内存在的媒体文件的对象键
  GetMediaInfoRequest request = new GetMediaInfoRequest(bucket, key);
  // 执行请求
  GetMediaInfoResult result = cosXml.GetMediaInfo(request);
  Console.WriteLine(result.GetResultInfo());
  // 获取视频媒体信息
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Video);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Video.Index);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Video.CodecName);
  // 获取音频信息
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Audio);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Audio.Index);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Audio.CodecName);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Stream.Audio.CodecLongNa
  // 获取Format字段
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Format);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Format.NumStream);
  Console.WriteLine(result.mediaInfoResult.MediaInfo.Format.NumProgram);
  //.cssg-snippet-body-end
}

static void Main(string[] args)
{
  GetMediaInfoModel m = new GetMediaInfoModel();
  /// 获取媒体文件信息
  m.GetMediaInfo();
```

版权所有：腾讯云计算（北京）有限责任公司

```
          // .cssg-methods-pragma
        }
      }
  }
```

说明：

更多完整示例，请前往 GitHub 查看。

媒体截图接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

功能说明

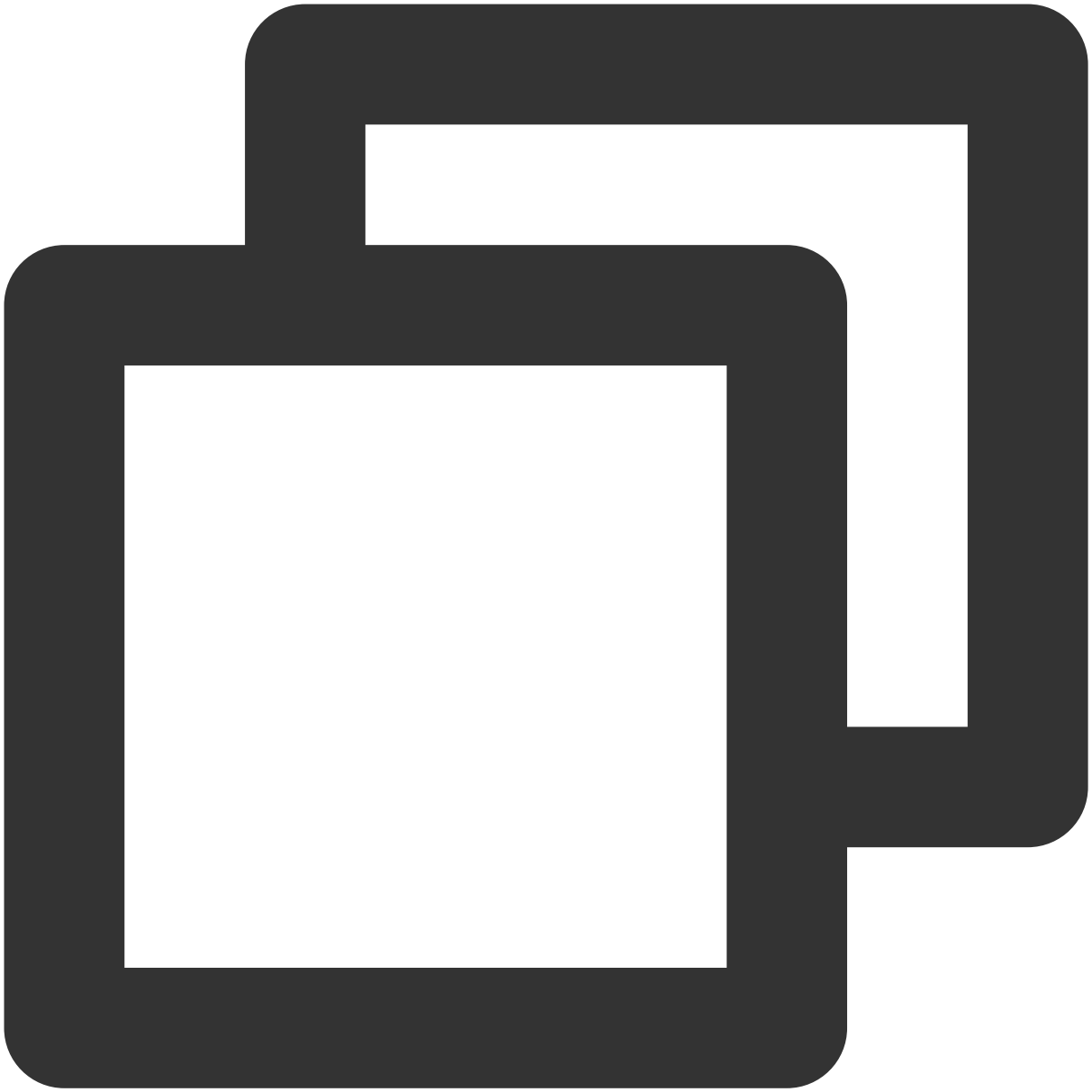
用于查询媒体文件在某个时间的截图。

说明：

GetSnapshot 接口从 5.4.24 版本开始支持，下载新版SDK前往 Releases 或参见 快速入门。

查看版本更新日志，请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using COSXML;

namespace COSSnippet
{
    public class GetSnapshotModel {

      private CosXml cosXml;
```

```
        GetSnapshotModel() {
          CosXmlConfig config = new CosXmlConfig.Builder()
            .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://www.t
            .Build();

          string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
          string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
          long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
          QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
            secretKey, durationSecond);

          this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
        }

        /// 视频截帧
        public void GetSnapshot()
        {
          // 存储桶名称, 此处填入格式必须为 BucketName-APPID, 其中 APPID 获取参考 https://co
          string bucket = "examplebucket-1250000000";
          string key = "video.mp4"; // 媒体文件的对象键, 需要替换成桶内存在的媒体文件的对象键
          float time = 1.5F; // 截取的具体时间, 用浮点数表示
          string destPath = @"temp-source-file"; // 截图文件保存路径, 需要替换成本地具体路径
          GetSnapshotRequest request = new GetSnapshotRequest(bucket, key, time, dest
          // 执行请求
          GetSnapshotResult result = cosXml.GetSnapshot(request);
          Console.WriteLine(result.GetResultInfo());
        }

        static void Main(string[] args)
        {
          GetSnapshotModel m = new GetSnapshotModel();
          /// 视频截帧
          m.GetSnapshot();
        }
      }
  }
```

说明：

更多完整示例，请前往 GitHub 查看。

内容审核
图片审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核的 API 概览以及 SDK 示例代码。

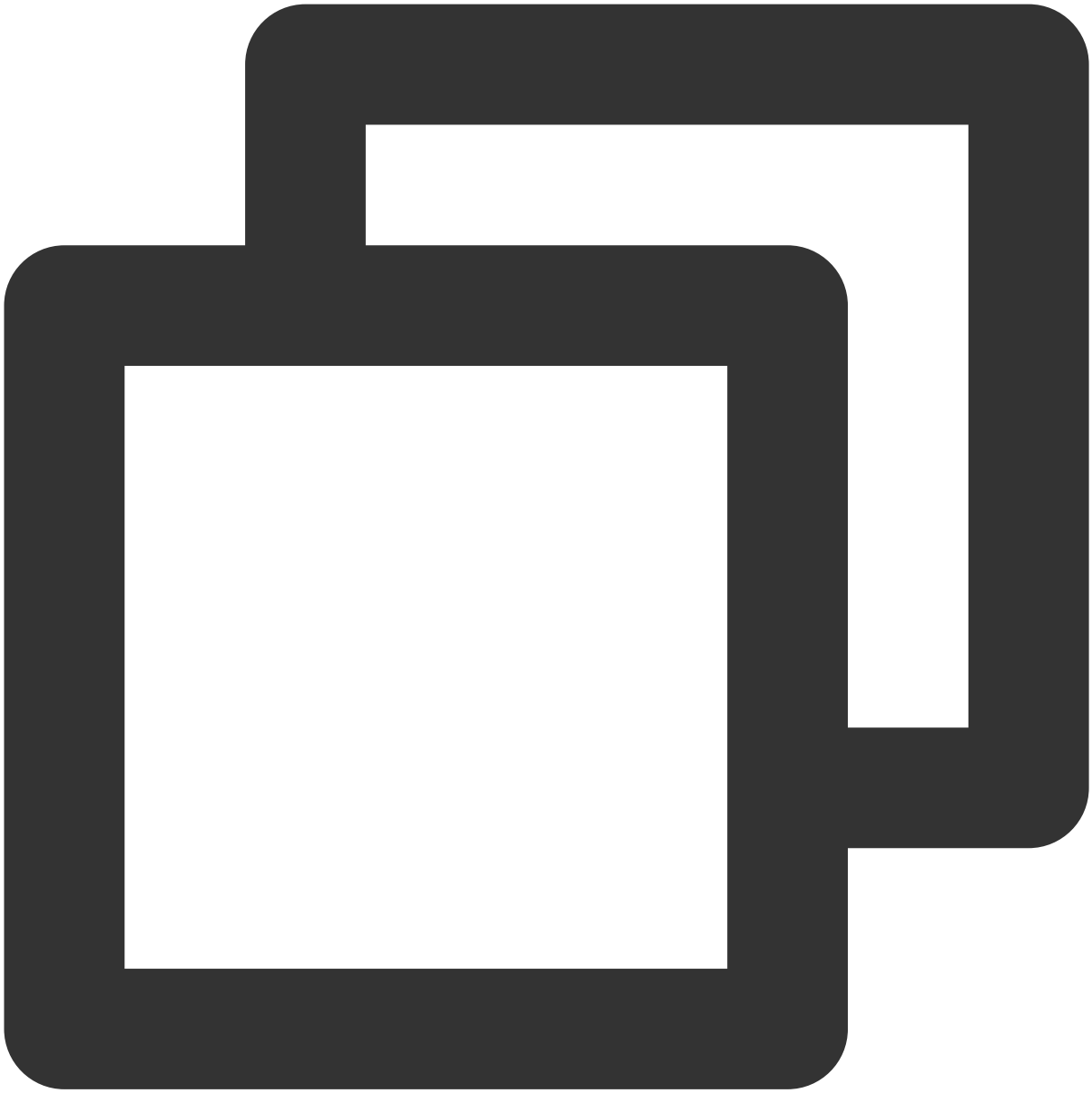
API	操作描述
图片审核	对对象存储（Cloud Object Storage，COS）存量数据进行涉黄、违法违规以及广告引导类图片的扫描

存量图片审核

功能说明

下面示例展示了如何在对已存储在 COS 的图片进行审核，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using COSXML;

namespace COSSnippet
{
    public class PictureOperationModel {

      private CosXml cosXml;

      PictureOperationModel() {
```

```
            CosXmlConfig config = new CosXmlConfig.Builder()
              .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
              .Build();

            string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
            string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
            long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
            QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
              secretKey, durationSecond);

            this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
        }

        /// 图片审核
        public void SensitiveContentRecognition()
        {
            // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
            string bucket = "examplebucket-1250000000";
            string key = "exampleobject"; //对象键
            //.cssg-snippet-body-start:[sensitive-content-recognition]
            SensitiveContentRecognitionRequest request =
              new SensitiveContentRecognitionRequest(bucket, key, "ads");
            SensitiveContentRecognitionResult result = cosXml.SensitiveContentRecogniti
        }

        static void Main(string[] args)
        {
            PictureOperationModel m = new PictureOperationModel();
            /// 图片审核
            m.SensitiveContentRecognition();
        }
    }
}
```

说明：

更多完整示例，请前往 GitHub 查看。

视频审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核的 API 概览和 SDK 示例代码。

API	操作描述
提交视频审核任务	用于提交一个视频审核任务
查询视频审核任务结果	用于查询指定的视频审核任务

提交视频审核任务

功能说明

视频审核功能为异步任务方式，您可以通过提交视频审核任务审核您的视频文件，然后通过查询视频审核任务接口查询审核结果。

下面示例展示了如何提交一个视频审核任务，然后通过 JobId 查询视频审核任务结果。

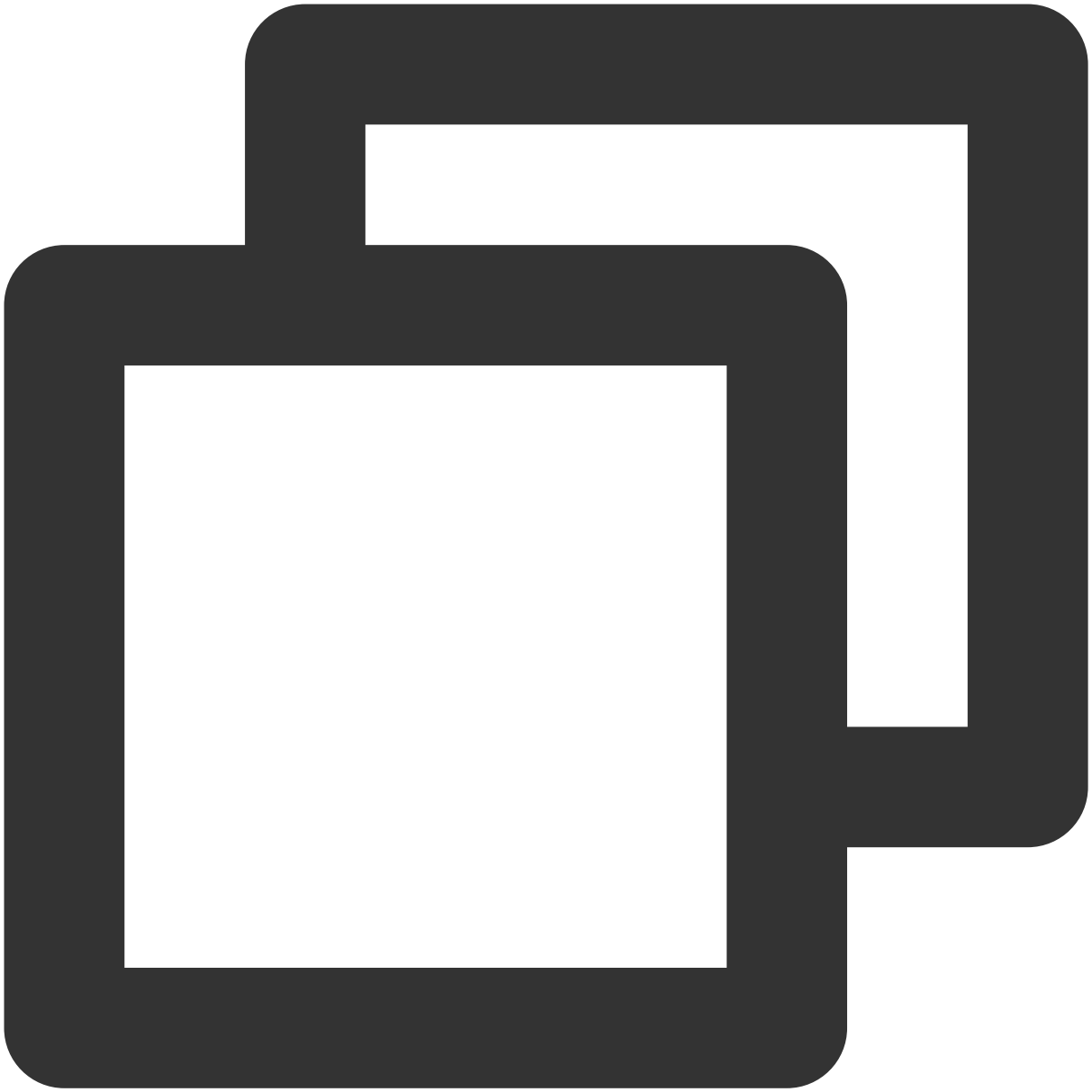
说明：

此操作需要存储桶开通数据万象相关功能。

视频审核接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本 Changelog 请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitVideoCensorJobModel {

      private CosXml cosXml;
```

```
    SubmitVideoCensorJobModel() {
      CosXmlConfig config = new CosXmlConfig.Builder()
        .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
        .Build();

      string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
      string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
      long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
      QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
        secretKey, durationSecond);

      this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
    }

    /// 提交视频审核任务
    public string SubmitVideoCensorJob()
    {
      // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
      string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
      SubmitVideoCensorJobRequest request = new SubmitVideoCensorJobRequest(bucke
      request.SetCensorObject("video.mp4"); // 媒体文件的对象键, 需要替换成桶内存在的媒体
      // 审核的场景类型, 有效值：Porn（涉黄）、Ads（广告）, 可以传入多种类型, 不同类型以逗号分
      request.SetDetectType("Porn,Ads");
      // 视频画面的审核通过视频截帧能力截取出一定量的截图, 通过对截图逐一审核而实现的, 该参数用
      request.SetSnapshotMode("Average"); // 截帧模式。Interval 表示间隔模式；Average
      request.SetSnapshotCount("100"); // 视频截帧数量, 范围为(0, 10000]
      request.SetSnapshotTimeInterval("1.0"); // 视频截帧频率, 范围为(0, 60], 单位为秒
      // 执行请求
      SubmitCensorJobResult result = cosXml.SubmitVideoCensorJob(request);
      Console.WriteLine(result.GetResultInfo());
      Console.WriteLine(result.censorJobsResponse.JobsDetail.JobId);
      Console.WriteLine(result.censorJobsResponse.JobsDetail.State);
      Console.WriteLine(result.censorJobsResponse.JobsDetail.CreationTime);
      return result.censorJobsResponse.JobsDetail.JobId;
    }

    static void Main(string[] args)
    {
      SubmitVideoCensorJobModel m = new SubmitVideoCensorJobModel();
      /// 提交审核任务, JobId唯一标识了本次提交的审核任务的结果
      string JobId = m.SubmitVideoCensorJob();
      /// 打印 JobId
      Console.WriteLine("JobId : " + JobId);
    }
  }
}
```

说明：

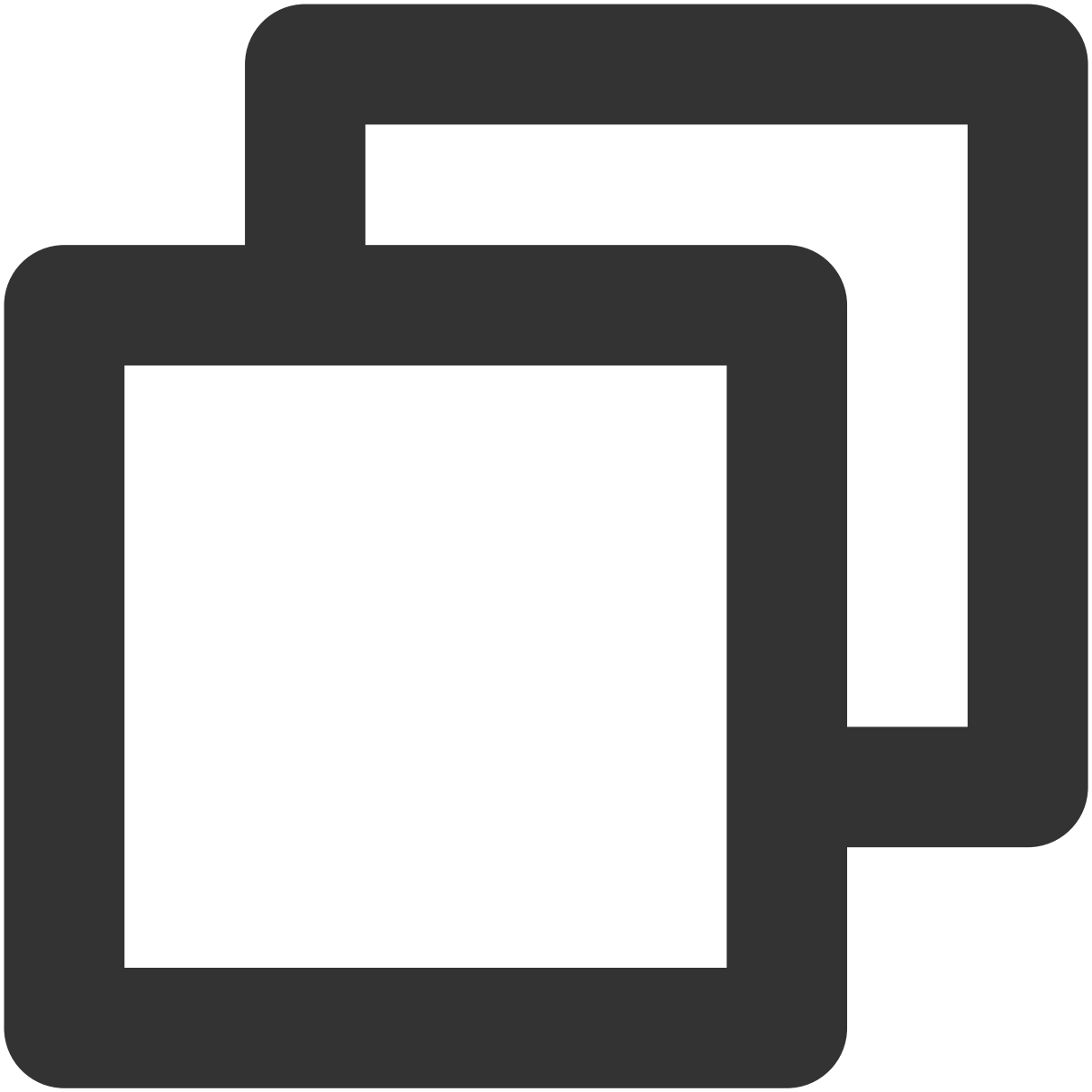
更多完整示例，请前往 GitHub 查看。

查询视频审核任务结果

功能说明

用于查询视频审核任务执行情况和结果。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitVideoCensorJobModel {

      private CosXml cosXml;
```

```
SubmitVideoCensorJobModel() {
  CosXmlConfig config = new CosXmlConfig.Builder()
    .SetRegion("COS_REGION") // 设置默认的区域, COS 地域的简称请参照 https://cloud
    .Build();

  string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
  string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
  long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
  QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
    secretKey, durationSecond);

  this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
}

/// 查询视频审核任务结果
public void GetVideoCensorJobResult(string JobId)
{
  // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
  string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
  GetVideoCensorJobRequest request = new GetVideoCensorJobRequest(bucket, Job
  // 执行请求
  GetVideoCensorJobResult result = cosXml.GetVideoCensorJob(request);
  Console.WriteLine(result.GetResultInfo());

  // 读取审核结果
  Console.WriteLine(result.resultStruct.JobsDetail.JobId);
  Console.WriteLine(result.resultStruct.JobsDetail.State);
  Console.WriteLine(result.resultStruct.JobsDetail.CreationTime);
  Console.WriteLine(result.resultStruct.JobsDetail.Object);
  Console.WriteLine(result.resultStruct.JobsDetail.SnapshotCount);
  Console.WriteLine(result.resultStruct.JobsDetail.Result);

  Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.HitFlag);
  Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.Count);

  Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.HitFlag);
  Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.Count);

  // 审核视频截图内容
  for(int i = 0; i < result.resultStruct.JobsDetail.Snapshot.Count; i++)
  {
      Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].Url);
      Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].Text);
      Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].SnapshotTi
      Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.H
      Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.S
```

```
            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.L
            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.S

            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.Hi
            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.Sc
            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.La
            Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.Su
        }
    }

    static void Main(string[] args)
    {
        SubmitVideoCensorJobModel m = new SubmitVideoCensorJobModel();
        // 输入在提交审核任务时获取的JobId
        string JobId = "xxx";
        /// 查询审核任务
        m.GetVideoCensorJobResult(JobId);
    }
  }
}
```

说明：

更多完整示例，请前往 GitHub 查看。

音频审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于音频审核的 API 概览和 SDK 示例代码。

API	操作描述
提交音频审核任务	用于提交一个音频审核任务
查询音频审核任务结果	用于查询指定的音频审核任务

提交音频审核任务

功能说明

本接口用于提交一个音频审核任务。音频审核功能为异步任务方式，您可以通过提交音频审核任务审核您的音频文件，然后通过查询音频审核任务接口查询审核结果。

下面示例展示了如何提交一个音频审核任务，然后通过 JobId 查询音频审核任务结果。

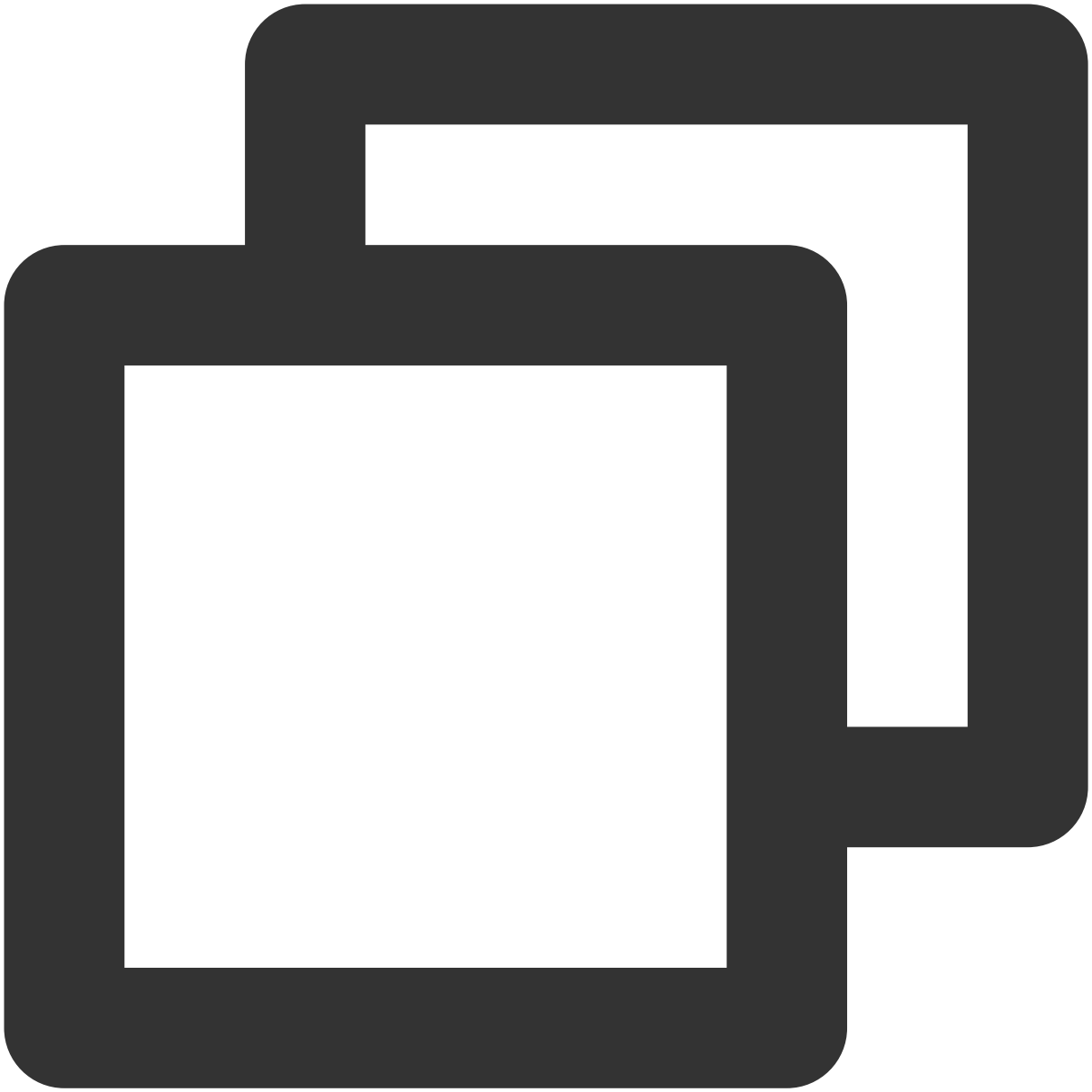
说明：

此操作需要存储桶开通数据万象相关功能。

音频审核接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本 Changelog 请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitAudioCensorJobModel {

      private CosXml cosXml;
```

```
    SubmitAudioCensorJobModel() {
      CosXmlConfig config = new CosXmlConfig.Builder()
        .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
        .Build();

      string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
      string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
      long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
      QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
        secretKey, durationSecond);

      this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
    }

    /// 提交音频审核任务
    public string SubmitAudioCensorJob()
    {
      // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
      string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
      SubmitAudioCensorJobRequest request = new SubmitAudioCensorJobRequest(bucke
      request.SetCensorObject("audio.mp3"); // 媒体文件的对象键, 需要替换成桶内存在的媒体
      // 审核的场景类型, 有效值：Porn（涉黄）、Ads（广告）, 可以传入多种类型, 不同类型以逗号分
      request.SetDetectType("Porn,Ads");
      // 执行请求
      SubmitCensorJobResult result = cosXml.SubmitAudioCensorJob(request);
      Console.WriteLine(result.GetResultInfo());
      Console.WriteLine(result.censorJobsResponse.JobsDetail.JobId);
      Console.WriteLine(result.censorJobsResponse.JobsDetail.State);
      Console.WriteLine(result.censorJobsResponse.JobsDetail.CreationTime);
      return result.censorJobsResponse.JobsDetail.JobId;
    }

    static void Main(string[] args)
    {
      SubmitAudioCensorJobModel m = new SubmitAudioCensorJobModel();
      /// 提交审核任务
      string JobId = m.SubmitAudioCensorJob();
      /// 打印唯一标识此次审核任务的 JobId
      Console.WriteLine("JobId : " + JobId);
    }
  }
}
```

说明：

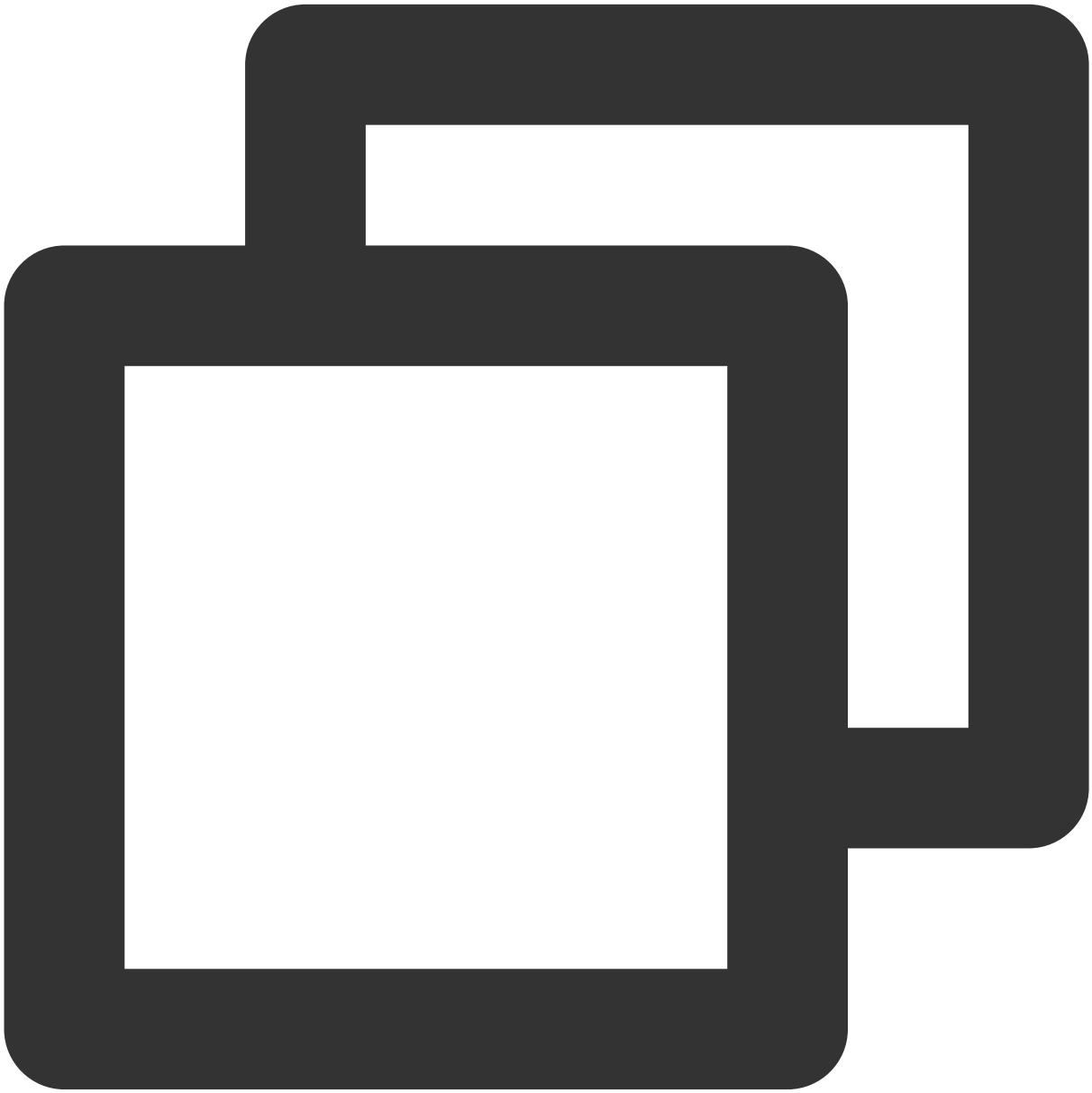
更多完整示例，请前往 GitHub 查看。

查询音频审核任务结果

功能说明

用于查询音频审核任务执行情况和结果。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
```

```
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitAudioCensorJobModel {

      private CosXml cosXml;

      SubmitAudioCensorJobModel() {
        CosXmlConfig config = new CosXmlConfig.Builder()
          .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
          .Build();

        string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
        string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
        long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
        QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
          secretKey, durationSecond);

        this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
      }

      /// 查询音频审核任务结果
      public void GetAudioCensorJobResult(string JobId)
      {
        // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
        string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
        GetAudioCensorJobRequest request = new GetAudioCensorJobRequest(bucket, Job
        // 执行请求
        GetAudioCensorJobResult result = cosXml.GetAudioCensorJob(request);
        Console.WriteLine(result.GetResultInfo());

        // 读取审核结果
        Console.WriteLine(result.resultStruct.JobsDetail.JobId);
        Console.WriteLine(result.resultStruct.JobsDetail.State);
        Console.WriteLine(result.resultStruct.JobsDetail.CreationTime);
        Console.WriteLine(result.resultStruct.JobsDetail.Object);
        Console.WriteLine(result.resultStruct.JobsDetail.Result);
        Console.WriteLine(result.resultStruct.JobsDetail.AudioText);

        Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.HitFlag);
        Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.Score);
        Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.Label);

        Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.HitFlag);
        Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.Score);
```

```
            Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.Label);

            // 音频节选Section信息
            for(int i = 0; i < result.resultStruct.JobsDetail.Section.Count; i++)
            {
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].Url);
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].OffsetTime)
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].Duration);
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].Text);
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Hi
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Sc

                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].AdsInfo.Hit
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].AdsInfo.Sco
            }
        }

        static void Main(string[] args)
        {
            SubmitAudioCensorJobModel m = new SubmitAudioCensorJobModel();
            /// 输入提交审核任务时获取的 JobId
            string JobId = "xxx";
            /// 查询审核任务结果
            m.GetAudioCensorJobResult(JobId);
        }
    }
}
```

说明：

更多完整示例，请前往 GitHub 查看。

文档审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文档审核的 API 概览和 SDK 示例代码。

API	操作描述
提交文档审核任务	用于提交一个文档审核任务
查询文档审核任务结果	用于查询指定的文档审核任务

提交文档审核任务

功能说明

本接口用于提交一个文档审核任务，可审核您的文档文件是否存在敏感违规信息。文档审核结合了对象存储（Cloud Object Storage，COS）文档预览功能，通过预先将文档转成图片，结合图片内容审核、图片 OCR 审核等方式，进行文档审核。

下面示例展示了如何提交一个文档审核任务，然后何通过 JobId 查询文档审核任务结果。

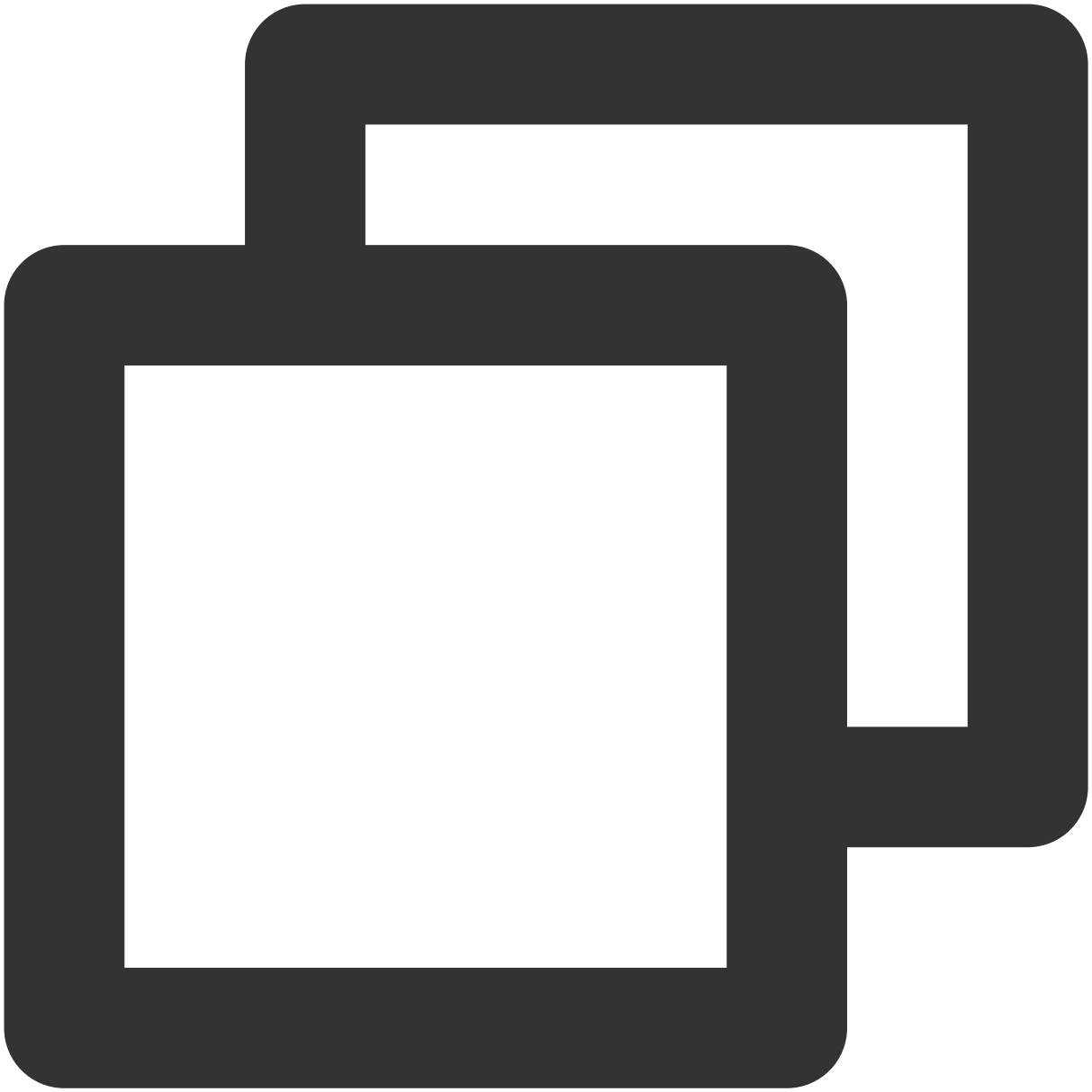
说明：

此操作需要存储桶开通数据万象相关功能。

文档审核接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本 Changelog 请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitDocumentCensorJobModel {

      private CosXml cosXml;
```

```
        SubmitDocumentCensorJobModel() {
          CosXmlConfig config = new CosXmlConfig.Builder()
            .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
            .Build();

          string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
          string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
          long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
          QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
            secretKey, durationSecond);

          this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
        }

        /// 提交文档审核任务
        public string SubmitDocumentCensorJob()
        {
          // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
          string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
          SubmitDocumentCensorJobRequest request = new SubmitDocumentCensorJobRequest
          request.SetUrl("url"); // 审核文档的URL, 需要替换成具体需要审核的文档URL
          // 审核的场景类型, 有效值：Porn（涉黄）、Ads（广告）, 可以传入多种类型, 不同类型以逗号分
          request.SetDetectType("Porn,Ads");
          // 执行请求
          SubmitCensorJobResult result = cosXml.SubmitDocumentCensorJob(request);
          Console.WriteLine(result.GetResultInfo());
          Console.WriteLine(result.censorJobsResponse.JobsDetail.JobId);
          Console.WriteLine(result.censorJobsResponse.JobsDetail.State);
          Console.WriteLine(result.censorJobsResponse.JobsDetail.CreationTime);
          return result.censorJobsResponse.JobsDetail.JobId;
        }

        static void Main(string[] args)
        {
          SubmitDocumentCensorJobModel m = new SubmitDocumentCensorJobModel();
          /// 提交审核任务
          string JobId = m.SubmitDocumentCensorJob();
        }
      }
    }
```

说明：

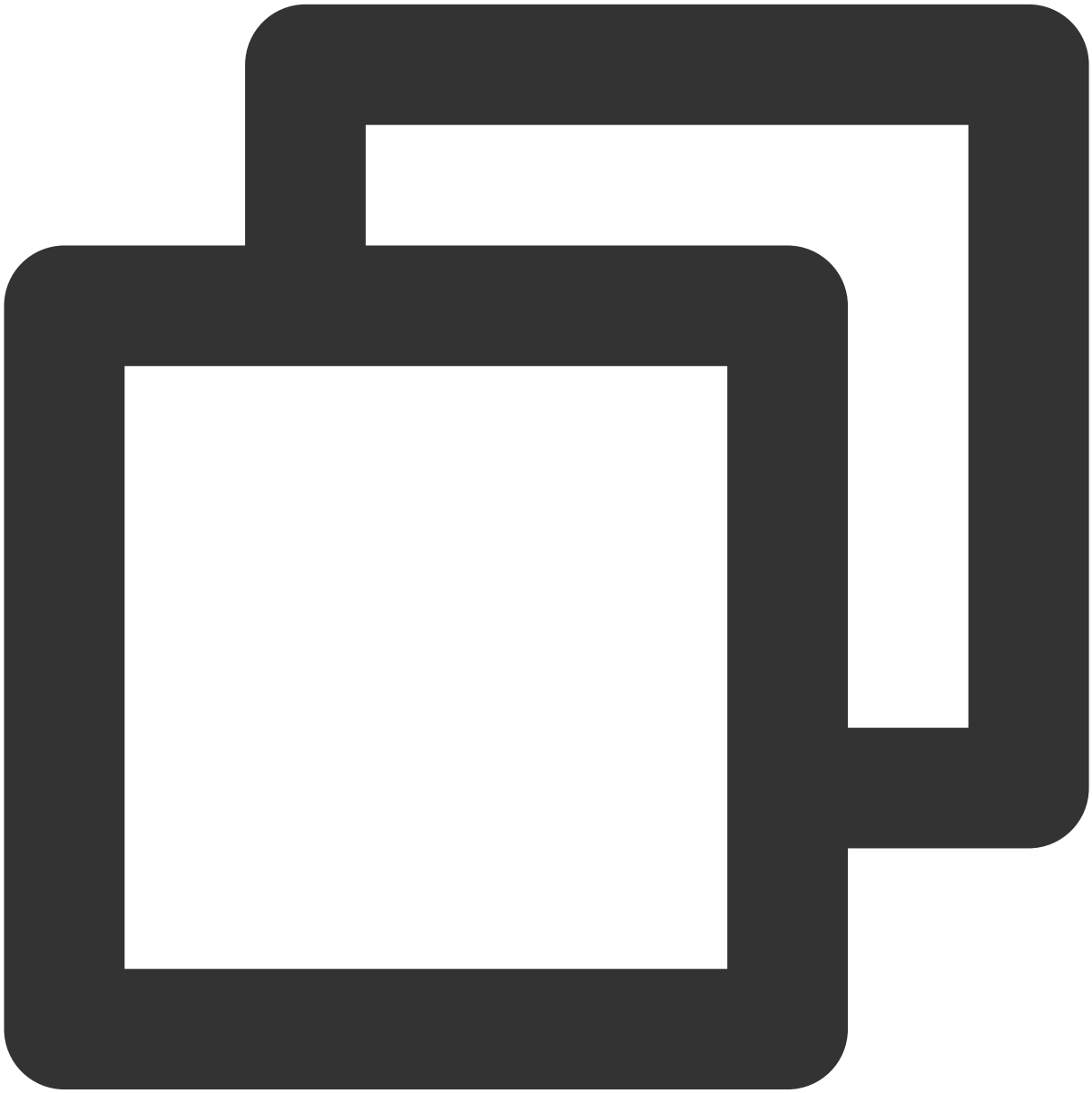
更多完整示例，请前往 GitHub 查看。

查询文档审核任务

功能说明

用于查询文档审核任务执行情况和结果。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
```

```
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitDocumentCensorJobModel {

      private CosXml cosXml;

      SubmitDocumentCensorJobModel() {
        CosXmlConfig config = new CosXmlConfig.Builder()
          .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
          .Build();

        string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
        string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
        long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
        QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
          secretKey, durationSecond);

        this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
      }

      /// 查询文档审核任务结果
      public void GetDocumentCensorJobResult(string JobId)
      {
        // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
        string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
        GetDocumentCensorJobRequest request = new GetDocumentCensorJobRequest(bucke
        // 执行请求
        GetDocumentCensorJobResult result = cosXml.GetDocumentCensorJob(request);
        Console.WriteLine(result.GetResultInfo());

        // 读取审核结果
        Console.WriteLine(result.resultStruct.JobsDetail.State);
        Console.WriteLine(result.resultStruct.JobsDetail.JobId);
        Console.WriteLine(result.resultStruct.JobsDetail.Suggestion);
        Console.WriteLine(result.resultStruct.JobsDetail.CreationTime);
        Console.WriteLine(result.resultStruct.JobsDetail.Url);
        Console.WriteLine(result.resultStruct.JobsDetail.PageCount);
        Console.WriteLine(result.resultStruct.JobsDetail.Labels);
        Console.WriteLine(result.resultStruct.JobsDetail.Labels.PornInfo.HitFlag);
        Console.WriteLine(result.resultStruct.JobsDetail.Labels.PornInfo.Score);
        Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.Url);
        Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.Text);
        Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PageNu
        Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornIn
```

```
            Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornIn
            Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornIn
        }

        static void Main(string[] args)
        {
            SubmitDocumentCensorJobModel m = new SubmitDocumentCensorJobModel();
            /// 输入提交审核任务时返回的, 唯一标识此次审核任务的 JobId
            string JobId = "xxx";
            /// 查询审核任务结果
            m.GetDocumentCensorJobResult(JobId);
        }
    }
}
```

说明：

更多完整示例，请前往 GitHub 查看。

文本审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文本审核的 API 概览和 SDK 示例代码。

API	操作描述
提交文本审核任务	用于提交一个文本审核任务
查询文本审核任务结果	用于查询指定的文本审核任务

提交文本审核任务

功能说明

本接口用于提交一个文本审核任务。您可以通过提交文本审核任务审核您的文本文件，然后通过查询文本审核任务接口查询审核结果。

下面示例展示了如何提交一个文本审核任务，然后通过 JobId 查询文本审核任务结果。

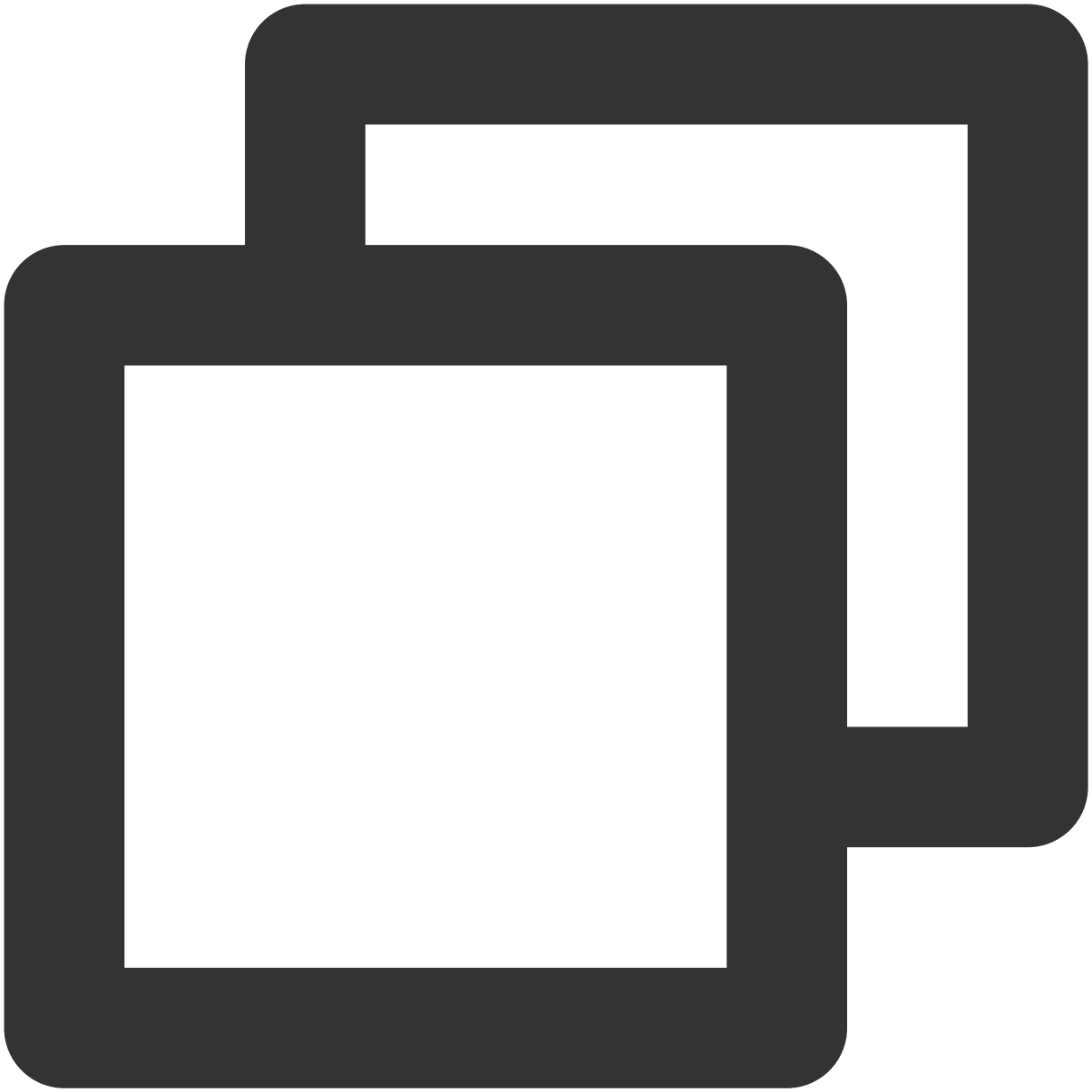
说明：

此操作需要存储桶开通数据万象相关功能。

文本审核接口从 5.4.24 版本开始支持，下载新版 SDK 前往 Releases 或参见 快速入门。

查看版本 Changelog 请前往 GitHub。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitTextCensorJobModel {

      private CosXml cosXml;
```

```
        SubmitTextCensorJobModel() {
          CosXmlConfig config = new CosXmlConfig.Builder()
            .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
            .Build();

          string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
          string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
          long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
          QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
            secretKey, durationSecond);

          this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
        }

        /// 提交文本审核任务
        public string SubmitTextCensorJob()
        {
          // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
          string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
           SubmitTextCensorJobRequest request = new SubmitTextCensorJobRequest(bucket
          request.SetCensorObject("text.txt"); // 媒体文件的对象键, 需要替换成桶内存在的媒体
          // 审核的场景类型, 有效值：Porn（涉黄）、Ads（广告）, 可以传入多种类型, 不同类型以逗号分
          request.SetDetectType("Porn,Ads");
          // 执行请求
          SubmitCensorJobResult result = cosXml.SubmitTextCensorJob(request);
          Console.WriteLine(result.GetResultInfo());
          Console.WriteLine(result.censorJobsResponse.JobsDetail.JobId);
          Console.WriteLine(result.censorJobsResponse.JobsDetail.State);
          Console.WriteLine(result.censorJobsResponse.JobsDetail.CreationTime);
          return result.censorJobsResponse.JobsDetail.JobId;
          //.cssg-snippet-body-end
        }

        static void Main(string[] args)
        {
          SubmitTextCensorJobModel m = new SubmitTextCensorJobModel();
          /// 提交审核任务
          string JobId = m.SubmitTextCensorJob();
          /// 打印唯一标识了此次审核任务的 JobId
          Console.WriteLine("JobId : " + JobId);
        }
      }
    }
```

说明：

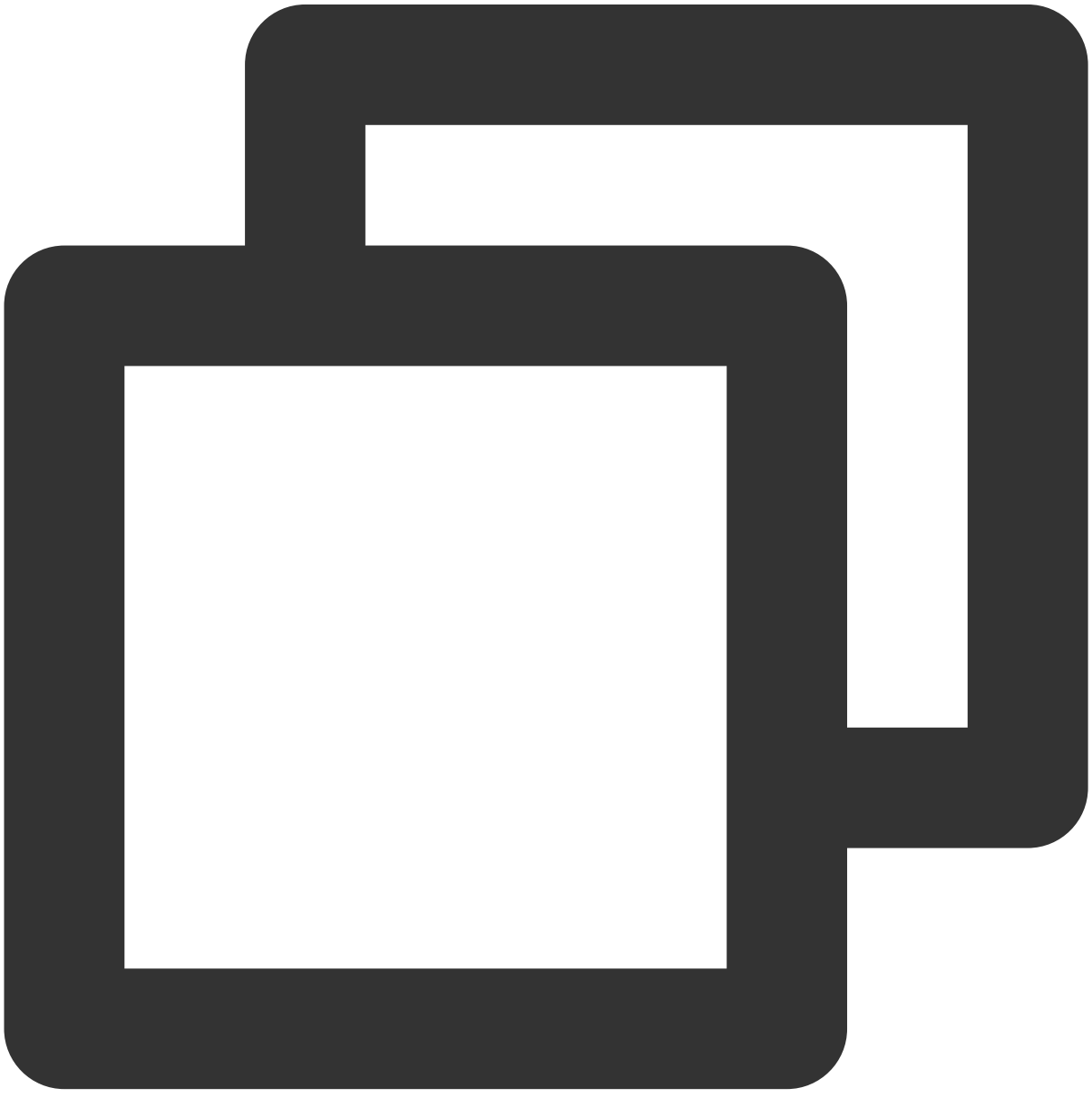
更多完整示例，请前往 GitHub 查看。

查询文本审核任务

功能说明

用于查询文本审核任务执行情况和结果。

示例代码



```
using COSXML.Model.CI;
using COSXML.Auth;
using System;
using System.Threading;
using COSXML;

namespace COSSnippet
{
    public class SubmitTextCensorJobModel {

      private CosXml cosXml;

      SubmitTextCensorJobModel() {
        CosXmlConfig config = new CosXmlConfig.Builder()
          .SetRegion("COS_REGION") // 设置默认的地域, COS 地域的简称请参照 https://cloud
          .Build();

        string secretId = "SECRET_ID";    // 云 API 密钥 SecretId, 获取 API 密钥请参照 h
        string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照
        long durationSecond = 600;          //每次请求签名有效时长, 单位为秒
        QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCreden
          secretKey, durationSecond);

        this.cosXml = new CosXmlServer(config, qCloudCredentialProvider);
      }

      /// 查询文本审核任务结果
      public void GetTextCensorJobResult(string JobId)
      {
        // 存储桶名称, 此处填入格式必须为 bucketname-APPID, 其中 APPID 获取参考 https://co
        string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容
        GetTextCensorJobRequest request = new GetTextCensorJobRequest(bucket, JobId
        // 执行请求
        GetTextCensorJobResult result = cosXml.GetTextCensorJob(request);
        Console.WriteLine(result.GetResultInfo());

        // 读取审核结果
        Console.WriteLine(result.resultStruct.JobsDetail.JobId);
        Console.WriteLine(result.resultStruct.JobsDetail.State);
        Console.WriteLine(result.resultStruct.JobsDetail.CreationTime);
        Console.WriteLine(result.resultStruct.JobsDetail.Object);
        Console.WriteLine(result.resultStruct.JobsDetail.SectionCount);
        Console.WriteLine(result.resultStruct.JobsDetail.Result);

        Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.HitFlag);
        Console.WriteLine(result.resultStruct.JobsDetail.PornInfo.Count);
```

```
            Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.HitFlag);
            Console.WriteLine(result.resultStruct.JobsDetail.AdsInfo.Count);

            // 文本节选Section信息
            for(int i = 0; i < result.resultStruct.JobsDetail.Section.Count; i++)
            {
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].StartByte);
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Hi
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Sc
                Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Ke
            }
        }

        static void Main(string[] args)
        {
            SubmitTextCensorJobModel m = new SubmitTextCensorJobModel();
            // 输入提交文本审核任务时获取到的 JobId
            string JobId = "xxx";
            // 查询审核任务结果
            m.GetTextCensorJobResult(JobId);
        }
    }
}
```

说明：

更多完整示例，请前往 GitHub 查看。

内容识别
图片二维码

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于图片二维码相关的 API 概览以及 SDK 示例代码。

API	操作描述
二维码识别	二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克

二维码识别

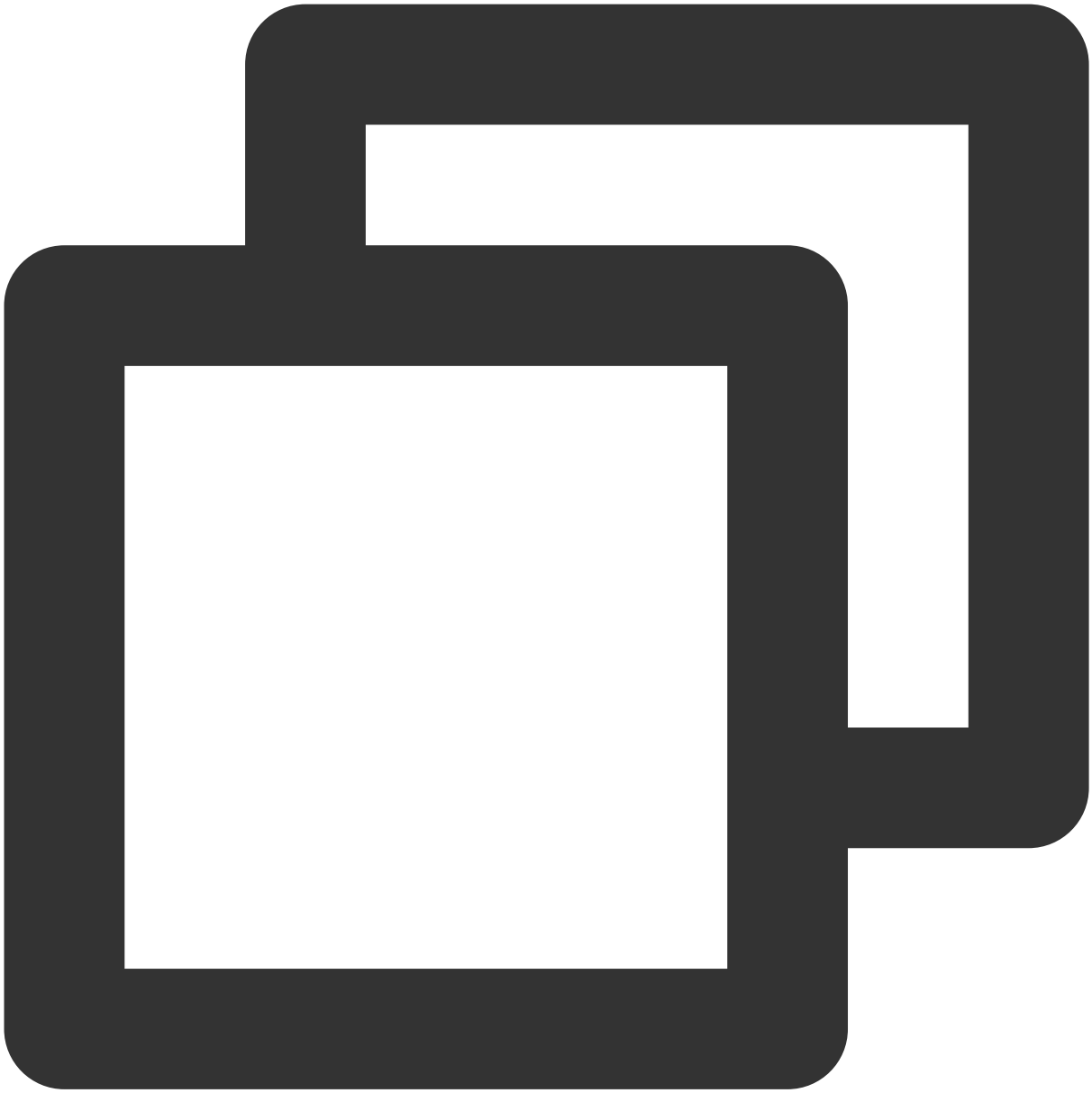
二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克。

上传时识别二维码

功能说明

二维码识别功能支持在文件上传时进行识别。

示例代码



```
PutObjectRequest request = new PutObjectRequest(bucket, key, srcPath);

JObject o = new JObject();
// 不返回原图
o["is_pic_info"] = 1;
JArray rules = new JArray();
JObject rule = new JObject();
rule["bucket"] = bucket;
rule["fileid"] = "qrcode.jpg";
//处理参数，规则参见：https://cloud.tencent.com/document/product/436/54070
rule["rule"] = "QRcode/cover/<mode>";
```

```
rules.Add(rule);
o["rules"] = rules;

string ruleString = o.ToString(Formatting.None);
request.SetRequestHeader("Pic-Operations", ruleString);
//执行请求
PutObjectResult result = cosXml.PutObject(request);
```

说明：

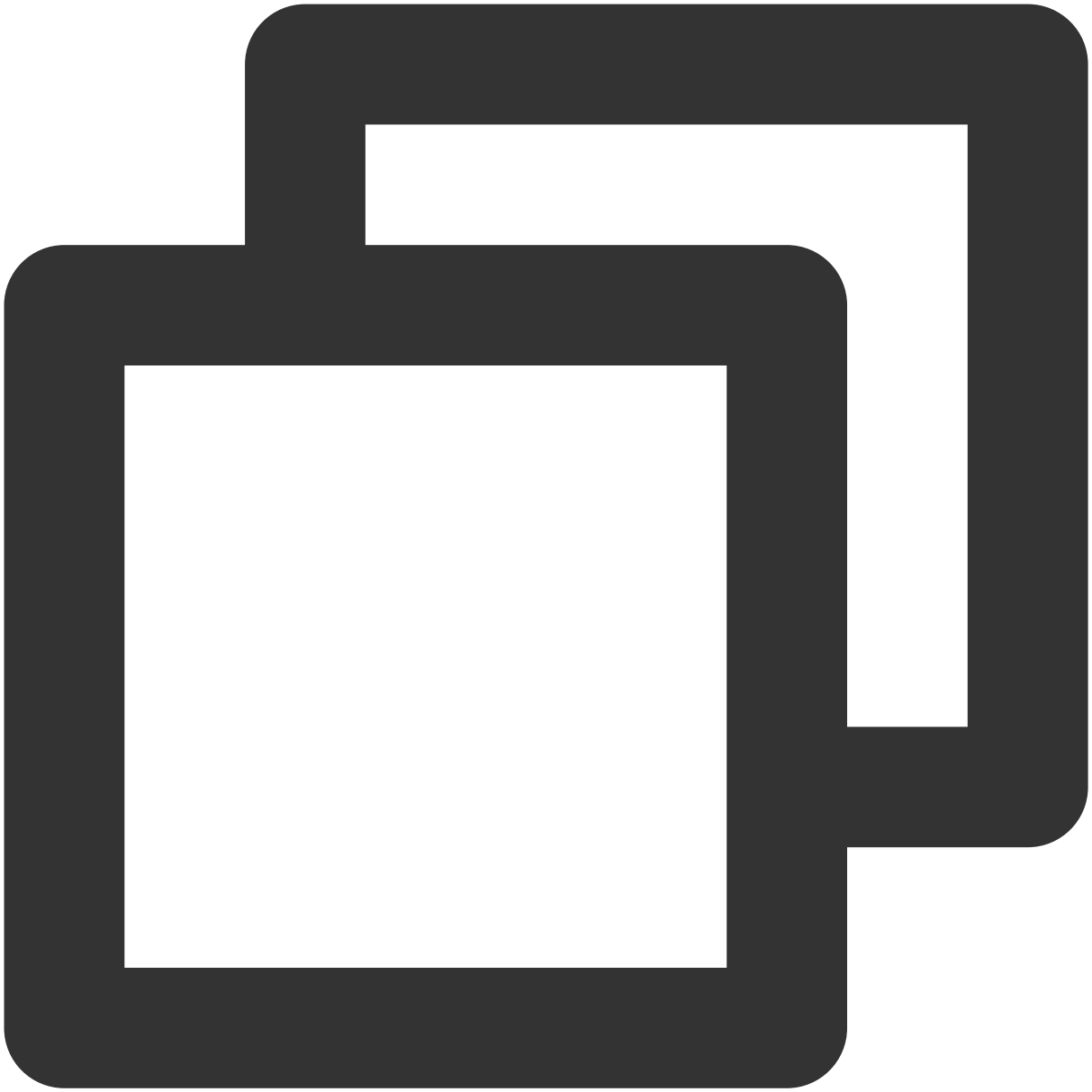
更多完整示例，请前往 GitHub 查看。

下载时识别二维码

功能说明

二维码识别功能可支持在下载时进行识别。

示例代码



```
//二维码覆盖功能，将对识别出的二维码覆盖上马赛克。取值为0或1。0表示不开启二维码覆盖，1表示开启二维
QRCodeRecognitionRequest request = new QRCodeRecognitionRequest(bucket, key, 0);

QRCodeRecognitionResult result = cosXml.QRCodeRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

Go SDK
快速入门

最近更新时间：2024-03-04 16:55:55

下载与安装

相关资源

对象存储 COS 的 XML Go SDK 源码下载地址：XML Go SDK。
示例 Demo 下载地址：COS XML Go SDK 示例。
更多信息请参见 COS Go SDK API 文档。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：Go SDK 常见问题。

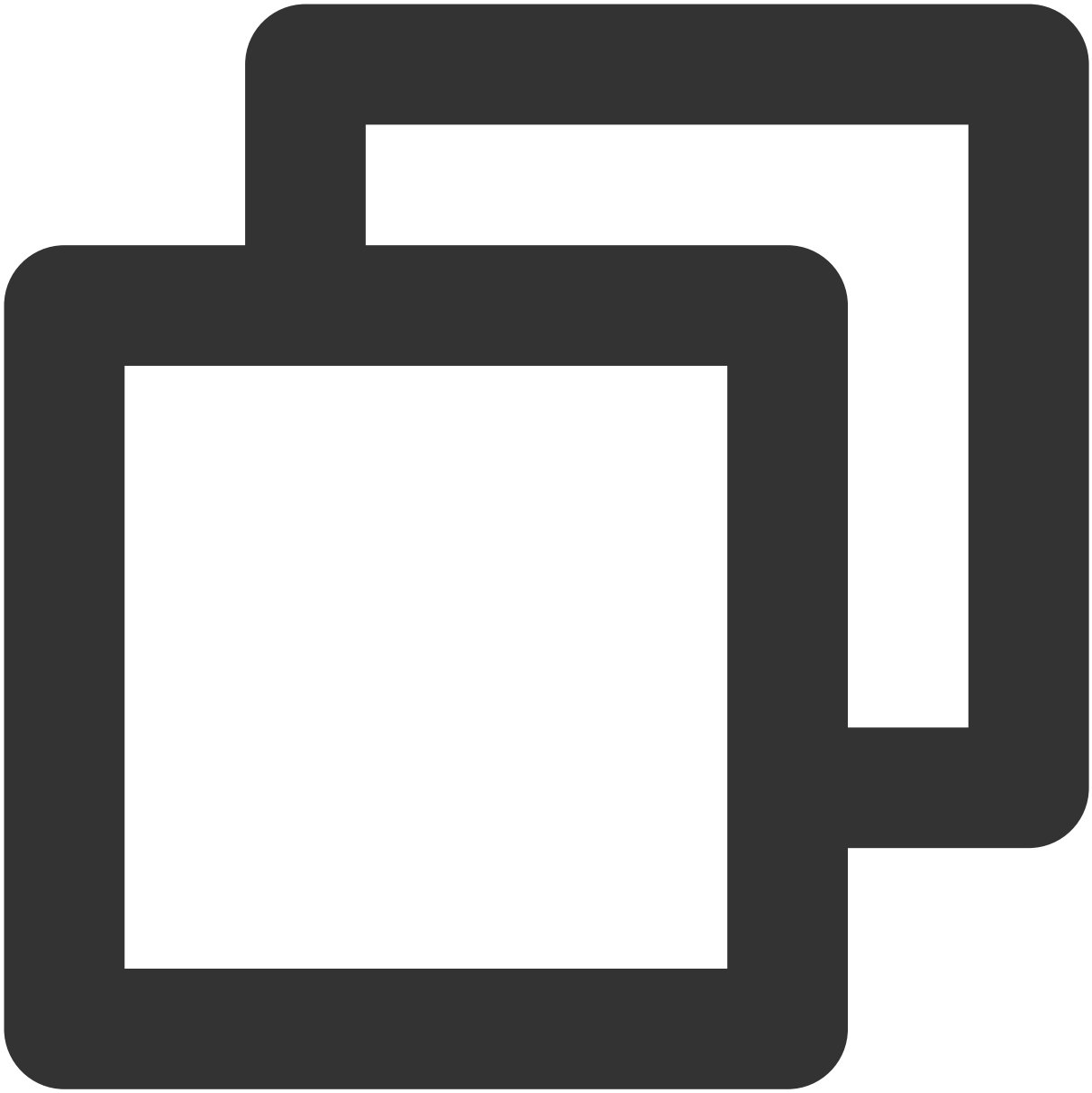
说明：
如果您在使用 SDK 时遇到函数或方法不存在等错误，请先将 SDK 升级到最新版再重试。

环境依赖

Golang：用于下载和安装 Go 编译运行环境，请前往 Golang 官网进行下载。

安装 SDK

执行以下命令安装 COS Go SDK：



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

开始使用

下面为您介绍如何使用 COS Go SDK 完成一个基础操作，如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。

初始化

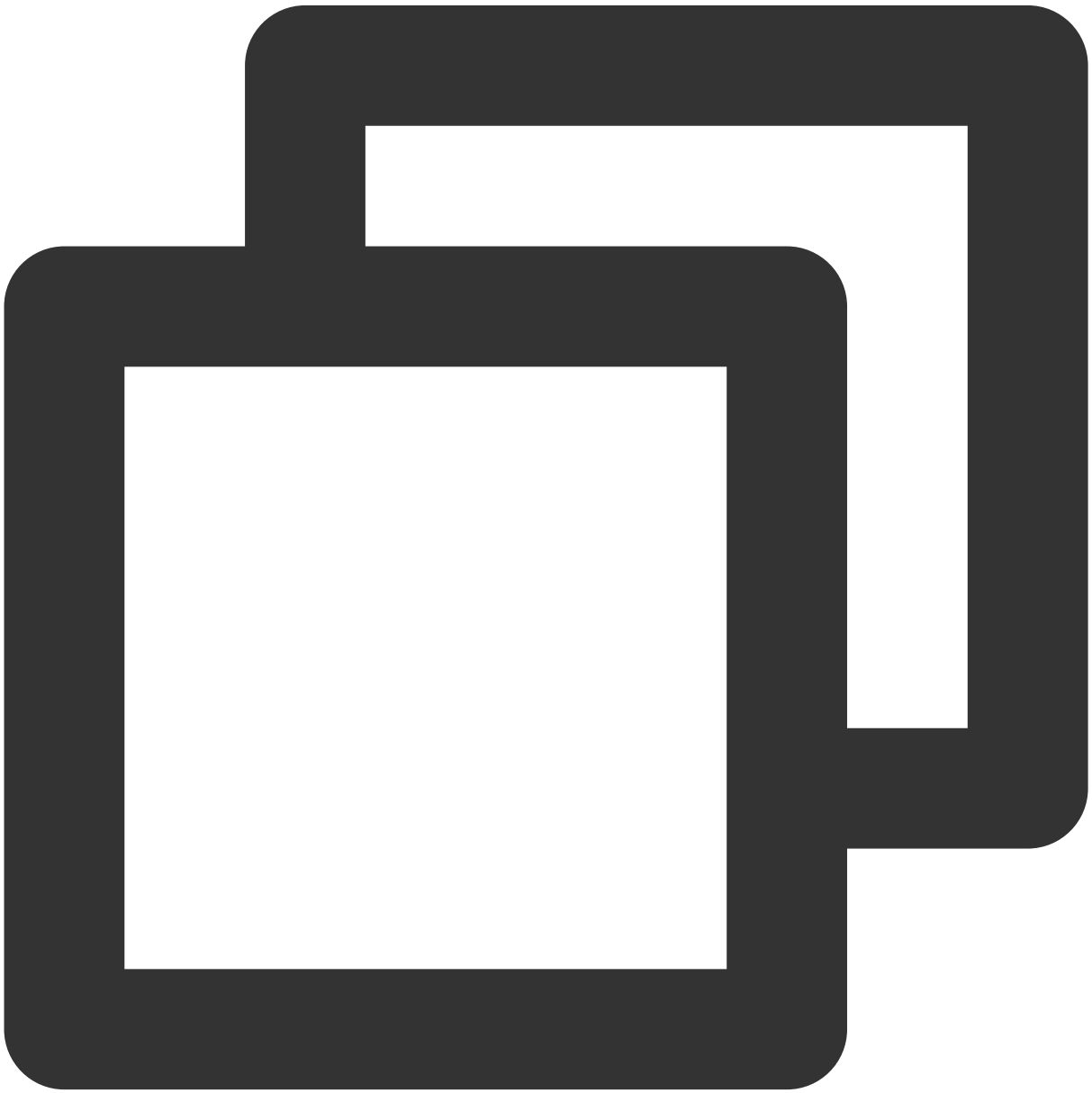
注意：

建议用户使用子账号密钥 + 环境变量的方式调用 SDK，提高 SDK 使用的安全性。为子账号授权时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

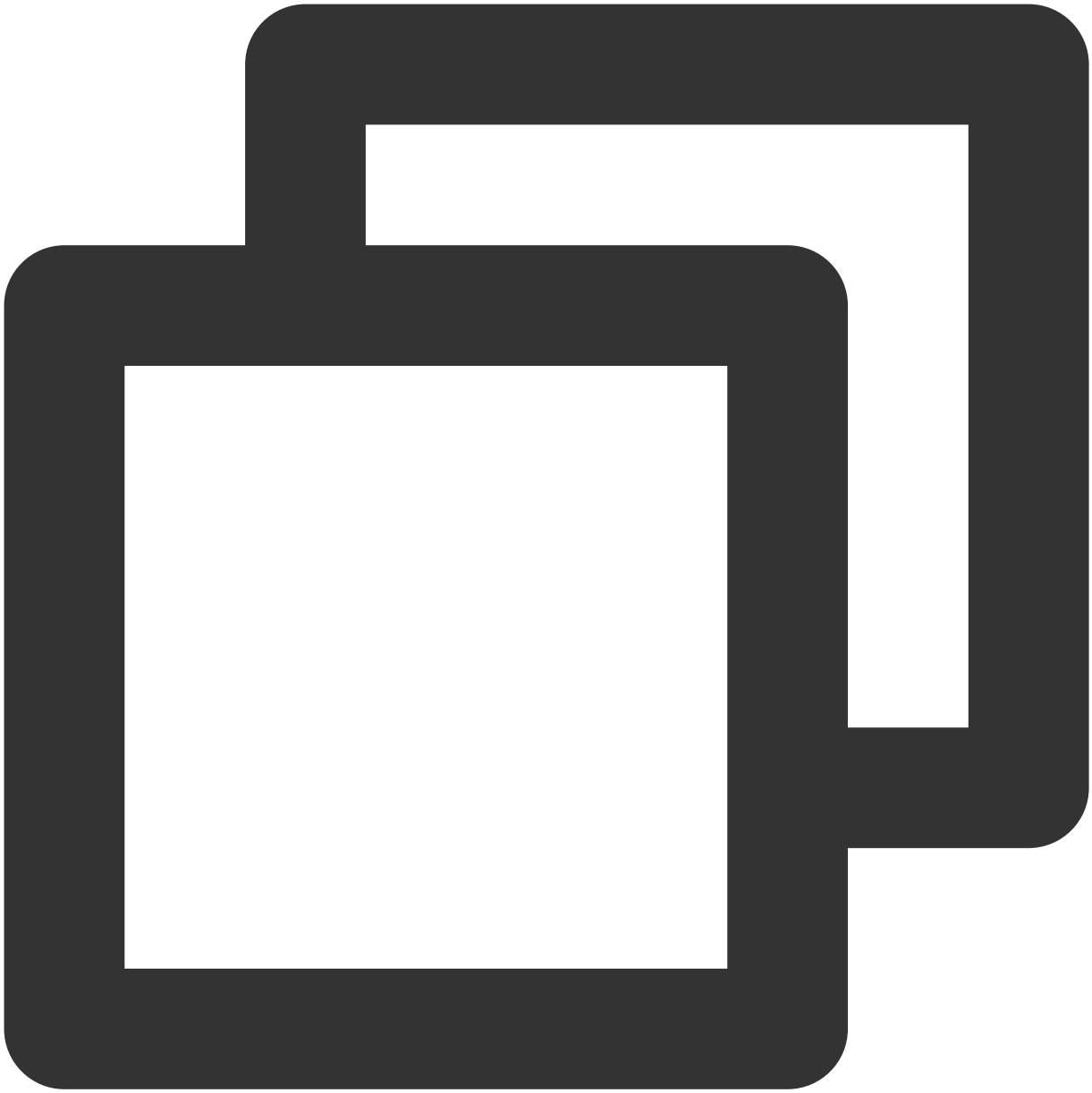
使用 COS 域名生成 COS GO 客户端 Client 结构。

方法原型



```
func NewClient(uri *BaseURL, httpClient *http.Client) *Client
```

参数说明

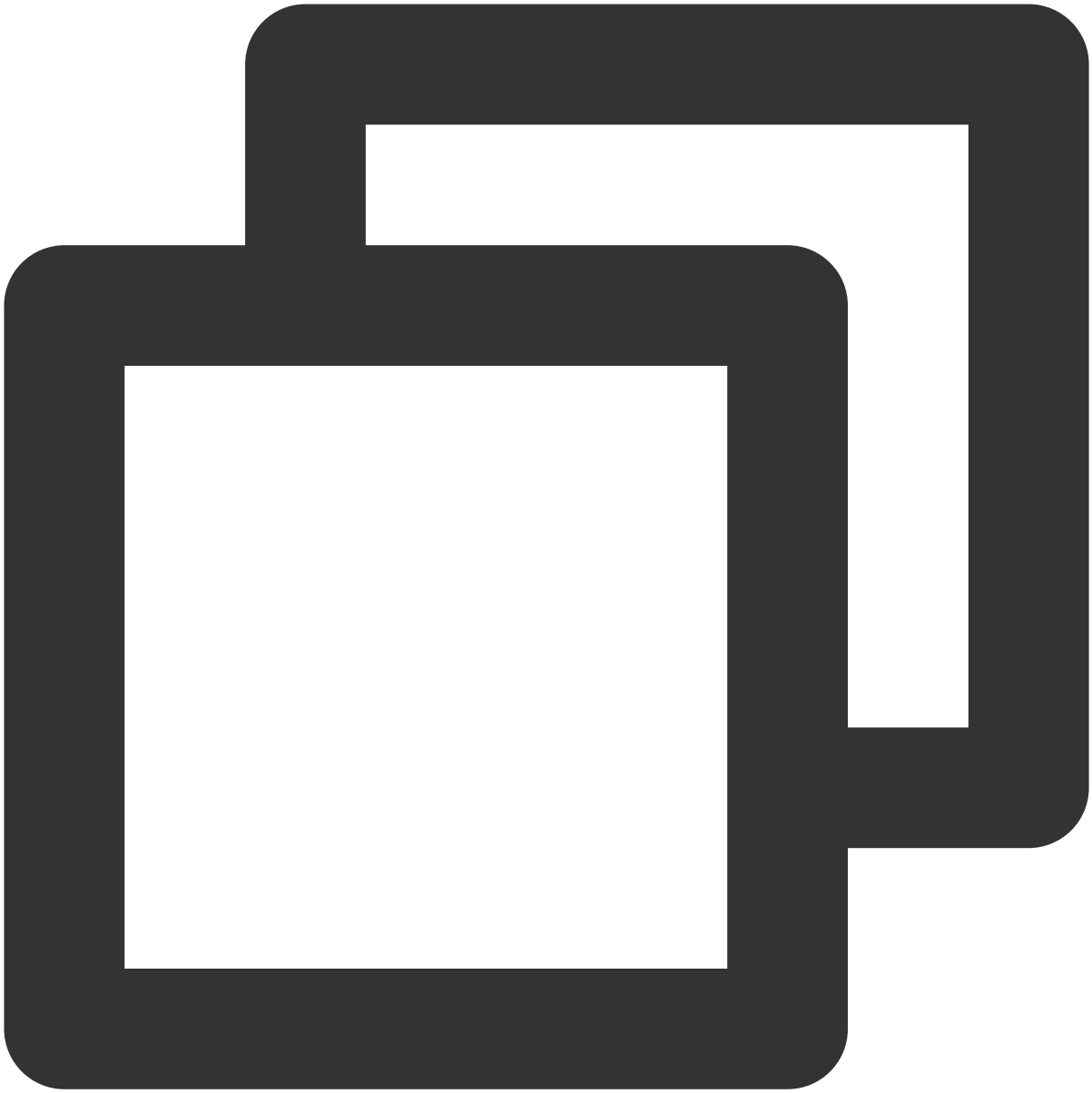


```
// BaseURL 访问各 API 所需的基础 URL
type BaseURL struct {
    // 访问 bucket, object 相关 API 的基础 URL（不包含 path 部分）: https://examplebucke
    BucketURL *url.URL
    // 访问 service API 的基础 URL（不包含 path 部分）: https://cos.<Region>.myqcloud.c
    ServiceURL *url.URL
    // 访问 Batch API 的基础 URL （不包含 path 部分）: https://<UIN>.cos-control.<Regio
    BatchURL *url.URL
    // 访问 CI 的基础 URL （不包含 path 部分）: https://examplebucket-1250000000.ci.<Re
```

```
    CIURL *url.URL
}
```

参数名称	参数描述	类型	是否必填
BucketURL	访问 bucket, object 相关 API 的基础 URL（不包含 path 部分）	string	是
ServiceURL	访问 service API 的基础 URL（不包含 path 部分）	string	否
BatchURL	访问 Batch API 的基础 URL （不包含 path 部分）	string	否
CIURL	访问 CI 的基础 URL （不包含 path 部分）	string	否

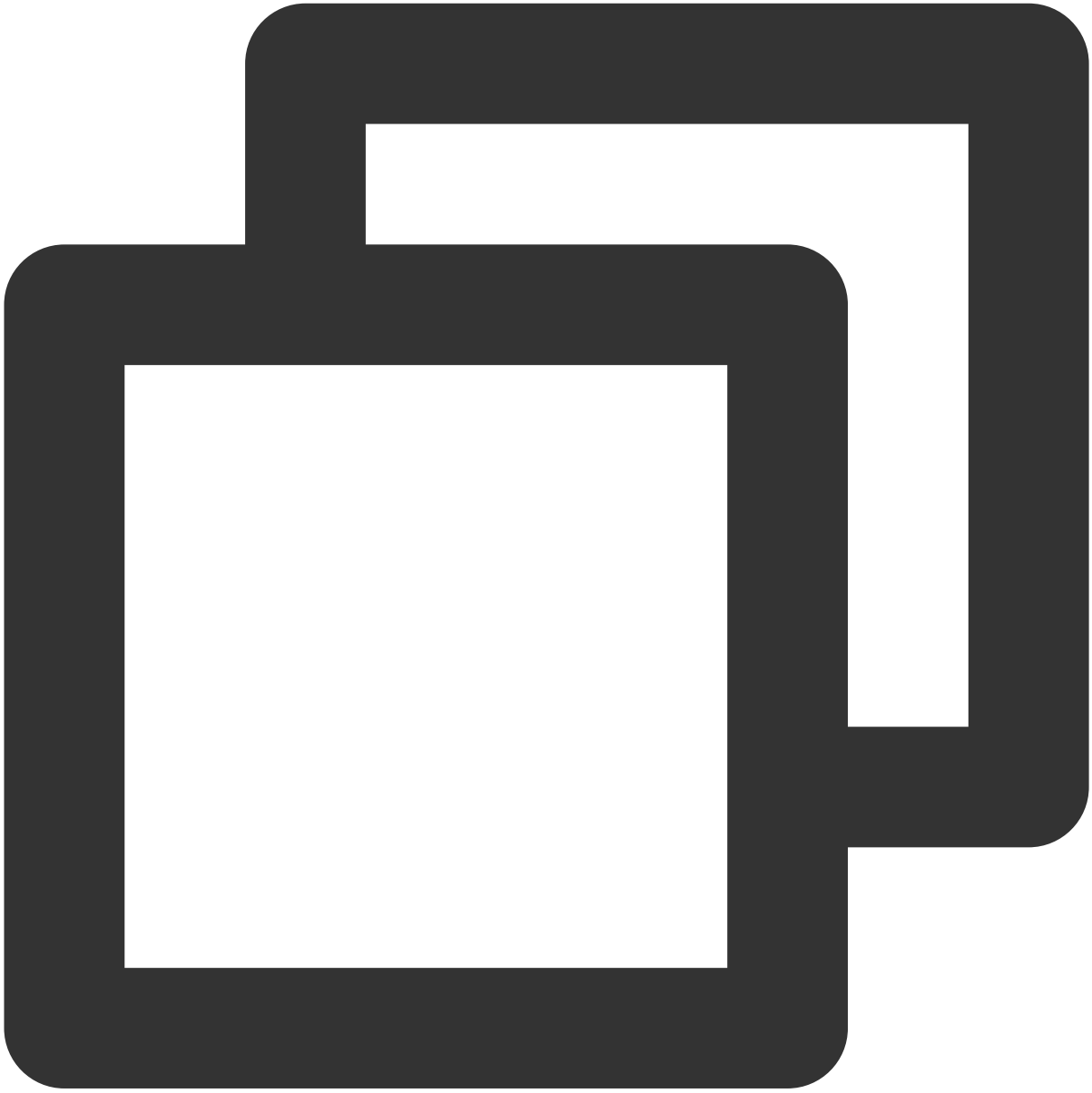
请求示例1：使用永久密钥



```
// 将 examplebucket-1250000000 和 COS_REGION 修改为用户真实的信息
// 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。htt
// COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地域的
u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
// 用于 Get Service 查询，默认全地域 service.cos.myqcloud.com
su, _ := url.Parse("https://cos.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: u, ServiceURL: su}
// 1.永久密钥
client := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，授权遵
```

```
        SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，授权
    },
})
```

请求示例2：使用临时密钥



```
// 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
// 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。htt
// COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地域的
u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
```

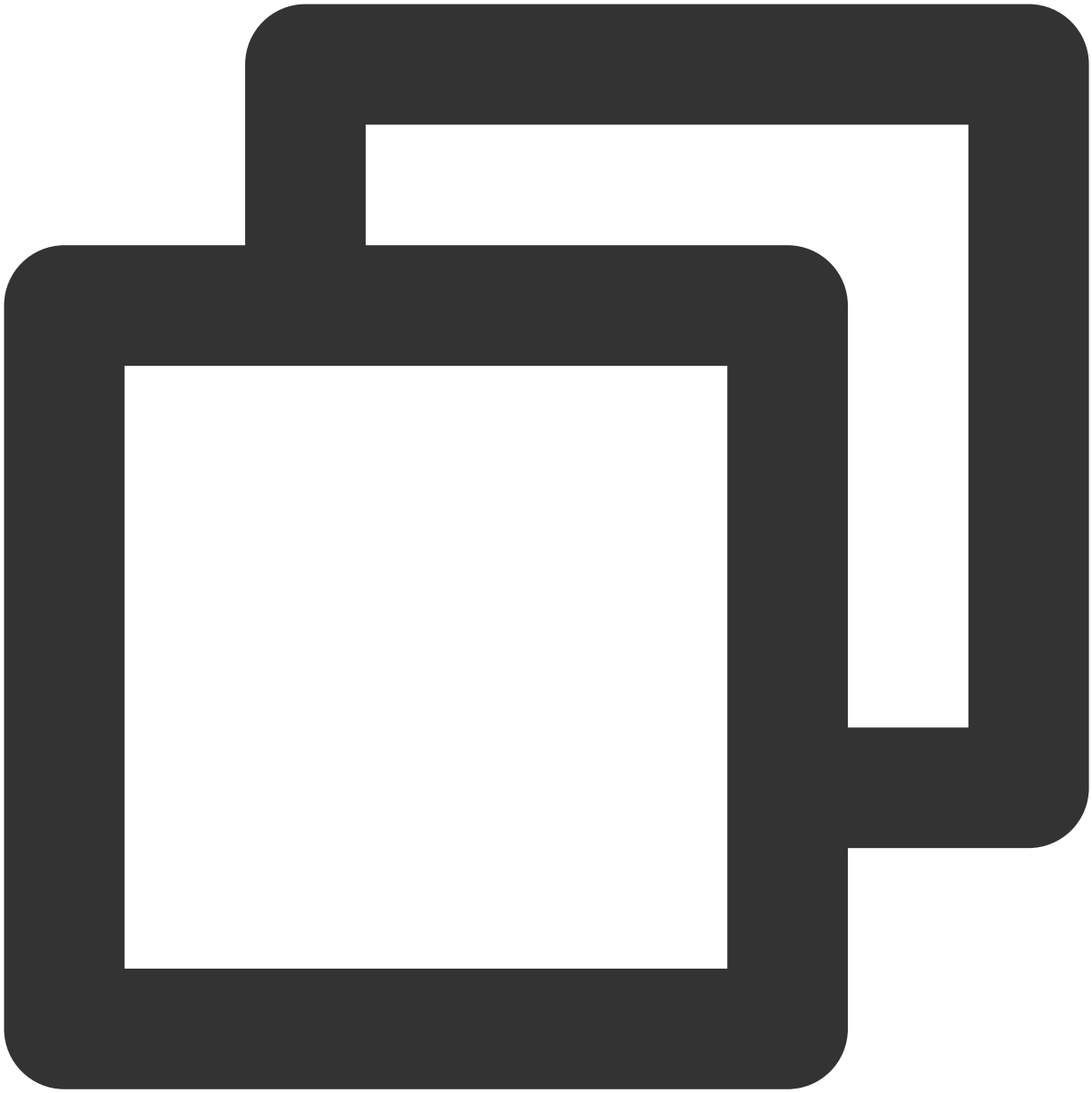
```
// 2.临时密钥
client := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        // 如果使用临时密钥需要填入，临时密钥生成和使用指引参见 https://www.tencentcloud.com
        SecretID:     "SECRETID",
        SecretKey:    "SECRETKEY",
        SessionToken: "SECRETTOKEN",
    },
})
if client != nil {
    // 调用 COS 请求
}
```

说明：

临时密钥生成和使用可参见 临时密钥生成及使用指引。

请求示例3：设置域名

通过修改 BaseURL，可以直接使用自定义域名或者全球加速域名访问 COS。



```
// 使用全球加速域名访问cos
u, _ := url.Parse("http://<BucketName-APPID>.cos.accelerate.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
// 2.临时密钥
client := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        // 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://www.tencentcloud.com
        SecretID:     "SECRETID",
        SecretKey:    "SECRETKEY",
        SessionToken: "SECRETTOKEN",
    },
```

```
})
```

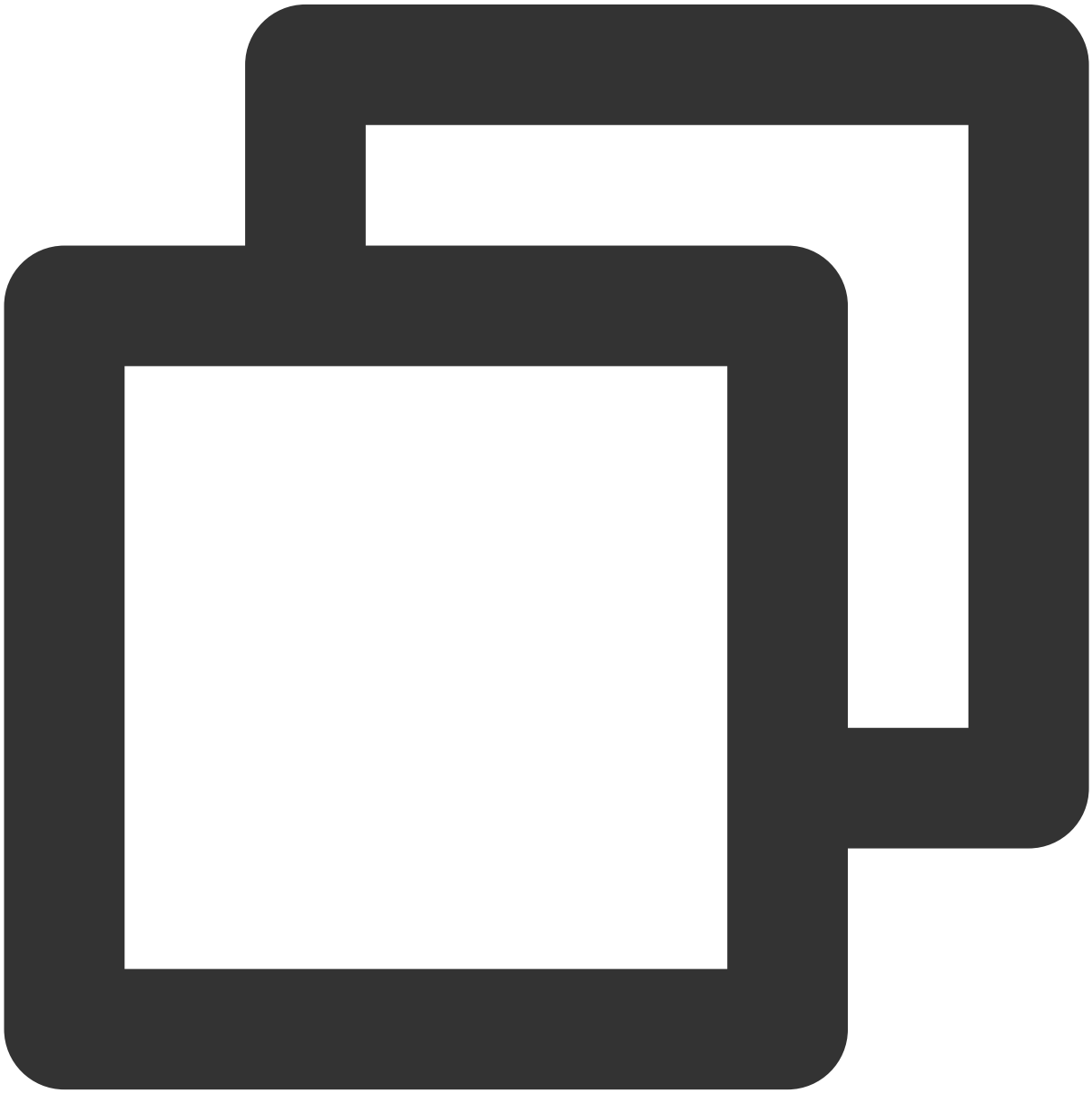
请求示例4：CRC64校验

COS Go SDK 默认开启了文件上传 CRC64 检验。

注意：

COS Go SDK 版本需要大于等于 v0.7.23。

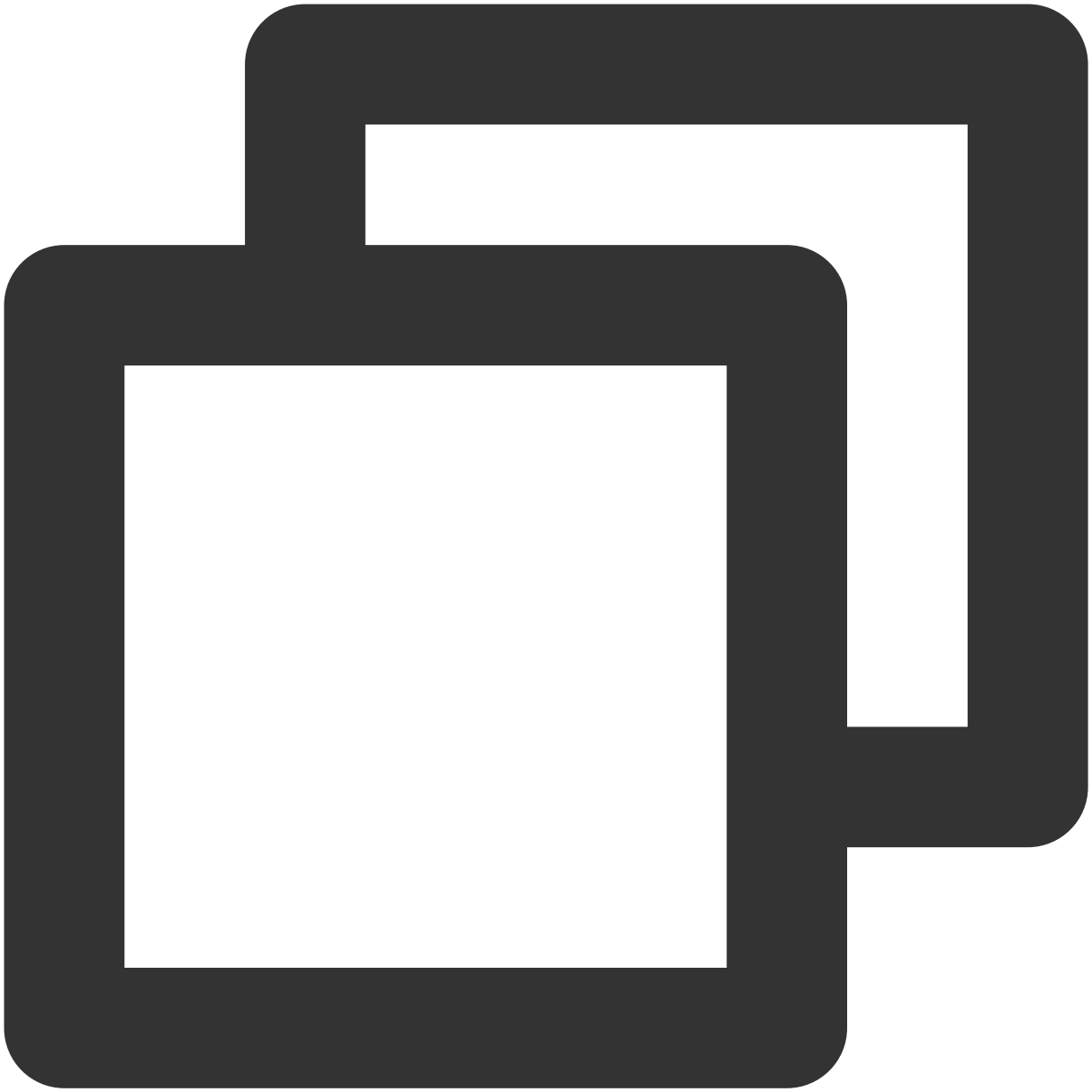
强烈建议用户不关闭 CRC64 校验。



```
// 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
// 存储桶名称，由 bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。https:
```

```
// COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地域的
u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
// 2.临时密钥
client := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        // 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://www.tencentcloud.com
        SecretID:     "SECRETID",
        SecretKey:    "SECRETKEY",
        SessionToken: "SECRETTOKEN",
    },
})
// 关闭 CRC64 校验
client.Conf.EnableCRC = false
```

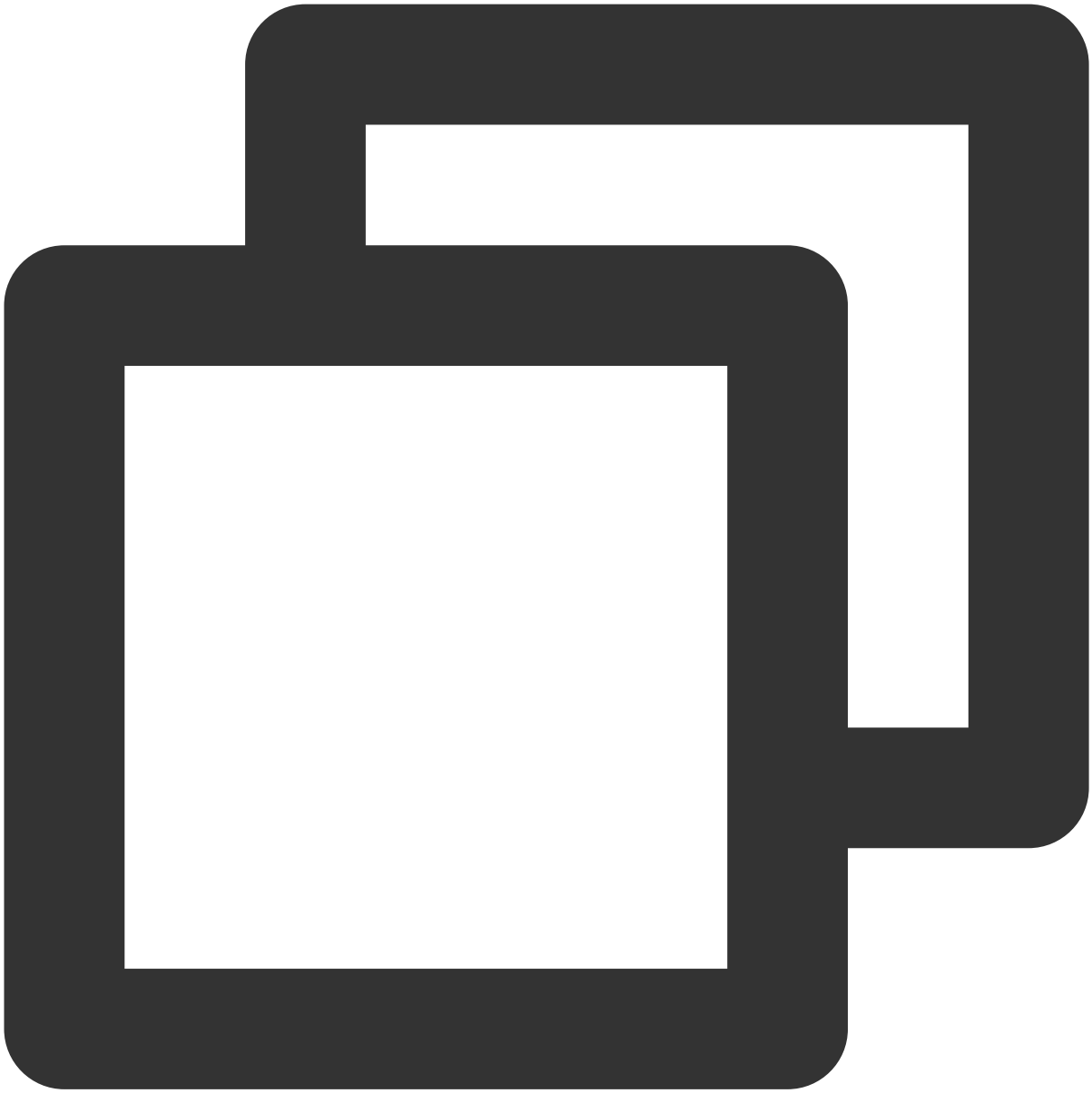
创建存储桶



```
package main


import (
    "context"
    "net/http"
    "net/url"
    "os"


    "github.com/tencentyun/cos-go-sdk-v5"
```

```
)


func main() {
    // 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
    // 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。
    // COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地
    u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，授
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })


    _, err := c.Bucket.Put(context.Background(), nil)
    if err != nil {
        panic(err)
    }
}
```

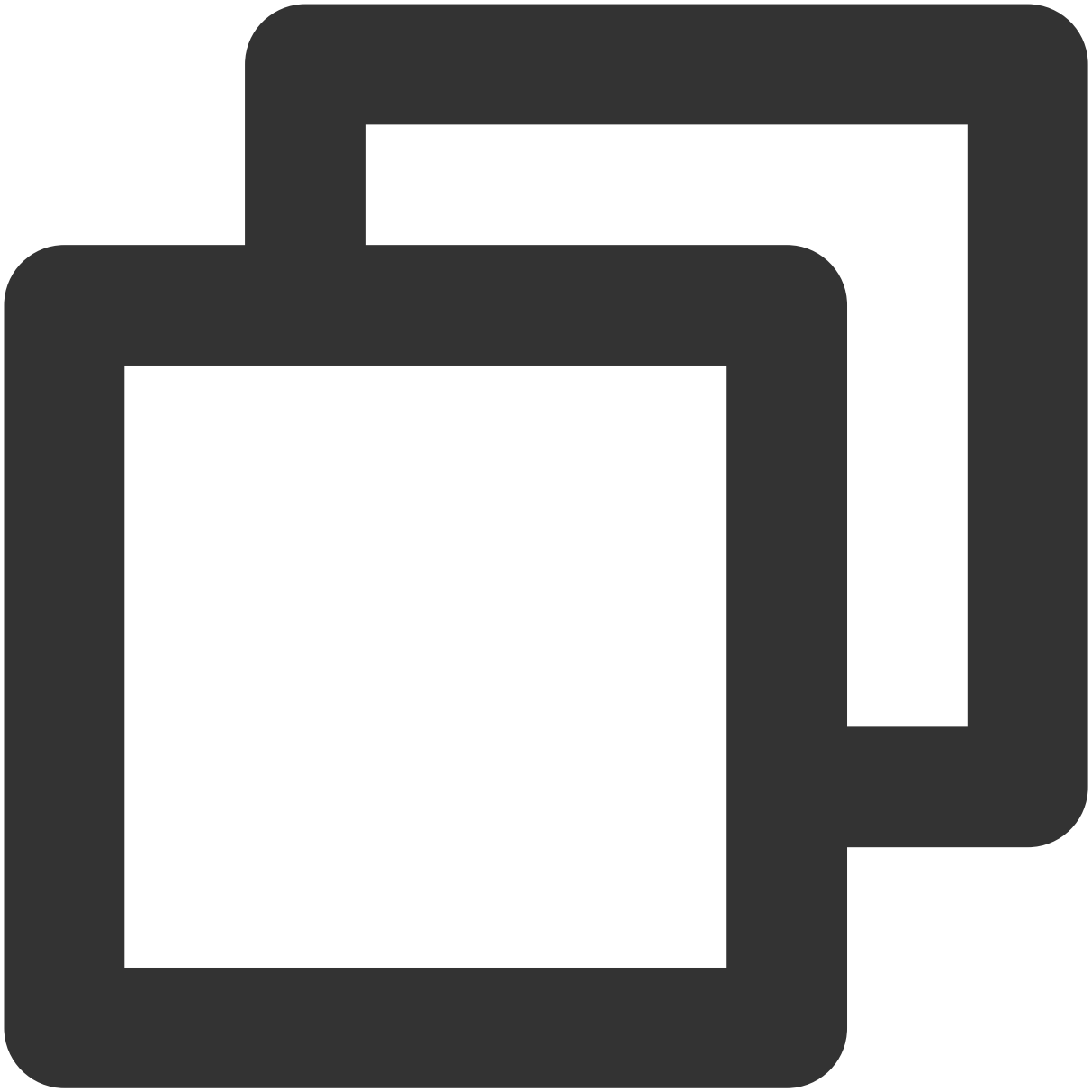
查询存储桶列表



```
package main


import (
    "context"
    "fmt"
    "net/http"
    "os"


    "github.com/tencentyun/cos-go-sdk-v5"
```

```
)


func main() {
    c := cos.NewClient(nil, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })


    s, _, err := c.Service.Get(context.Background())
    if err != nil {
        panic(err)
    }


    for _, b := range s.Buckets {
        fmt.Printf("%#v\\n", b)
    }
}
```

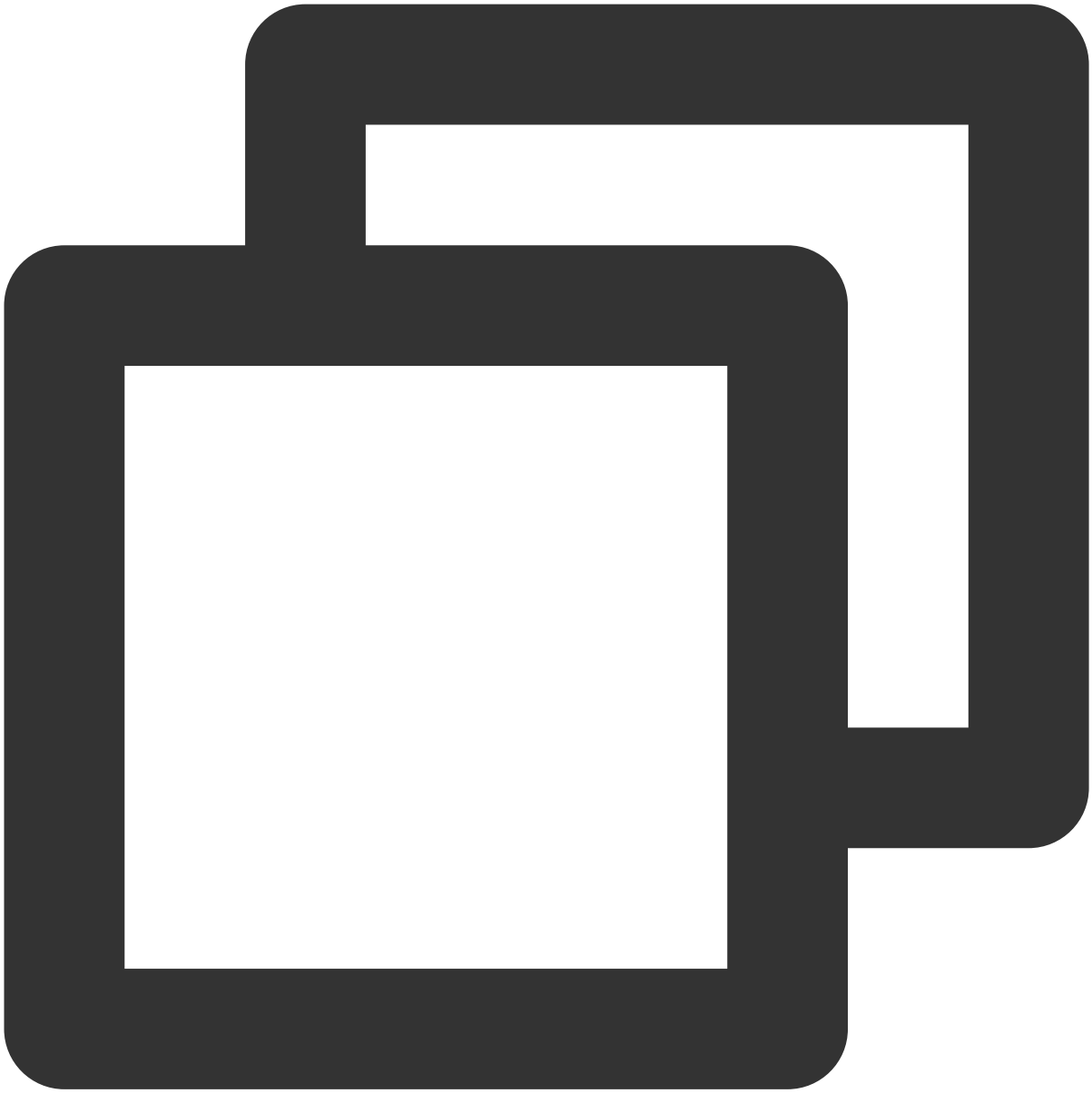
上传对象



```
package main


import (
    "context"
    "net/http"
    "net/url"
    "os"
    "strings"

    "github.com/tencentyun/cos-go-sdk-v5"
```

```
)


func main() {
    // 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
    // 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。
    // COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地
    u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，授
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })
    // 对象键（Key）是对象在存储桶中的唯一标识。
    // 例如，在对象的访问域名 `examplebucket-1250000000.cos.COS_REGION.myqcloud.com/tes
    name := "test/objectPut.go"
    // 1.通过字符串上传对象
    f := strings.NewReader("test")


    _, err := c.Object.Put(context.Background(), name, f, nil)
    if err != nil {
        panic(err)
    }
    // 2.通过本地文件上传对象
    _, err = c.Object.PutFromFile(context.Background(), name, "../test", nil)
    if err != nil {
        panic(err)
    }
    // 3.通过文件流上传对象
    fd, err := os.Open("./test")
    if err != nil {
        panic(err)
    }
    defer fd.Close()
    _, err = c.Object.Put(context.Background(), name, fd, nil)
    if err != nil {
        panic(err)
    }
}
```

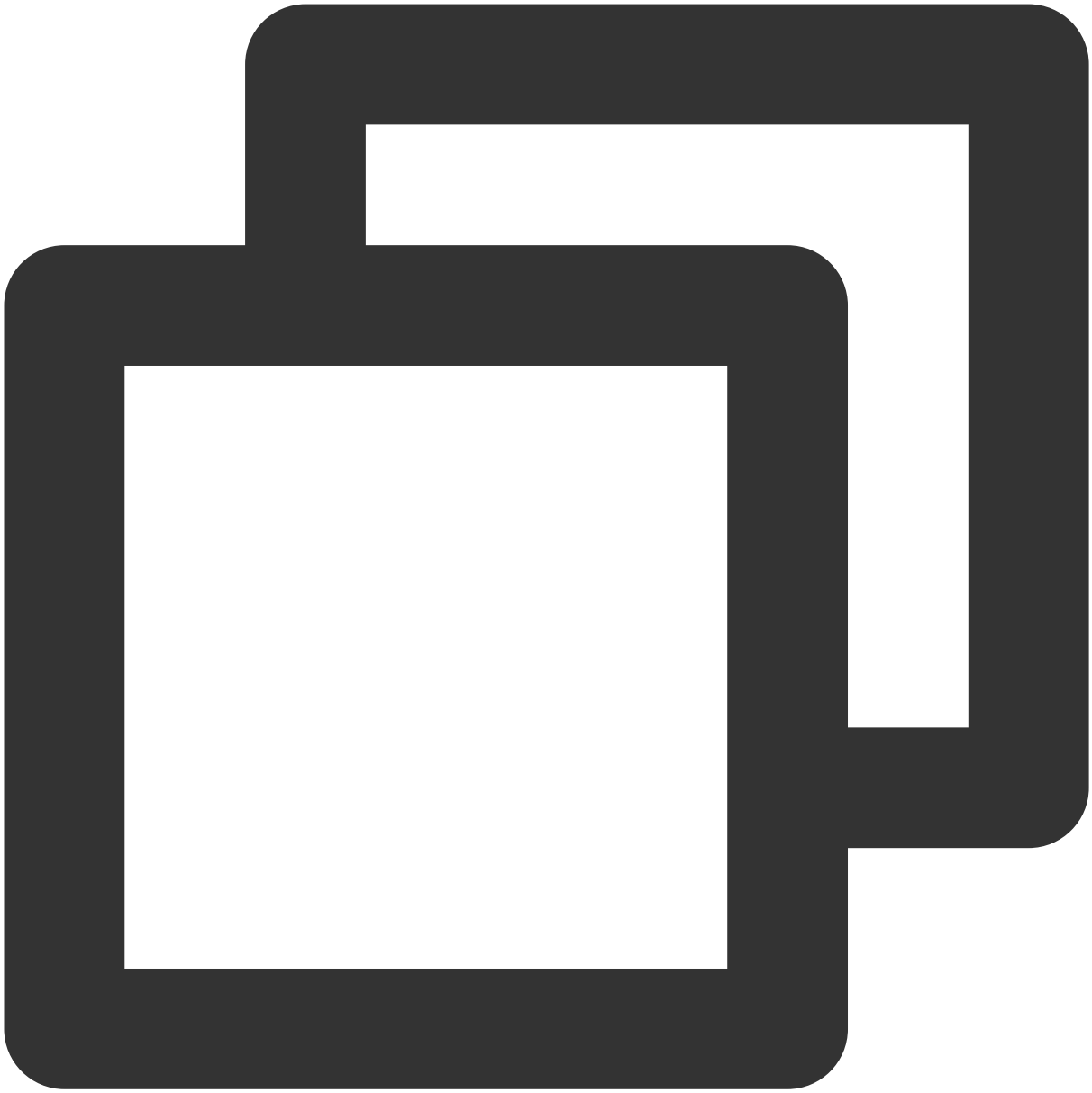
查询对象列表



```
package main


import (
    "context"
    "fmt"
    "net/http"
    "net/url"
    "os"

    "github.com/tencentyun/cos-go-sdk-v5"
```

```
)


func main() {
    // 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
    // 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。
    // COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地
    u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，按
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })


    opt := &cos.BucketGetOptions{
        Prefix:  "test",
        MaxKeys: 3,
    }
    v, _, err := c.Bucket.Get(context.Background(), opt)
    if err != nil {
        panic(err)
    }


    for _, c := range v.Contents {
        fmt.Printf("%s, %d\\n", c.Key, c.Size)
    }
}
```

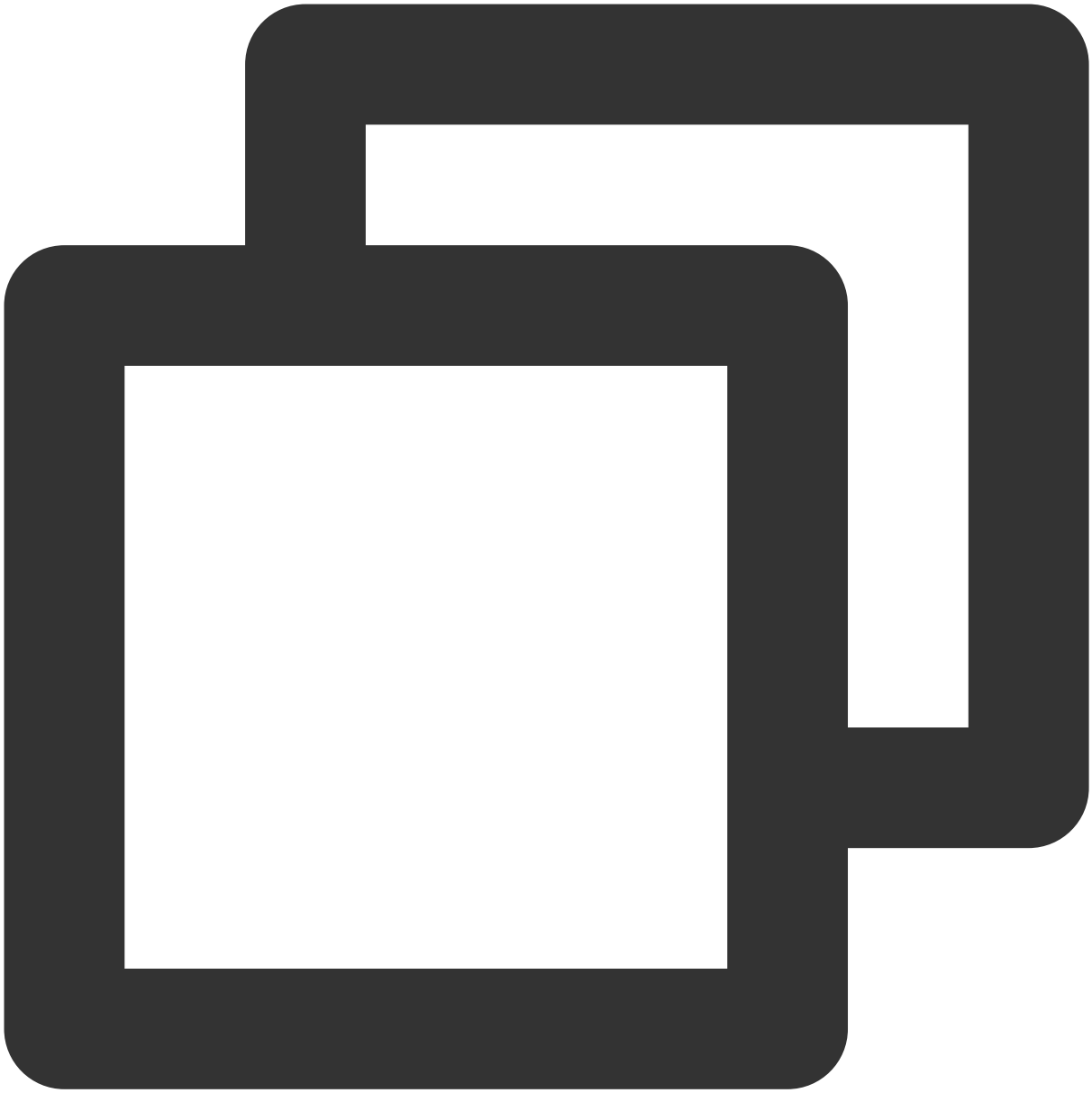
下载对象



```
package main


import (
    "context"
    "fmt"
    "io/ioutil"
    "net/http"
    "net/url"
    "os"
```

```
    "github.com/tencentyun/cos-go-sdk-v5"
)


func main() {
    // 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
    // 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。
    // COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地
    u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，授
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })
    // 1.通过响应体获取对象
    name := "test/objectPut.go"
    resp, err := c.Object.Get(context.Background(), name, nil)
    if err != nil {
        panic(err)
    }
    bs, _ := ioutil.ReadAll(resp.Body)
    resp.Body.Close()
    fmt.Printf("%s\\n", string(bs))
    // 2.获取对象到本地文件
    _, err = c.Object.GetToFile(context.Background(), name, "exampleobject", nil)
    if err != nil {
        panic(err)
    }
}
```

删除对象



```
package main


import (
    "context"
    "net/http"
    "net/url"
    "os"

    "github.com/tencentyun/cos-go-sdk-v5"
)
```

```
func main() {
    // 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
    // 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。
    // COS_REGION 可以在控制台查看，https://console.tencentcloud.com/cos5/bucket，关于地
    u, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com
    b := &cos.BaseURL{BucketURL: u}
    c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),  // 用户的 SecretId，建议使用子账号密钥，按
            SecretKey: os.Getenv("SECRETKEY"), // 用户的 SecretKey，建议使用子账号密钥，
        },
    })
    name := "test/objectPut.go"
    _, err := c.Object.Delete(context.Background(), name)
    if err != nil {
        panic(err)
    }
}
```

图片处理
图片持久化处理

最近更新时间：2024-03-04 16:55:55

简介

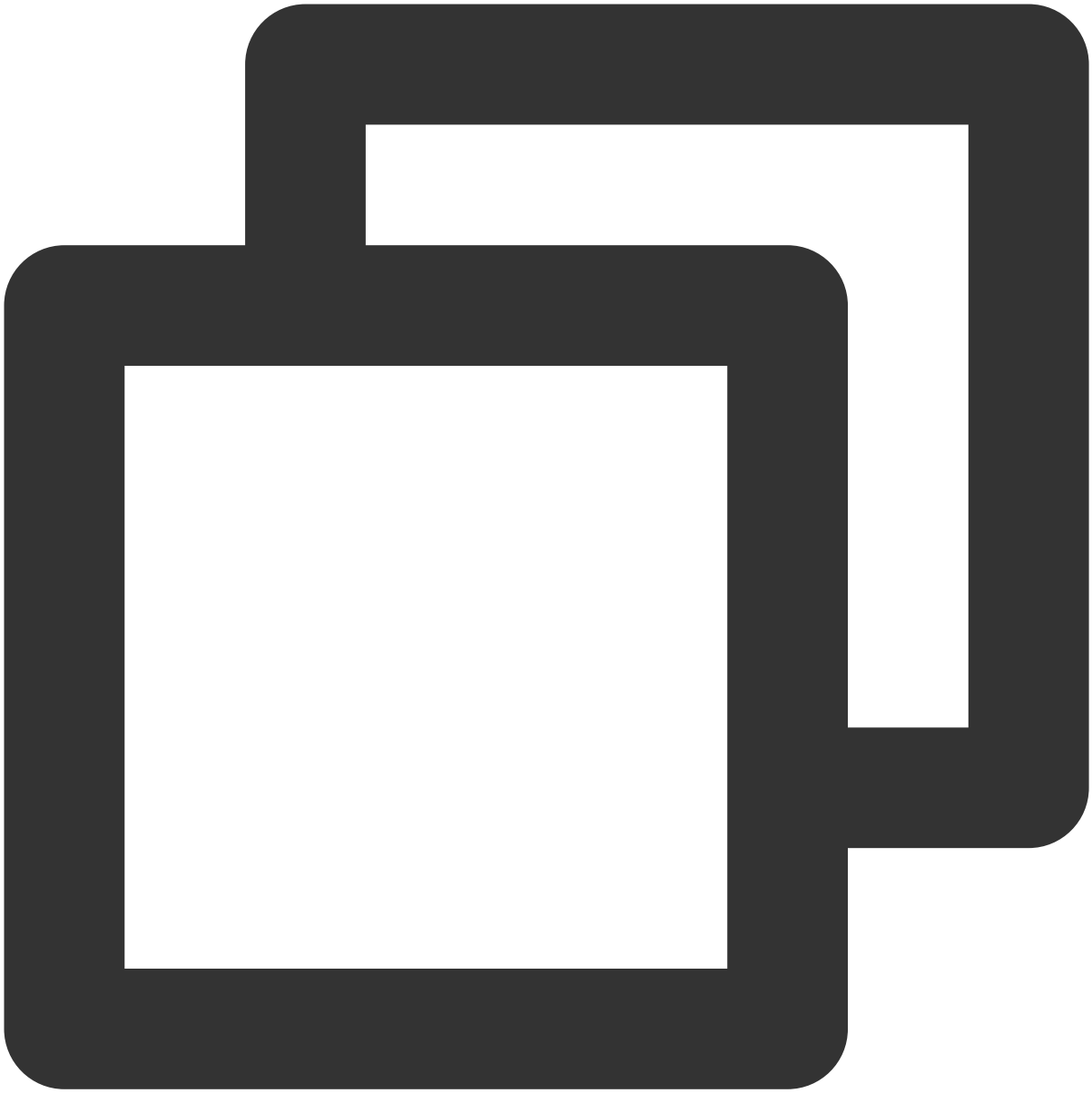
本文档提供关于图片持久化处理的 API 概览以及 SDK 示例代码。

API	说明
图片持久化处理	用于图片持久化处理

上传时处理

COS 提供的上传时处理功能可以帮助使用者在上传时实现图片处理。您只需要在请求包头部中加入 Pic-Operations 项并设置好相应参数，即可在图片上传时实现相应的图片处理，并可将原图和处理结果存入到 COS。目前支持20M以内文件处理。

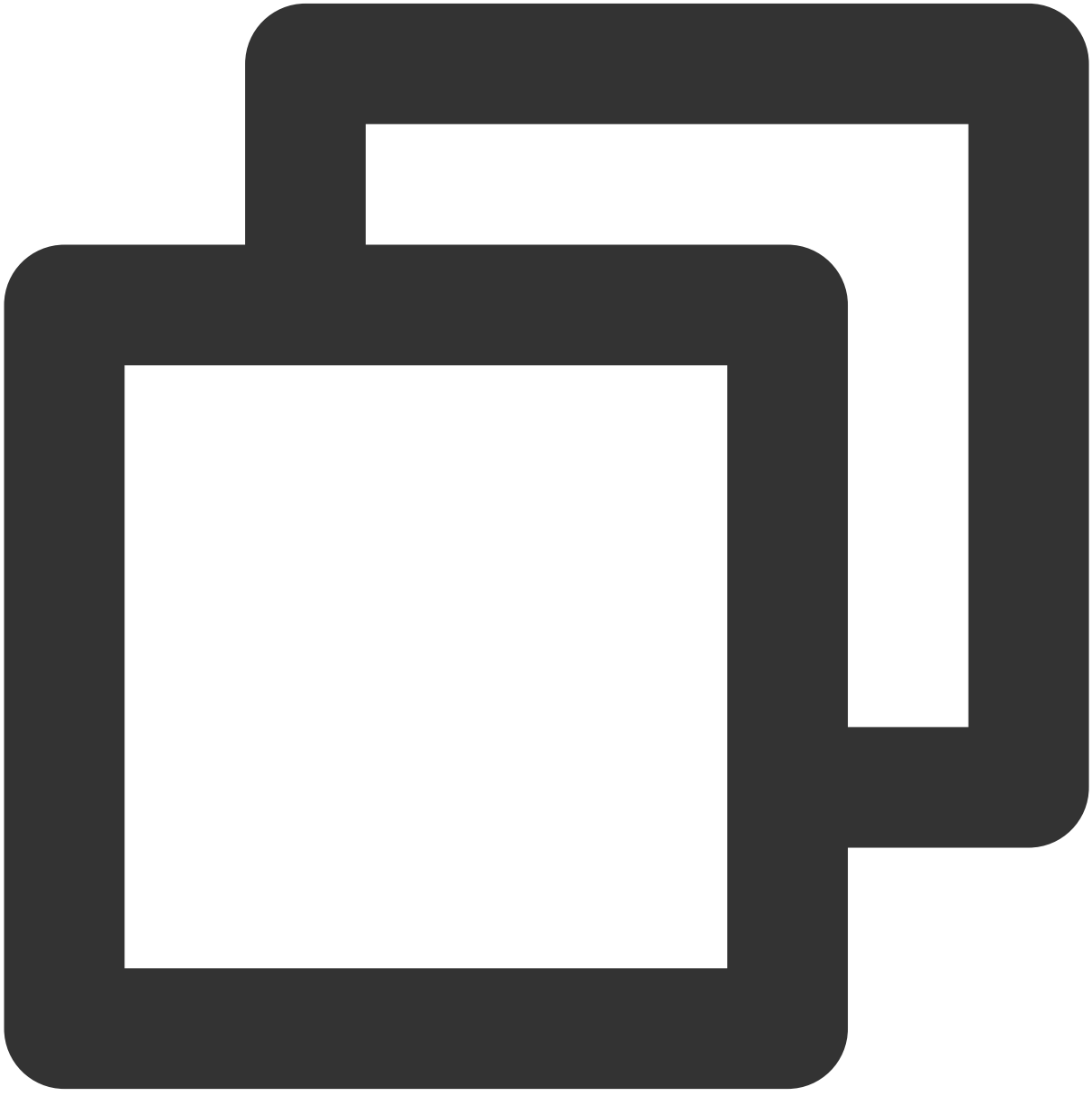
请求示例



```
opt := &cos.ObjectPutOptions{
        nil,
        &cos.ObjectPutHeaderOptions{
            XOptionHeader: &http.Header{},
        },
}
pic := &cos.PicOperations{
    IsPicInfo: 1,
    Rules: []cos.PicOperationsRules{
        {
            FileId: "format.jpg",
```

```
            Rule:   "imageView2/format/png",
        },
    },
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
name := "test.jpg"
local_filename := "./test.jpg"
res, _, err := c.CI.PutFromFile(context.Background(), name, local_filename, opt)
```

参数说明



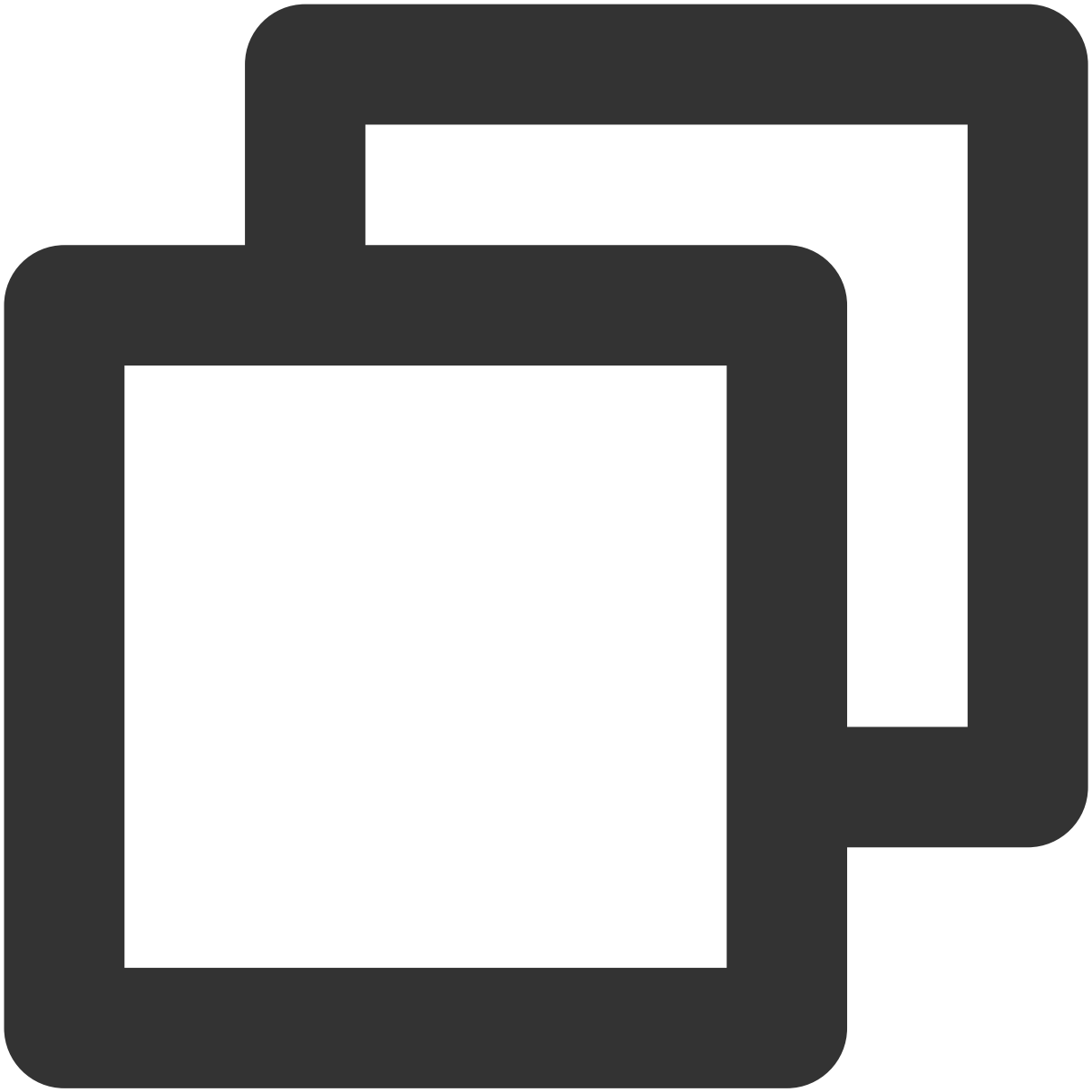
```
type PicOperations struct {
    IsPicInfo int
    Rules     []PicOperationsRules
}
type PicOperationsRules struct {
    Bucket string
    FileId string
    Rule   string
}
```

参数名称	参数描述	类型	是否必填
IsPicInfo	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	int	否
Rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	Array	否
Bucket	存储结果的目标存储桶名称，格式为 BucketName-APPID，如果不指定的话默认保存到当前存储桶	string	否
FileId	处理结果的文件路径名称，如以 `/` 开头，则存入指定文件夹中，否则，存入原图文件存储的同目录	string	是
Rule	处理参数，参见图片持久化处理 API。若按指定样式处理，则以 `style/` 开头，后加样式名，如样式名为 `test`，则 rule 字段为 `style/test`	string	否

云上数据处理

该接口能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

请求示例



```
opt := &cos.ImageProcessOptions{
    IsPicInfo: 1,
    Rules: []cos.PicOperationsRules{
        {
            FileId: "format.jpg",
            Rule:   "imageView2/format/png",
        },
    },
}
name := "test.jpg"
res, _, err := c.CI.ImageProcess(context.Background(), name, opt)
```

版权所有：腾讯云计算（北京）有限责任公司

基础图片处理

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于基础图片处理的 API 概览以及 SDK 示例代码。

API	操作描述
缩放	对图片进行缩小或放大
裁剪	对图片进行裁剪，包括普通裁剪、缩放裁剪、内切圆裁剪、圆角裁剪和人脸智能裁剪
旋转	对图片进行旋转，包括普通旋转和自适应旋转
格式转换	对图片进行格式转换、gif 格式优化、渐进显示
质量变换	对图片质量进行调节
高斯模糊	对图片进行模糊处理
锐化	对图片进行锐化
图片水印	对图片进行水印处理
文字水印	对图片进行实时文字水印处理
获取图片基本信息	查询图片基本信息，包括格式、长、宽等
获取图片 EXIF	查询 EXIF 信息
获取图片主色调	查询图片主色调信息
去除元信息	去除图片元信息，包括 exif 信息
快速缩略模板	通过图片处理模板，生成相应的缩略图
管道操作符	实现对图片按顺序进行多种处理

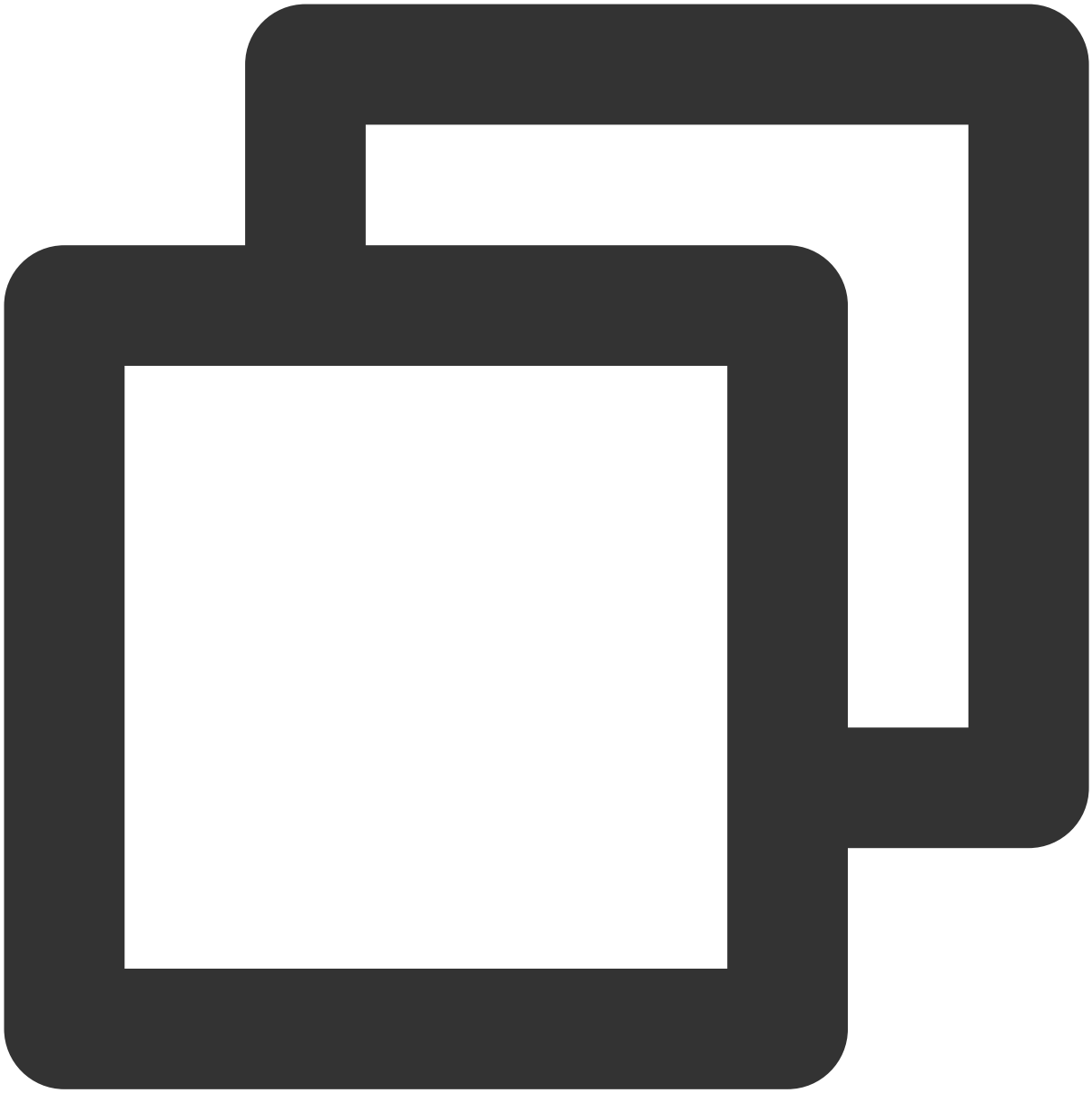
缩放

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供图片缩放功能。

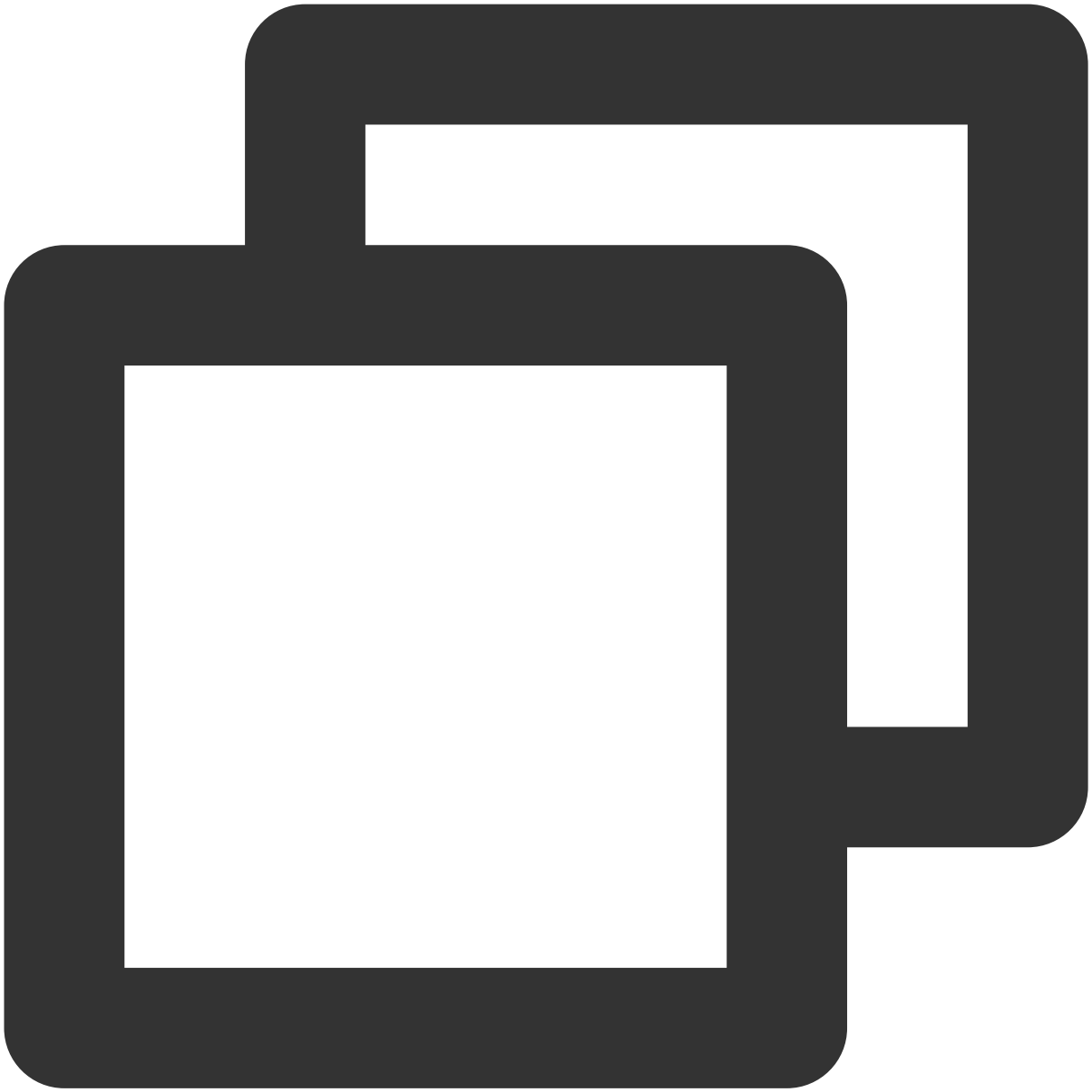
版权所有：腾讯云计算（北京）有限责任公司

方法原型



```
func (s *CIService) Get(ctx context.Context, key string, operation string, opt *Obj

func (s *CIService) GetToFile(ctx context.Context, key, localpath, operation string
```

请求示例



```
name := "test.jpg"
// Case 1 从响应体中获取对象
resp, err := c.CI.Get(context.Background(), name, "imageMogr2/thumbnail/!50px", nil
if err != nil {
    //ERROR
}
defer resp.Body.Close()
ioutil.ReadAll(resp.Body)

// Case 2 下载对象到文件
filepath := "test.jpg"
```

```
_, err = c.CI.GetToFile(context.Background(), name, filepath, "imageMogr2/thumbnail
```

参数说明

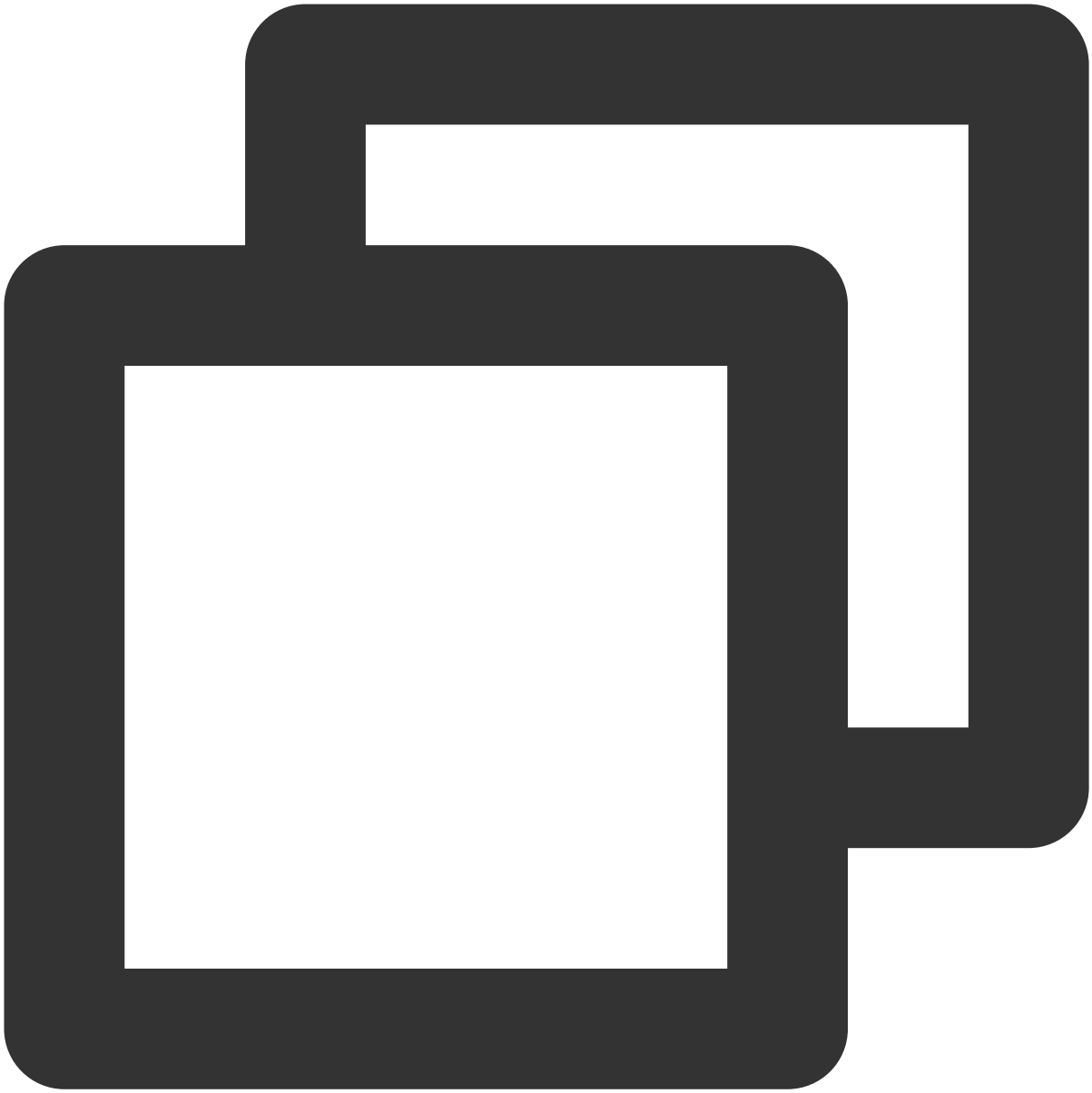
参数名称	参数描述	类型	是否必填
key	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	string	是
operation	基础图片处理，可以通过该参数实现缩放、裁剪、旋转、格式转换、质量变换等基础图片处理功能。	string	是
ObjectGetOptions	对象下载参数，详见 下载对象	string	否
id	对象 VersionId	string	否

裁剪

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供裁剪功能，包括普通裁剪、缩放裁剪、内切圆裁剪、圆角裁剪和人脸智能裁剪。

请求示例

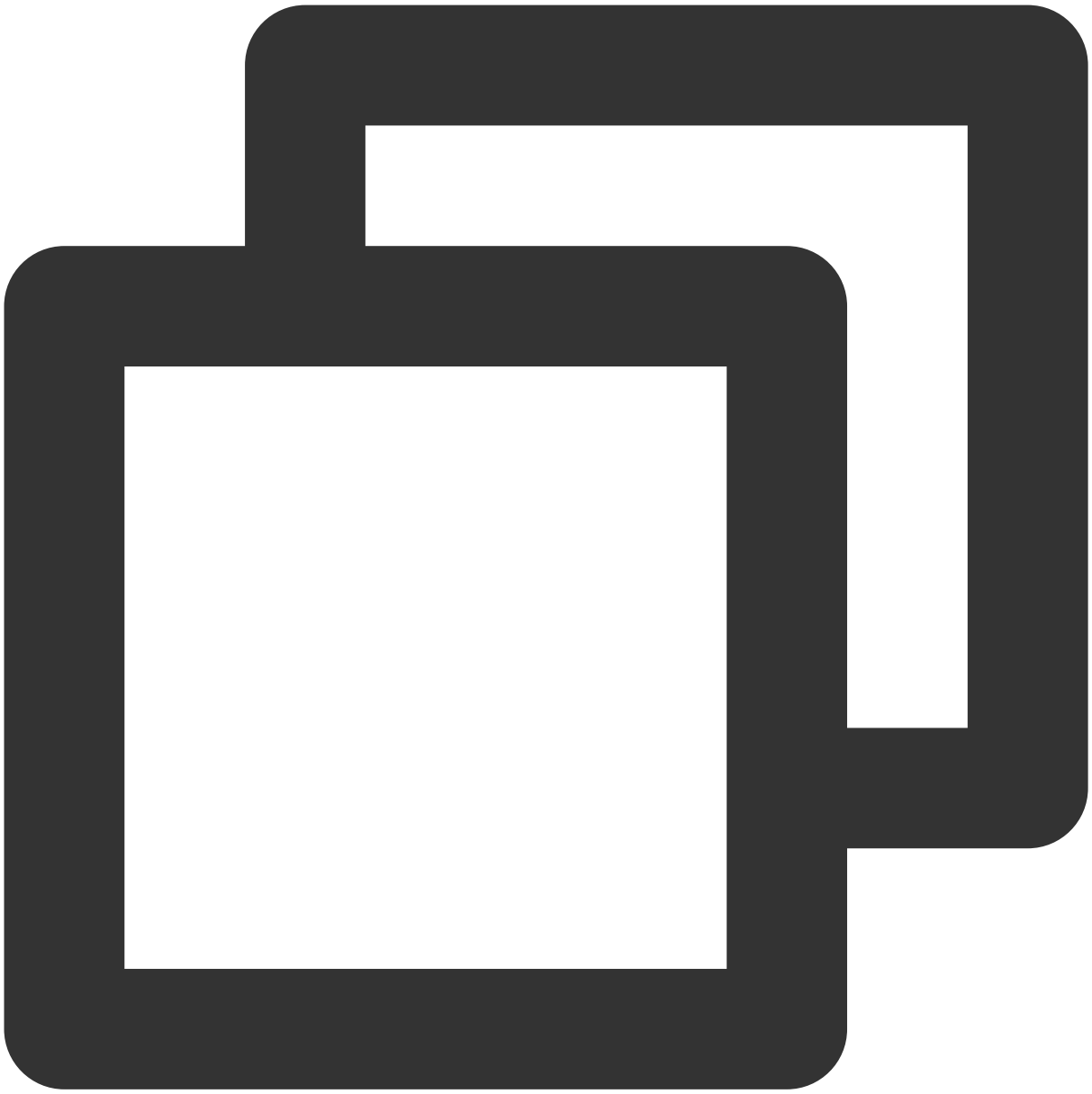


```
name := "test.jpg"
filepath := "test.jpg"
_, err = c.CI.GetToFile(context.Background(), name, filepath, "imageMogr2/cut/600x6
```

其他基础图片处理

功能说明

基础图片处理统一使用上述方法完成，其他基础图片处理只需要修改 operation 参数的值即可，例如将图片顺时针旋转90度的样例如下。



```
name := "test.jpg"
filepath := "test.jpg"
_, err = c.CI.GetToFile(context.Background(), name, filepath, "imageMogr2/rotate/90
```

图片高级压缩

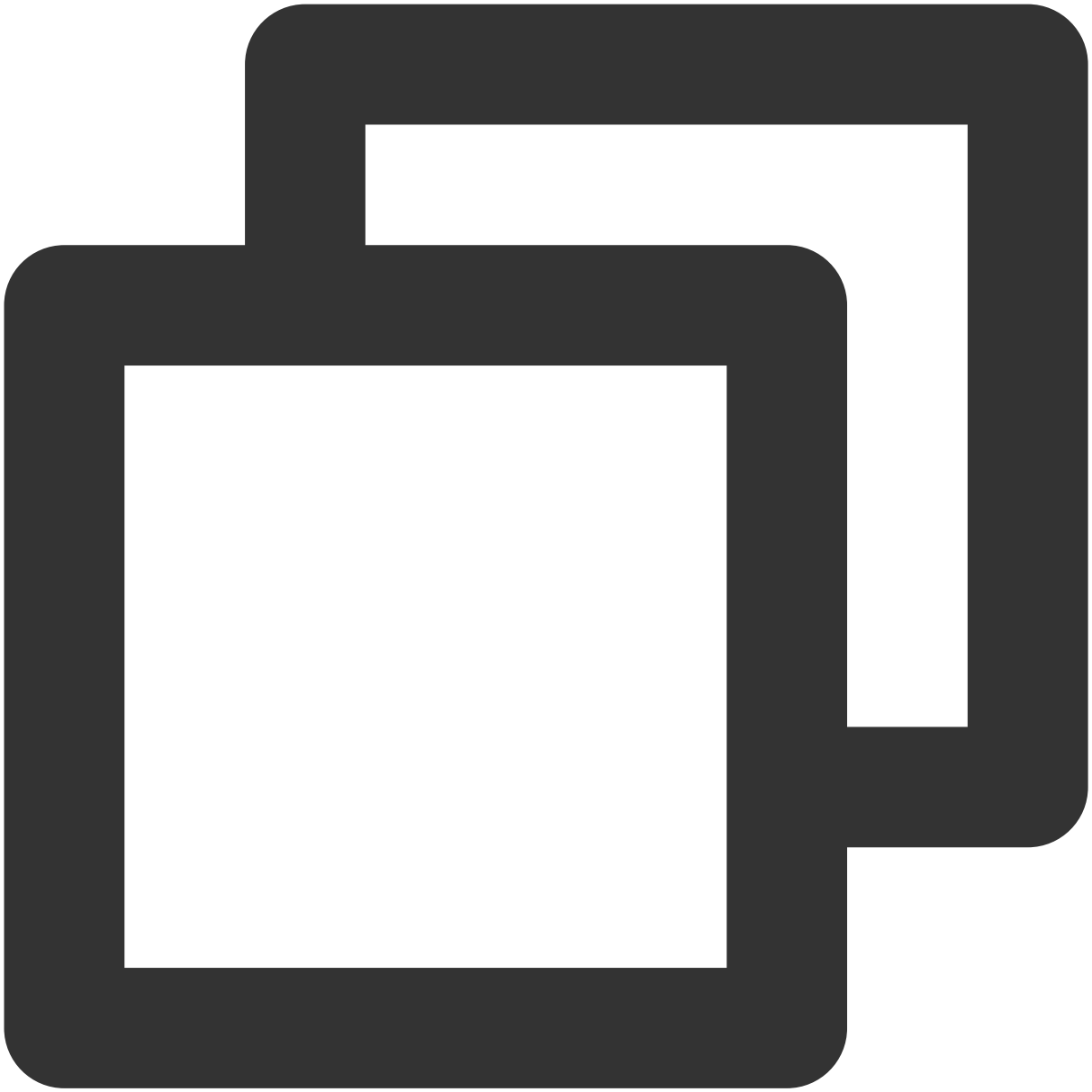
最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于图片高级压缩的 API 概览以及 SDK 示例代码。

API	操作描述
图片高级压缩	图片高级压缩可以更加高效地将图片转换为 TPG 或 HEIF 等高压缩比格式，有效降低图片传输链路及加载耗时，降低带宽及流量成本

请求示例



```
// 将原图转换为 TPG 格式
name := "test.png"
filepath := "test.tpg"
_, err := c.CI.GetToFile(context.Background(), name, filepath, "imageMogr2/format/t
if err != nil {
    // ERROR
}

// 将原图转换为 HEIF 格式
filepath = "test.heif"
_, err = c.CI.GetToFile(context.Background(), name, filepath, "imageMogr2/format/he
```

```
if err != nil {
    // ERROR
}
```

版权所有：腾讯云计算（北京）有限责任公司

Guetzli 压缩

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于 Guetzli 压缩的相关的 API 概览以及 SDK 示例代码。

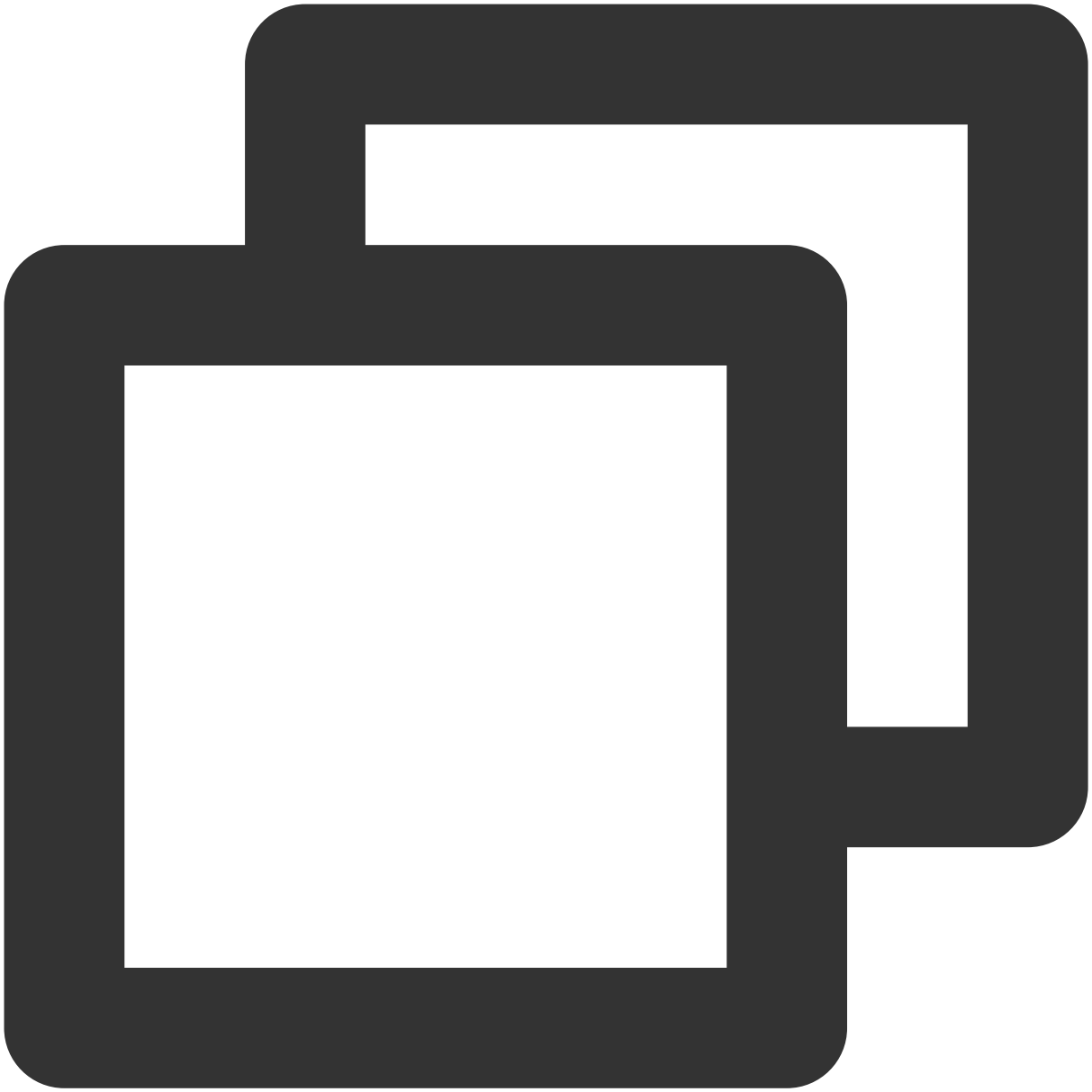
API	操作描述
开通 Guetzli 压缩	对 Bucket 开通 Guetzli 压缩功能
查询 Guetzli 状态	用于查询 Guetzli 压缩功能是否开启
关闭 Guetzli 压缩	用于关闭 Guetzli 压缩功能

开通 Guetzli 压缩

功能说明

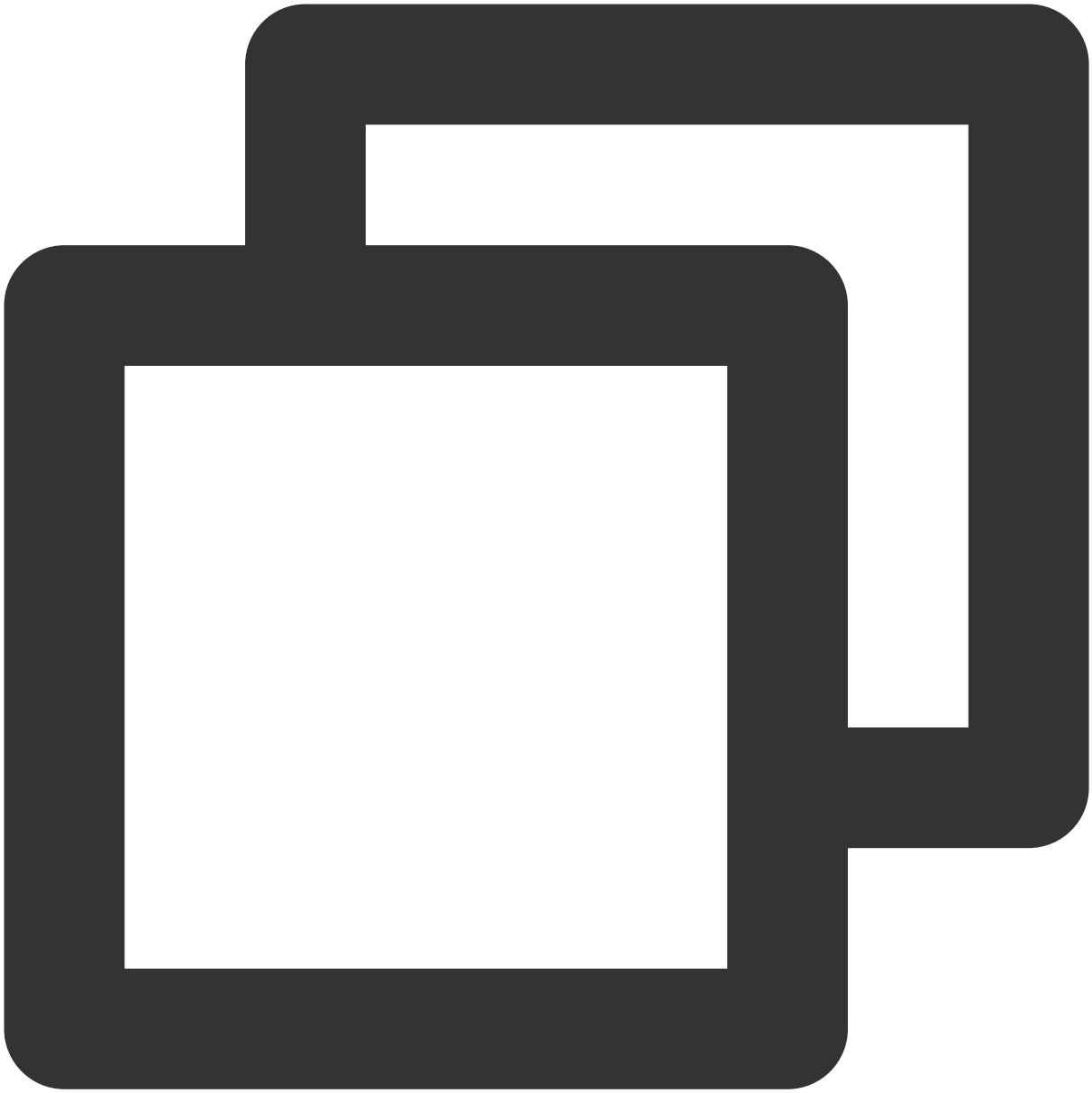
对 Bucket 开通 Guetzli 压缩功能。

方法原型



```
func (s *CIService) PutGuetzli(ctx context.Context) (*Response, error)
```

请求示例



```
// 需要填写CIURL
u, _ := url.Parse("https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com")
cu, _ := url.Parse("http://examplebucket-1250000000.pic.ap-guangzhou.myqcloud.com")
b := &cos.BaseURL{BucketURL: u, CIURL: cu}
c := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        SecretID:  os.Getenv("SECRETID"),
        SecretKey: os.Getenv("SECRETKEY"),
    },
})
```

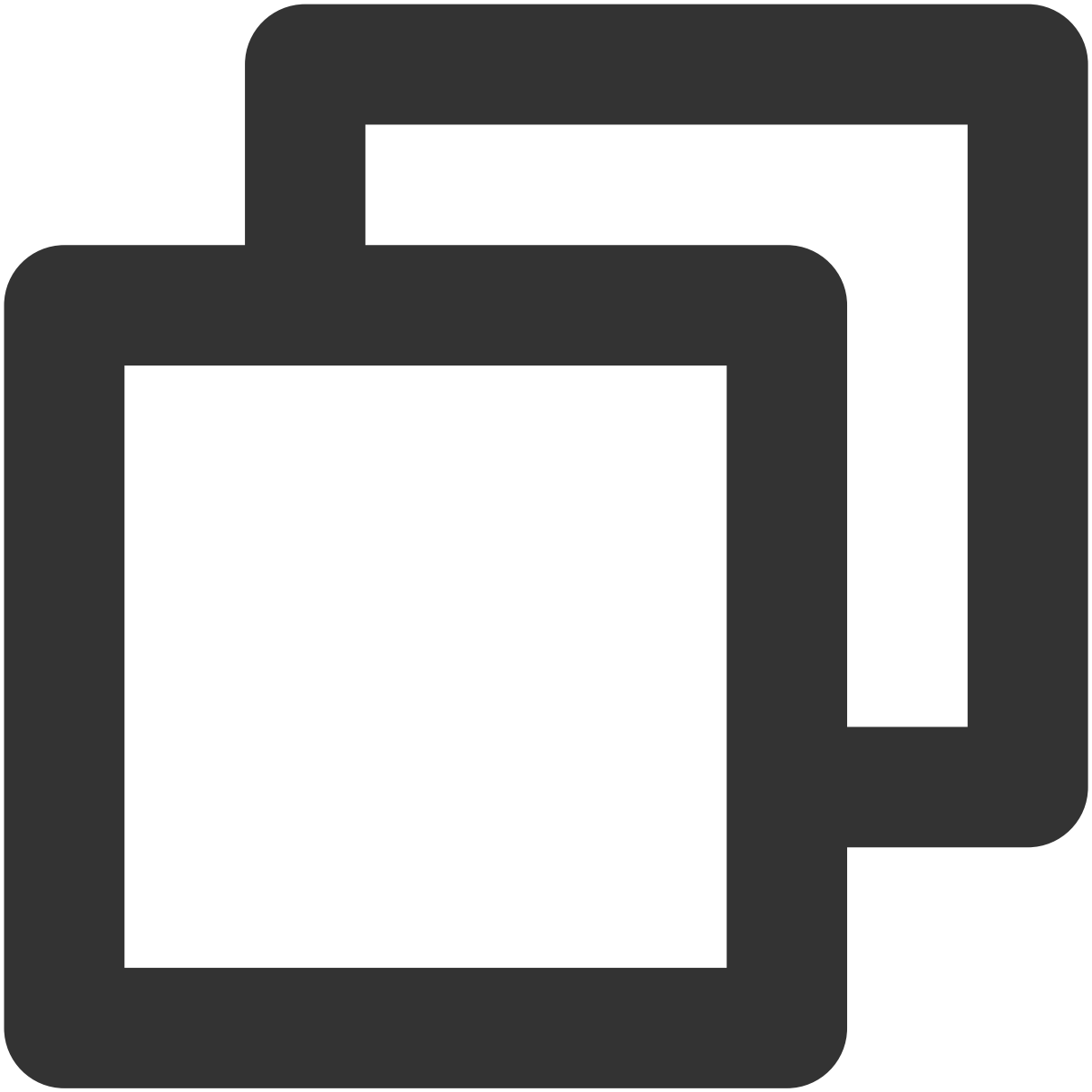
```
_, err := c.CI.PutGuetzli(context.Background())
log_status(err)
```

查询 Guetzli 状态

功能说明

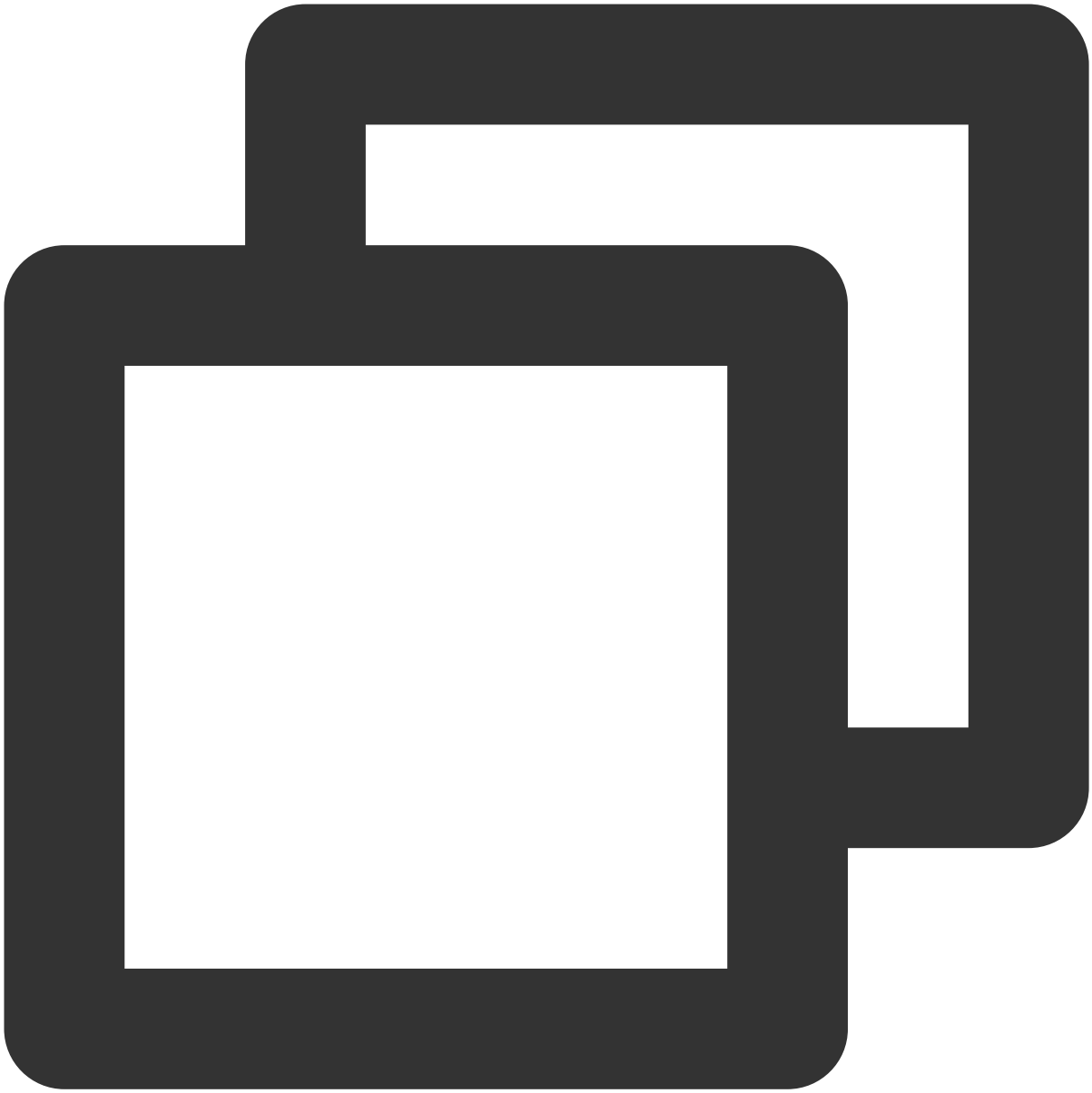
用于查询 Guetzli 压缩功能是否开启。

方法原型



```
func (s *CIService) GetGuetzli(ctx context.Context) (*GetGuetzliResult, *Response,
```

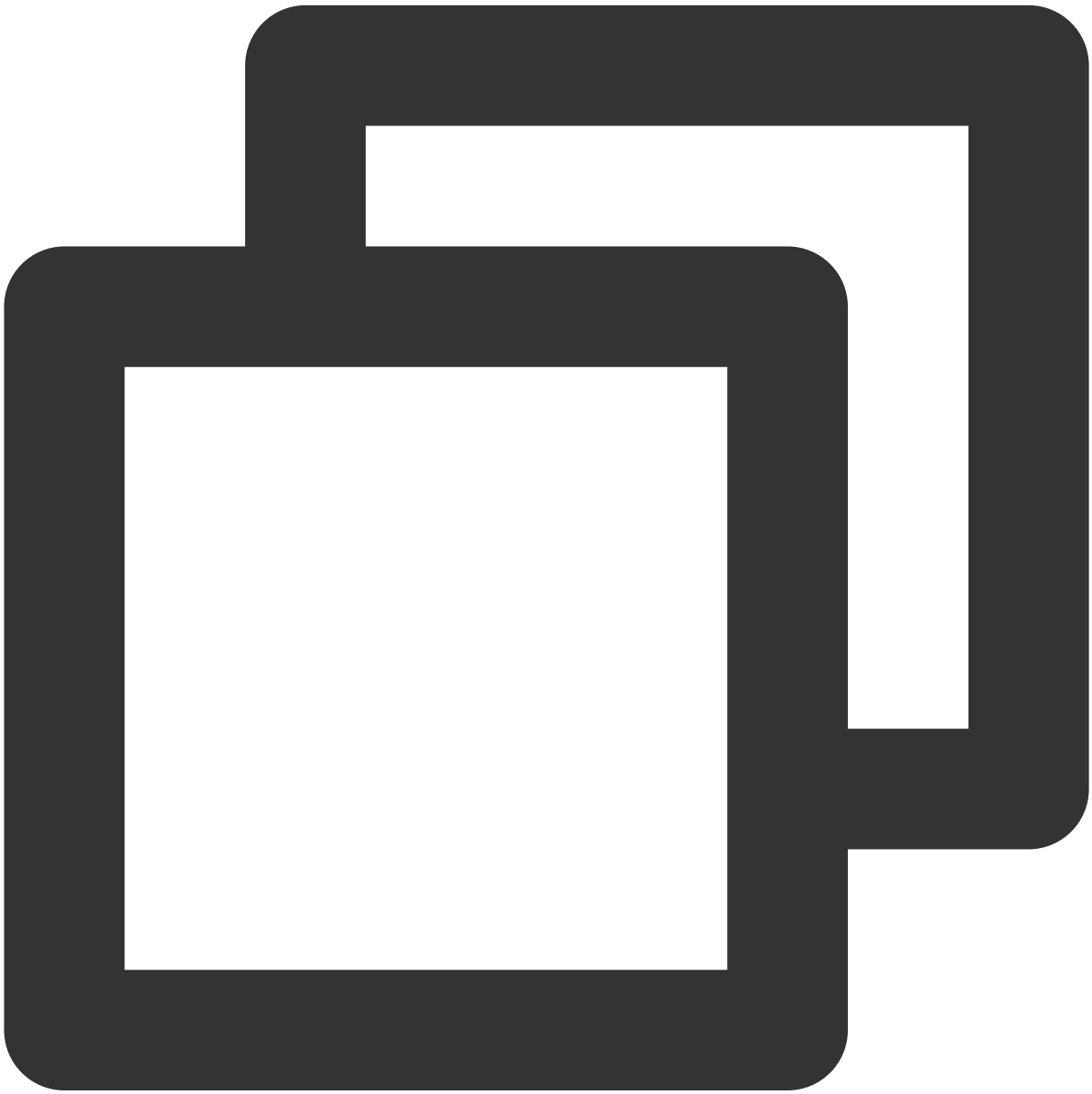
请求示例



```
// 需要填写 CIURL
u, _ := url.Parse("https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com")
cu, _ := url.Parse("http://examplebucket-1250000000.pic.ap-guangzhou.myqcloud.com")
b := &cos.BaseURL{BucketURL: u, CIURL: cu}
c := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        SecretID:  os.Getenv("SECRETID"),
        SecretKey: os.Getenv("SECRETKEY"),
    },
})
```

```
res, _, err := c.CI.GetGuetzli(context.Background())
if err == nil {
    fmt.Printf("%v\\n", res.GuetzliStatus)
}
```

结果说明



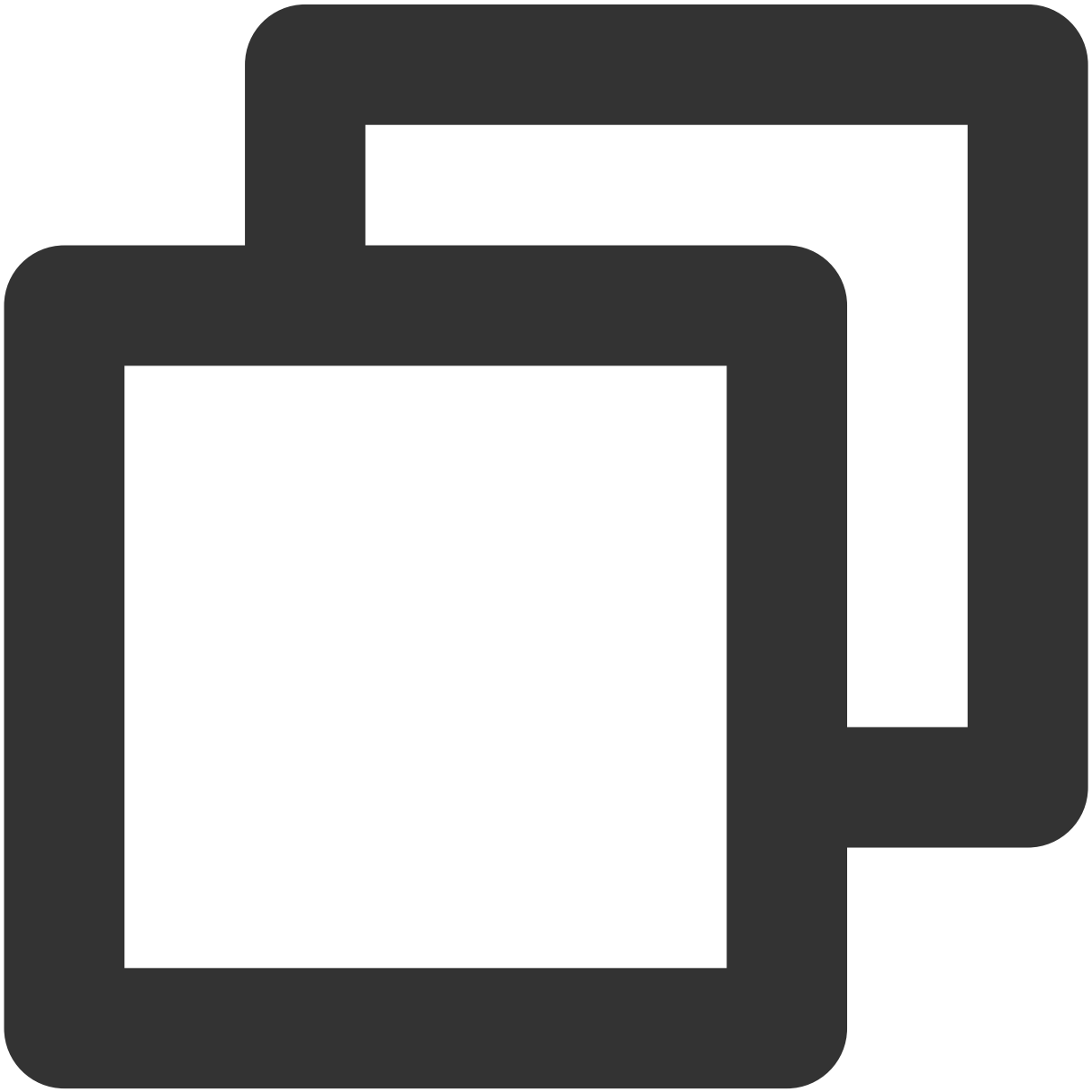
```
type GetGuetzliResult struct {
    GuetzliStatus string
}
```

节点名称	描述	类型
GuetzliStatus	是否开启 Guetzli 压缩功能，拥有 `on` 与 `off` 两种状态。	String

关闭 Guetzli 压缩

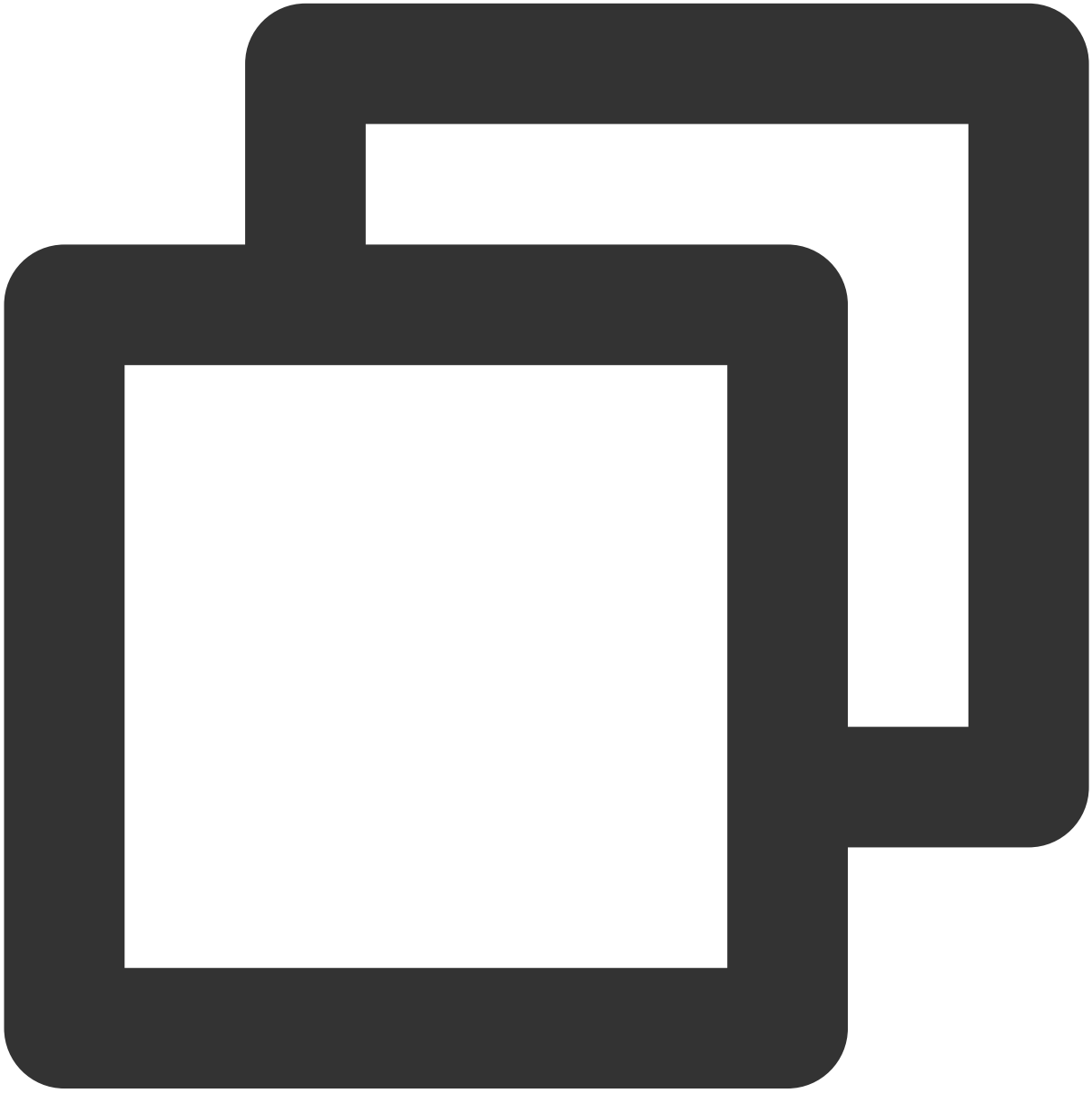
用于关闭 Guetzli 压缩功能。

方法原型



```
func (s *CIService) DeleteGuetzli(ctx context.Context) (*Response, error)
```

请求示例



```
// 需要填写CIURL
u, _ := url.Parse("https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com")
cu, _ := url.Parse("http://examplebucket-1250000000.pic.ap-guangzhou.myqcloud.com")
b := &cos.BaseURL{BucketURL: u, CIURL: cu}
c := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
```

```
        SecretID:  os.Getenv("SECRETID"),
        SecretKey: os.Getenv("SECRETKEY"),
    },
})

_, err := c.CI.DeleteGuetzli(context.Background())
```

盲水印

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。

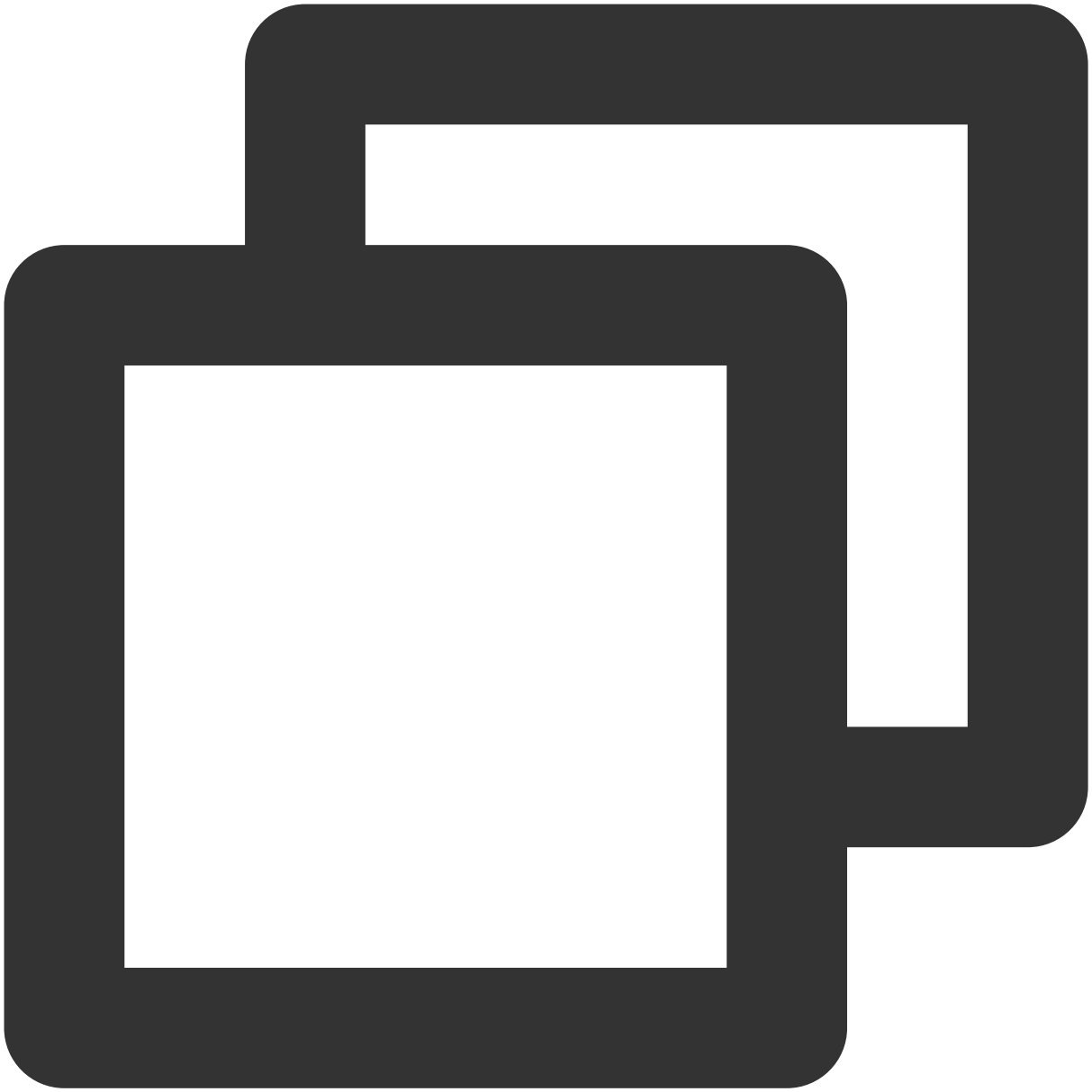
API	操作描述
盲水印	对本地图片添加或提取盲水印并上传至存储桶

添加盲水印

功能说明

盲水印支持在上传时添加以及下载时添加。

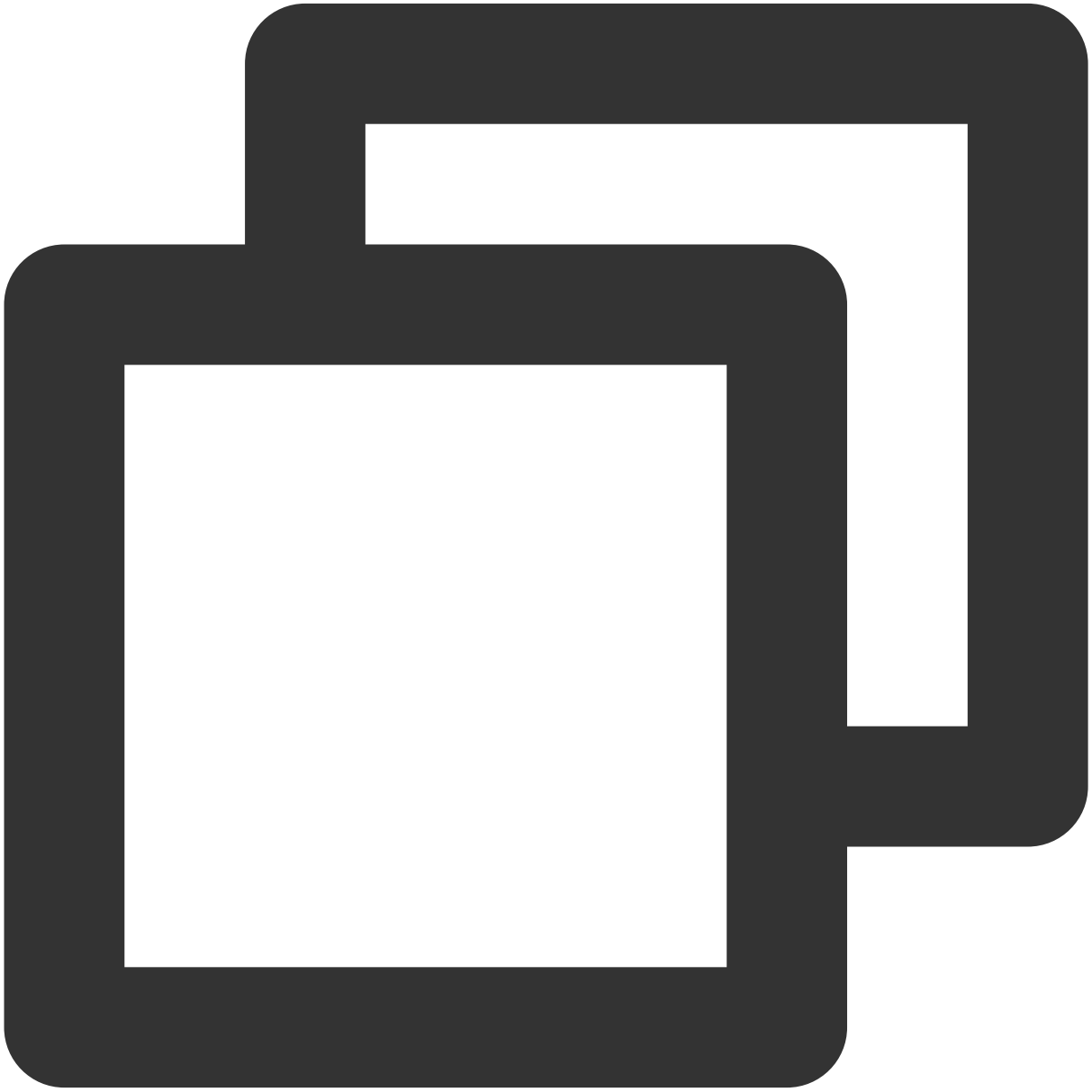
请求示例1: 上传时添加盲水印



```
opt := &cos.ObjectPutOptions{
    nil,
    &cos.ObjectPutHeaderOptions{
        XOptionHeader: &http.Header{},
    },
}
pic := &cos.PicOperations{
    IsPicInfo: 1,
    Rules: []cos.PicOperationsRules{
        {
            FileId: ".jpg",
```

```
            Rule:   "watermark/3/type/3/text/" + base64.StdEncoding.EncodeToString(
        },
    },
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
name := "test.jpg"
filepath := "./test.jpg"
res, _, err := c.CI.PutFromFile(context.Background(), name, filepath, opt)
```

请求示例2：下载时添加盲水印

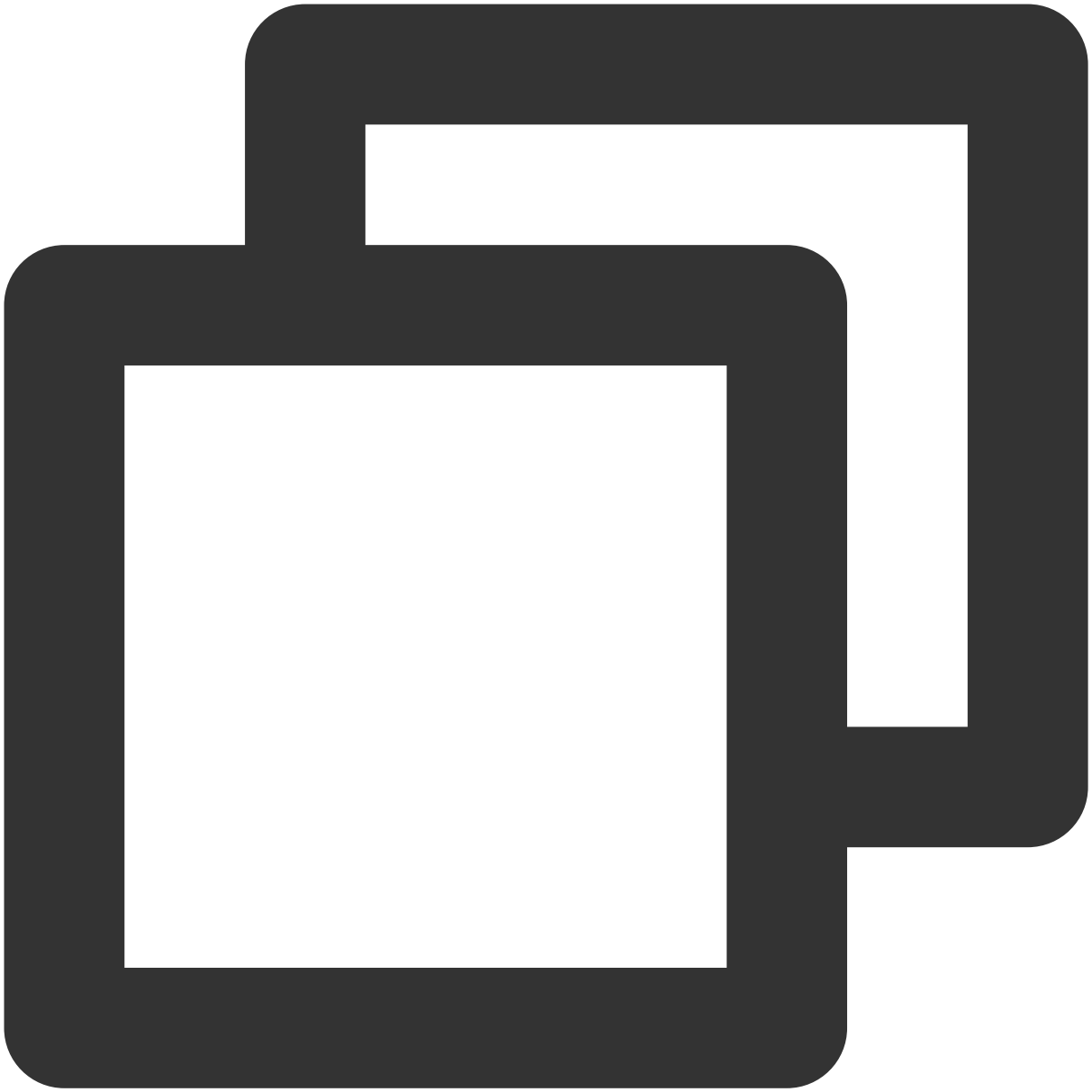


```
name = "test.jpg"
filepath := "watermark.jpg"
_, err = c.CI.GetToFile(context.Background(), name, filepath, "watermark/3/type/3/t
```

提取盲水印

提取盲水印的请求包与 COS 简单上传文件接口一致，只需在请求包头部增加图片处理参数 Pic-Operations 并使用提取盲水印参数（watermark/4）即可。

请求示例



```
opt := &cos.ObjectPutOptions{
    nil,
    &cos.ObjectPutHeaderOptions{
        XOptionHeader: &http.Header{},
    },
}
pic := &cos.PicOperations{
    IsPicInfo: 1,
    Rules: []cos.PicOperationsRules{
        {
            FileId: "format2.jpg",
```

```
            Rule:   "watermark/4/type/3/text/" + base64.StdEncoding.EncodeToString(
        },
    },
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
name := "test2.jpg"
_, err := c.Object.PutFromFile(context.Background(), name, filepath, opt)
```

数据万象服务

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于数据万象的 API 概览以及 SDK 示例代码。

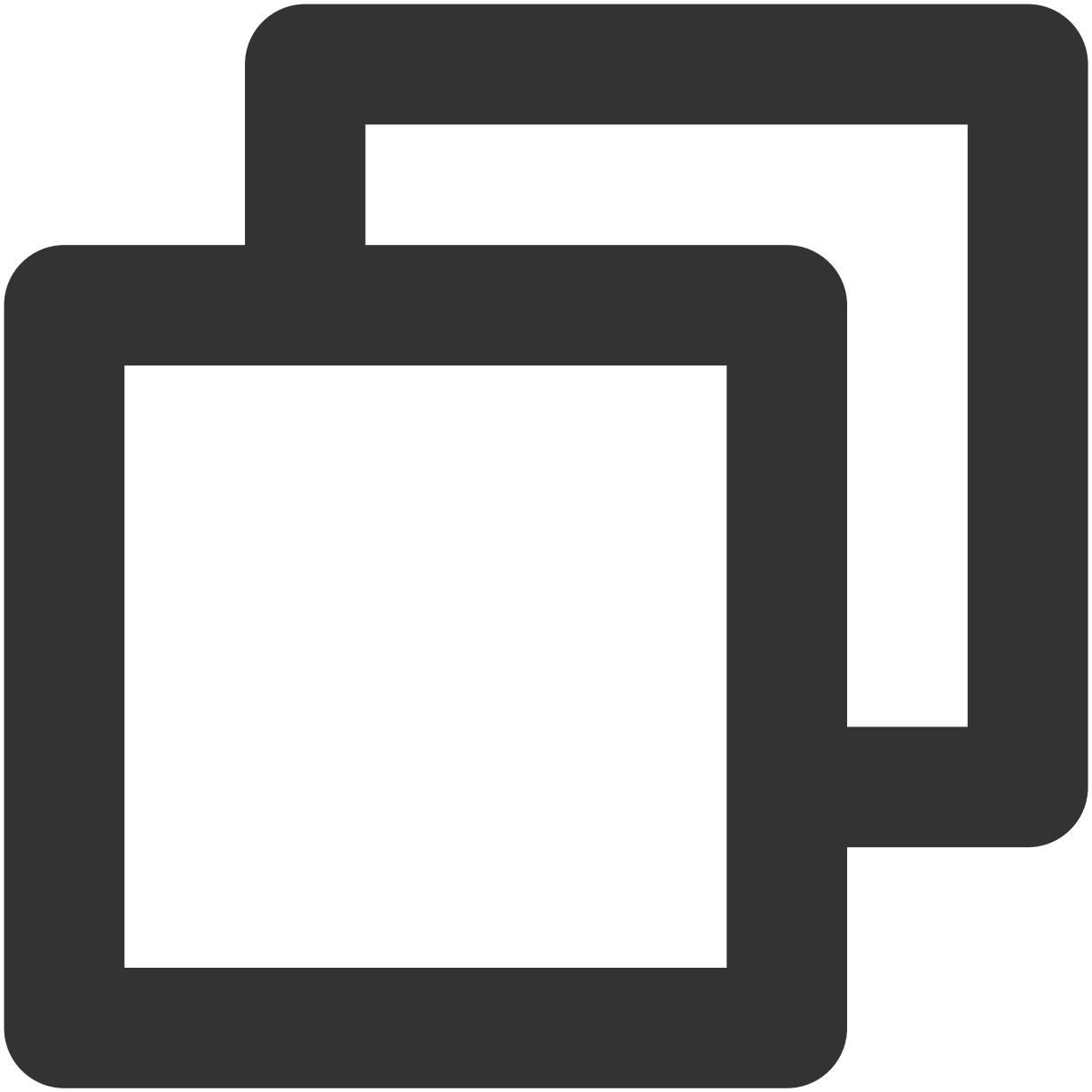
API	操作描述
绑定数据万象服务	用于开通数据万象服务，需要绑定对象存储（Cloud Object Storage，COS）上已有的存储桶。
查询数据万象服务	用于查询该 Bucket 是否已开通数据万象服务。
解绑数据万象服务	解绑开通数据万象服务的 Bucket（Bucket 及桶中文件会保留在对象存储 COS）。

绑定数据万象服务

功能说明

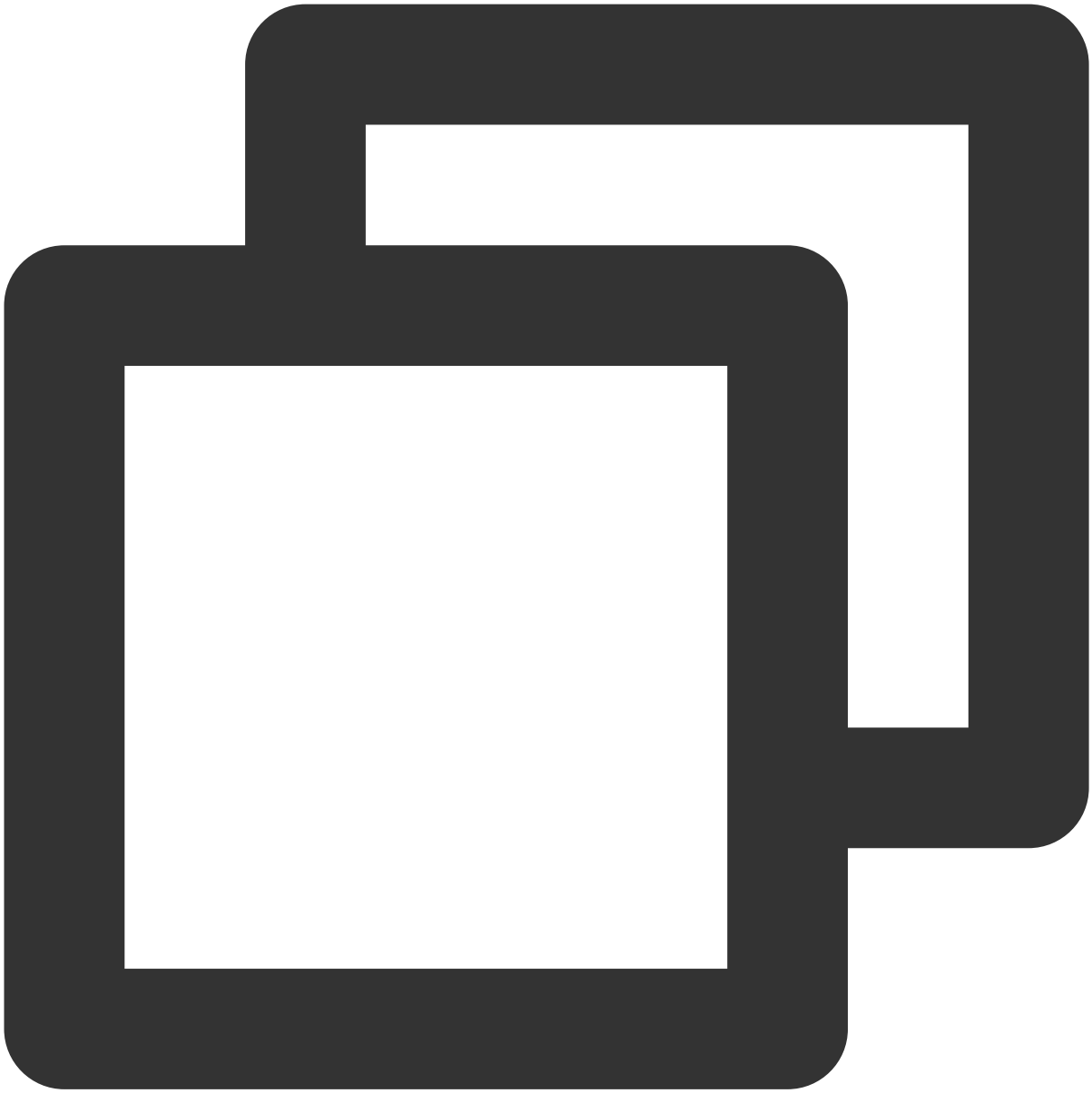
用于开通数据万象服务。

方法原型



```
func (s *CIService) OpenCIService(ctx context.Context) (*Response, error)
```

请求示例



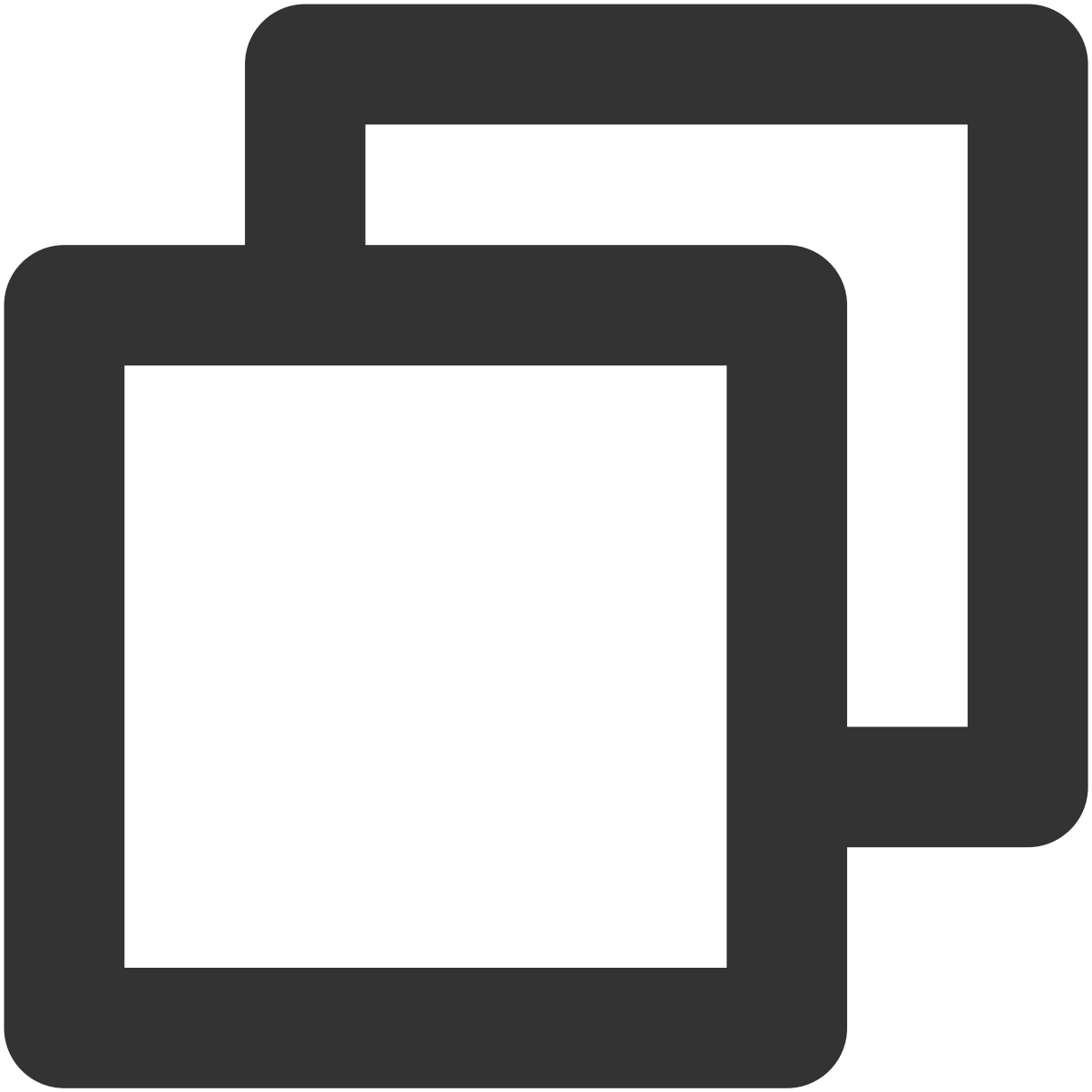
```
_, err := c.CI.OpenCIService(context.Background())
```

查询数据万象服务

功能说明

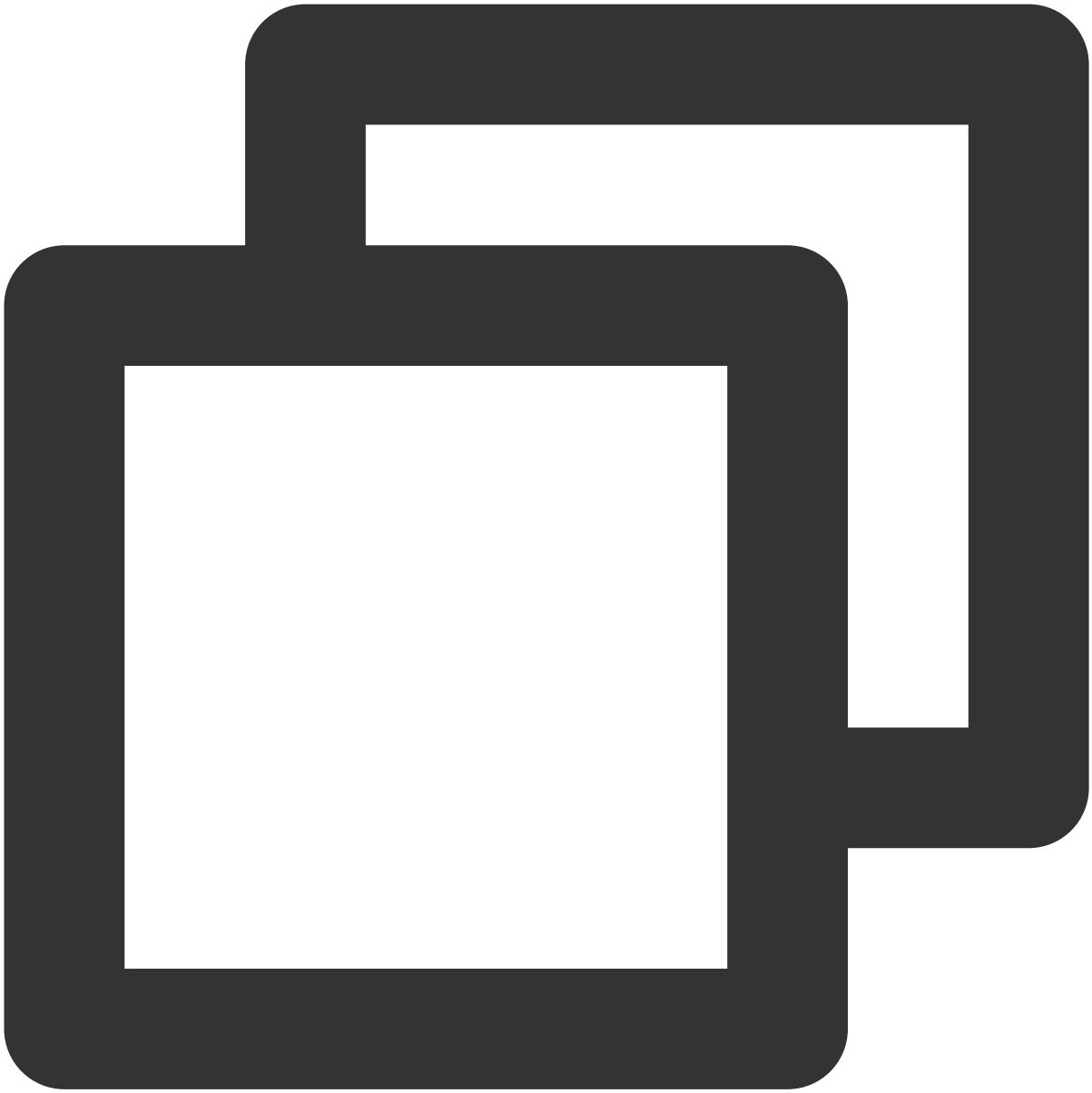
用于查询该 Bucket 是否已开通数据万象服务。

方法原型



```
func (s *CIService) GetCIService(ctx context.Context) (*CIServiceResult, *Response,
```

请求示例



```
res, _, err := c.CI.GetCIService(context.Background())
```

参数说明

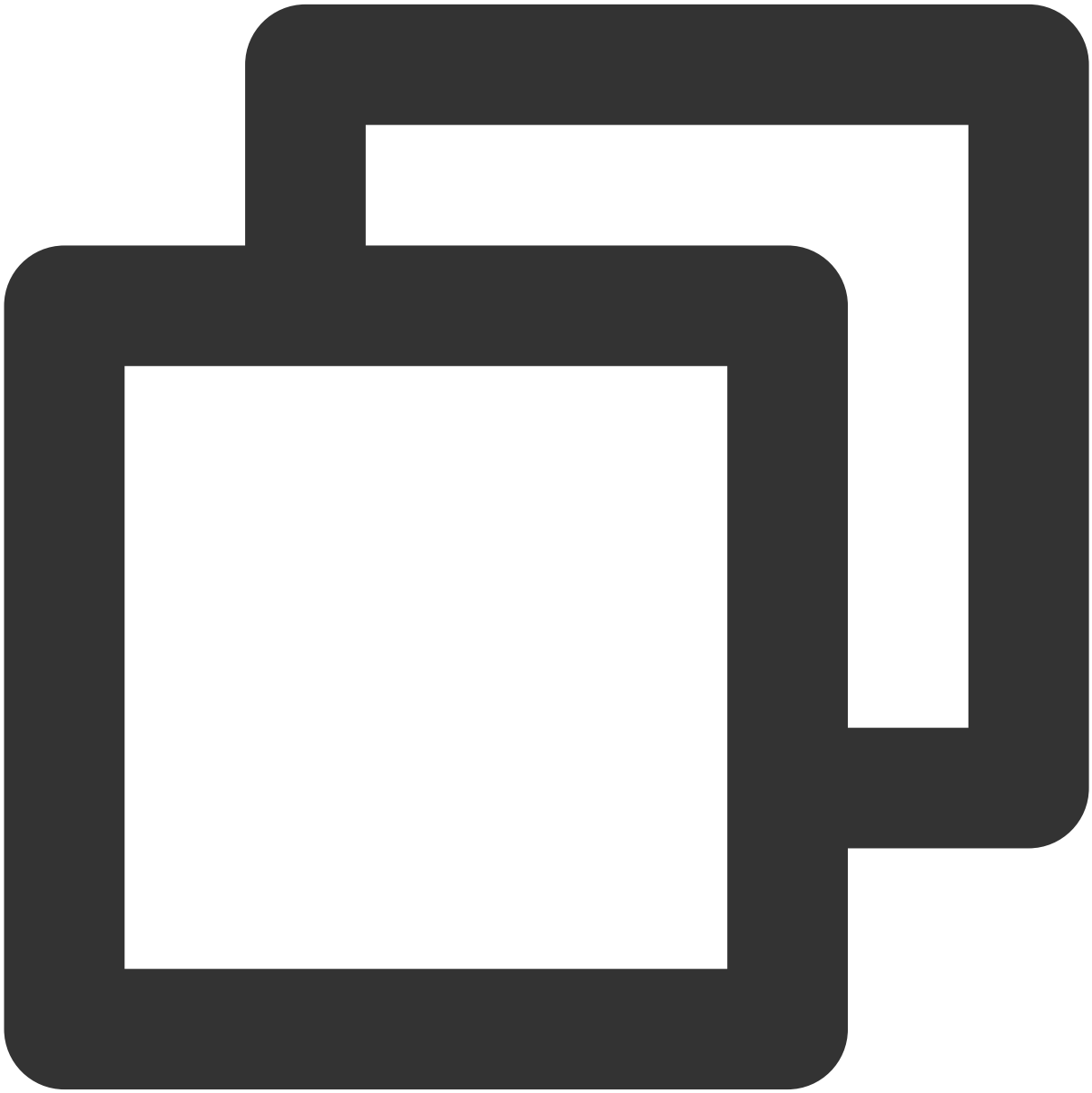
参数名称	参数描述
CIServiceResult	查询数据万象的结果

解绑数据万象服务

功能说明

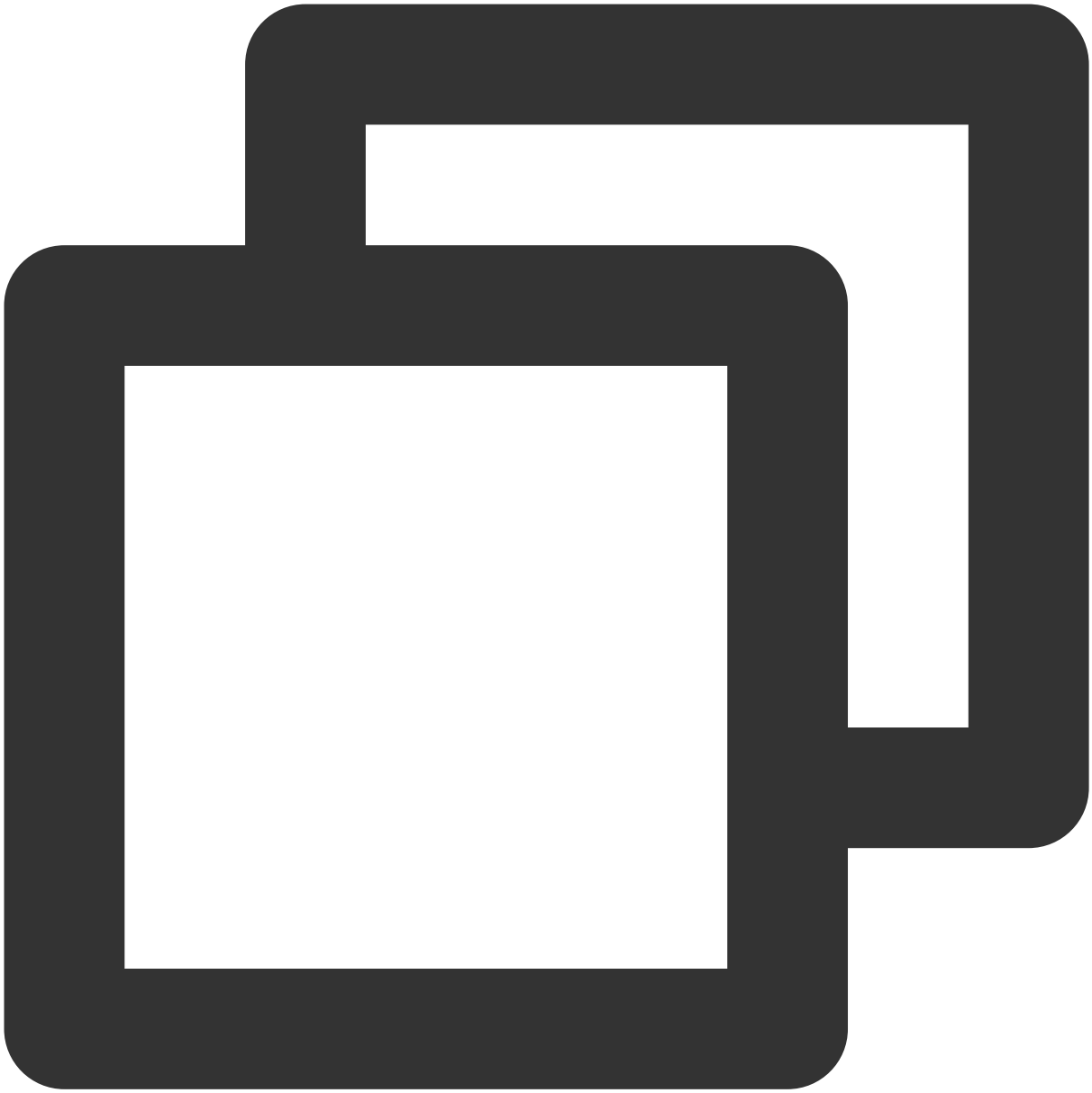
解绑开通数据万象服务的 Bucket。

方法原型



```
func (s *CIService) CloseCIService(ctx context.Context) (*Response, error)
```

请求示例



```
_, err := c.CI.CloseCIService(context.Background())
```

原图保护

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于原图保护的 API 概览以及 SDK 示例代码。

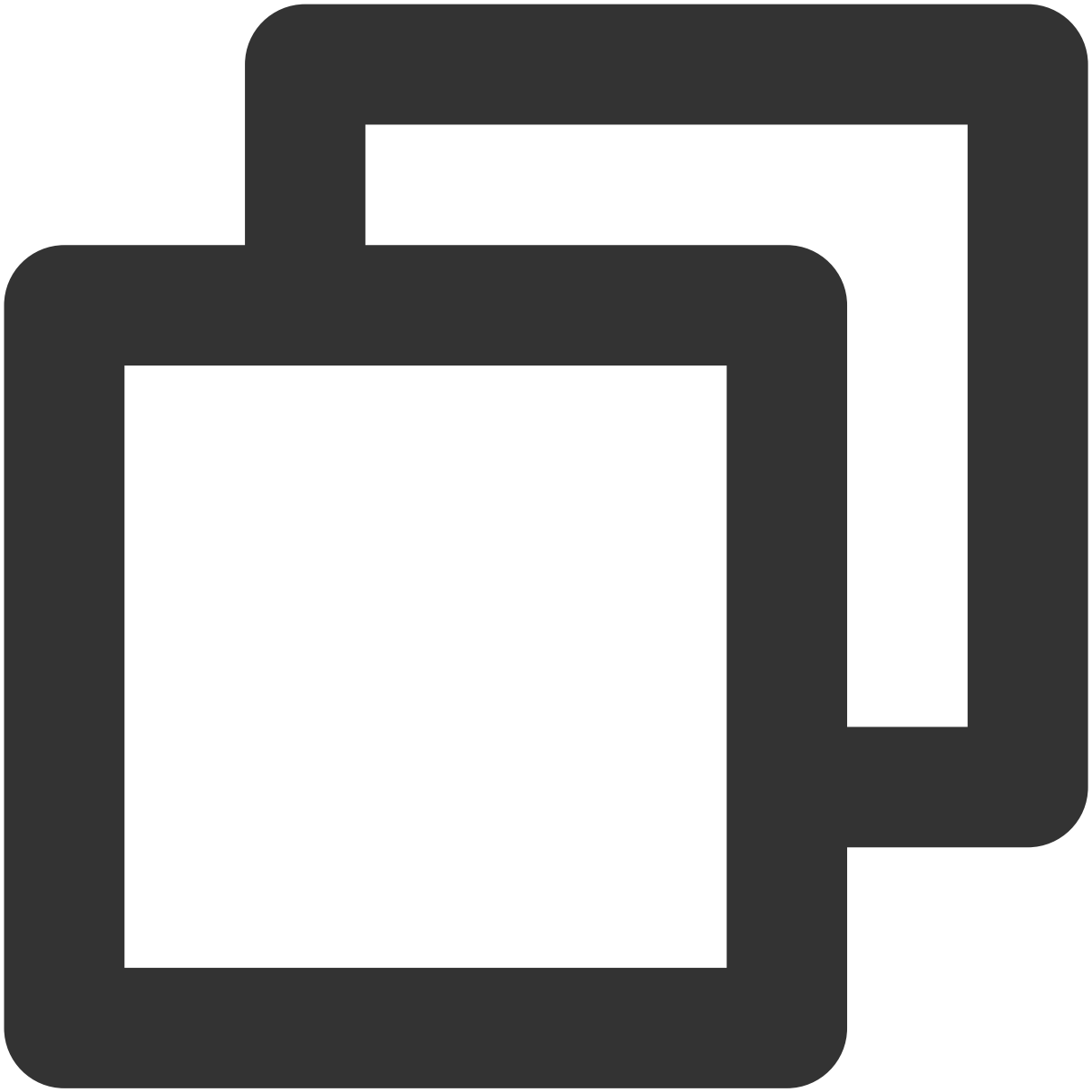
API	操作描述
开通原图保护	对 Bucket 开通原图保护功能。
查询原图保护状态	用于查询原图保护功能是否开启。
关闭原图保护	用于关闭原图保护功能。

开通原图保护

功能说明

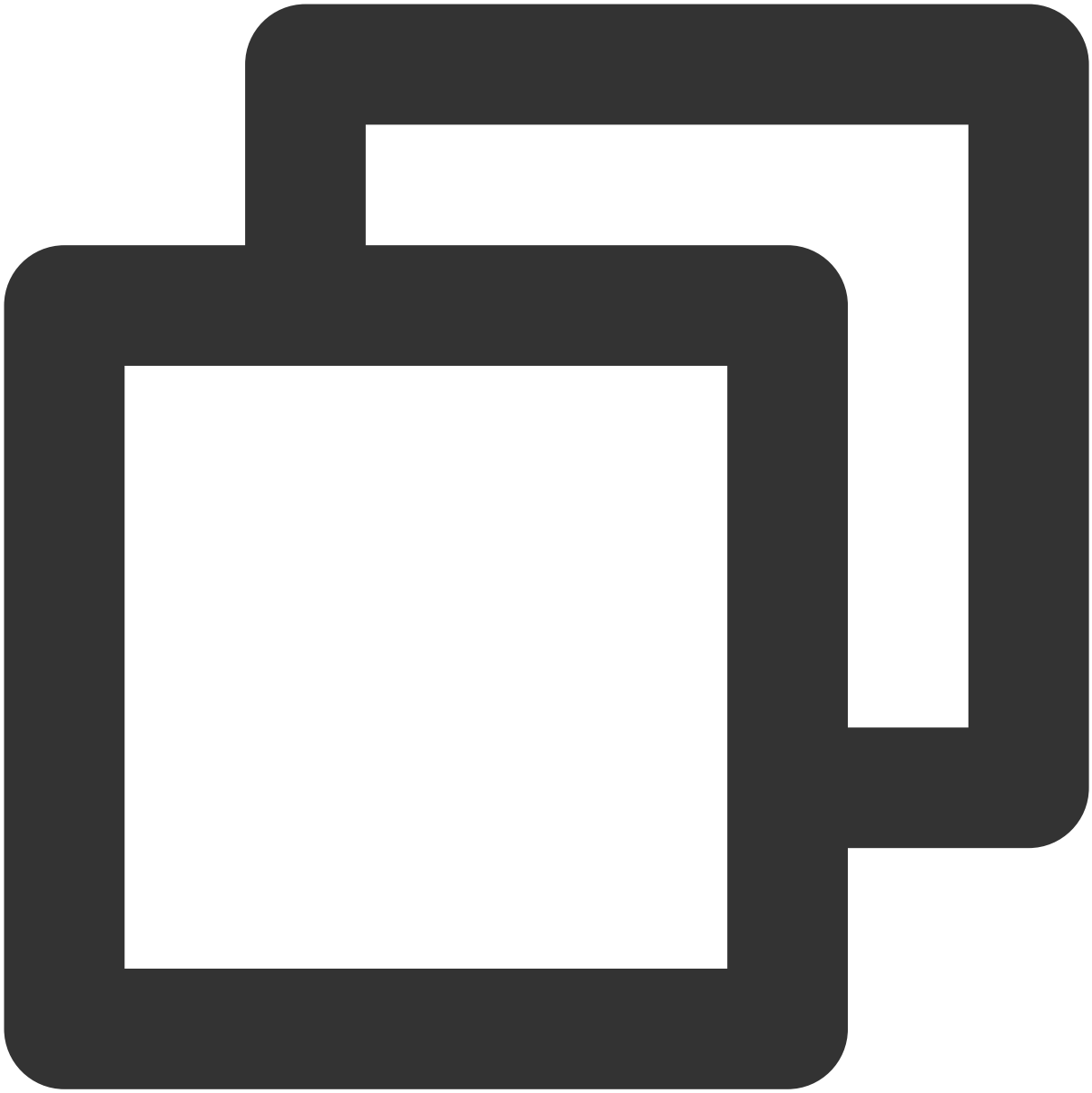
对 Bucket 开通原图保护功能。

方法原型



```
func (s *CIService) OpenOriginProtect(ctx context.Context) (*Response, error)
```

请求示例



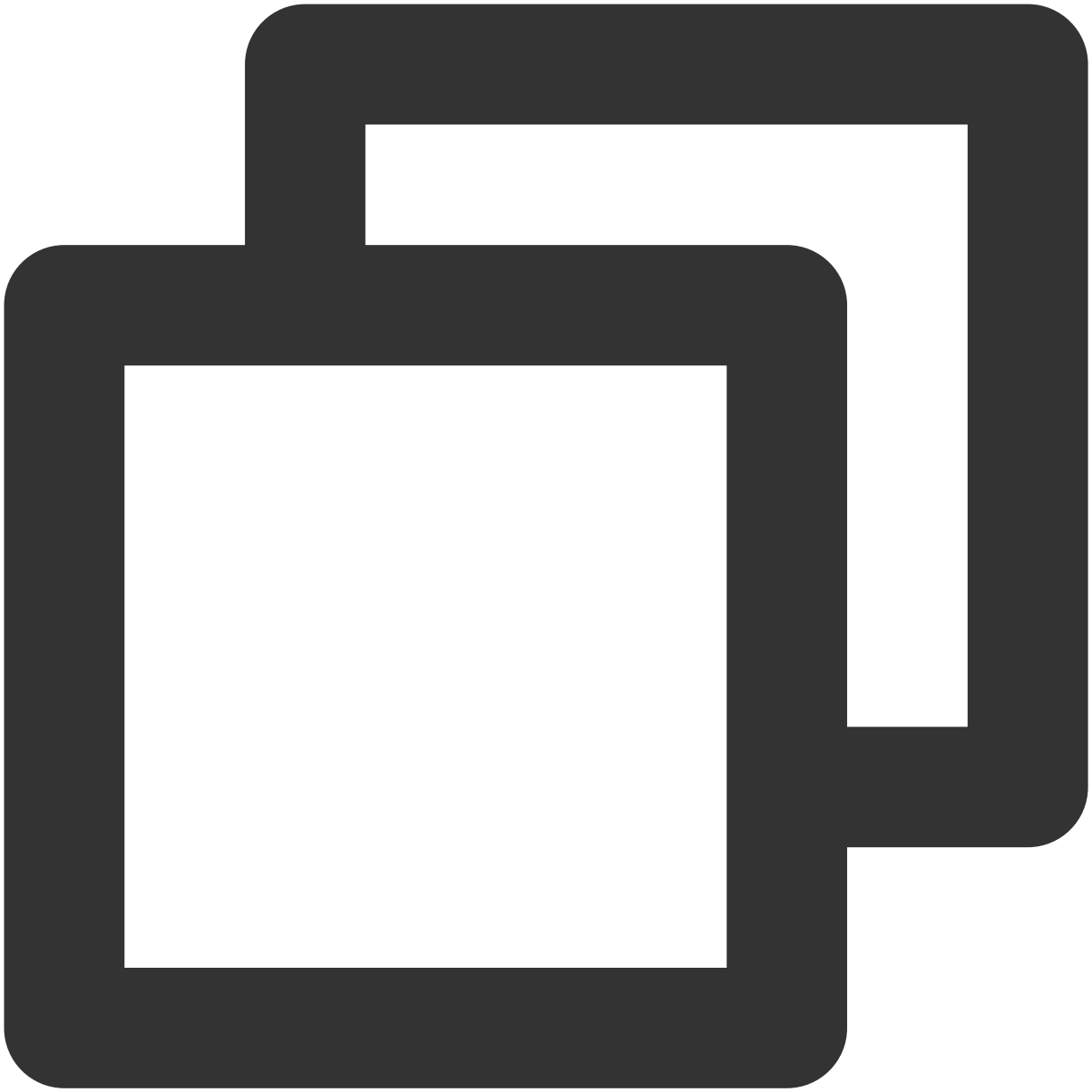
```
_, err := c.CI.OpenOriginProtect(context.Background())
```

查询原图保护状态

功能说明

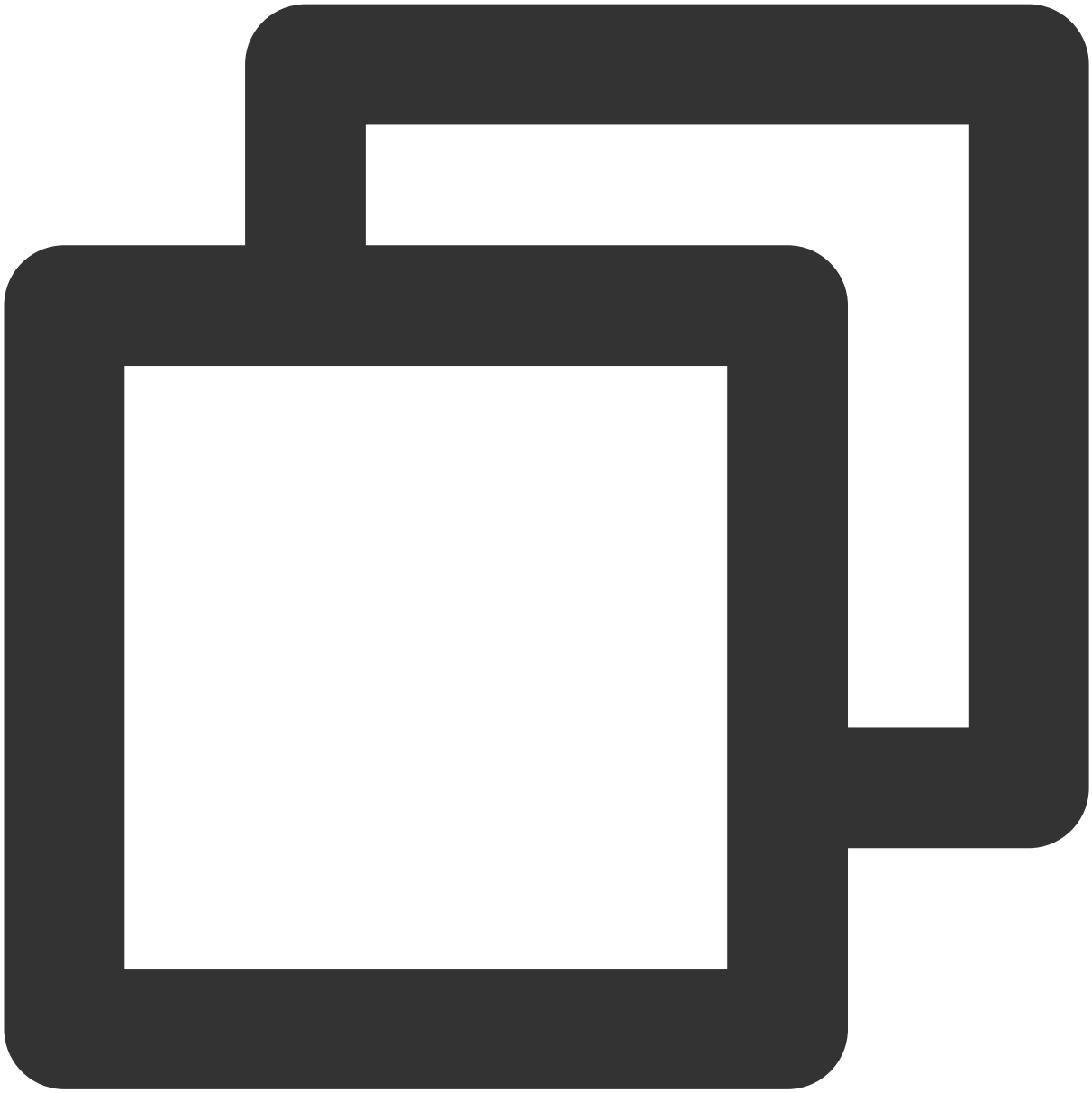
用于查询原图保护功能是否开启。

方法原型



```
func (s *CIService) GetOriginProtect(ctx context.Context) (*OriginProtectResult, *R
```

请求示例



```
res, _, err := c.CI.GetOriginProtect(context.Background())
```

参数说明

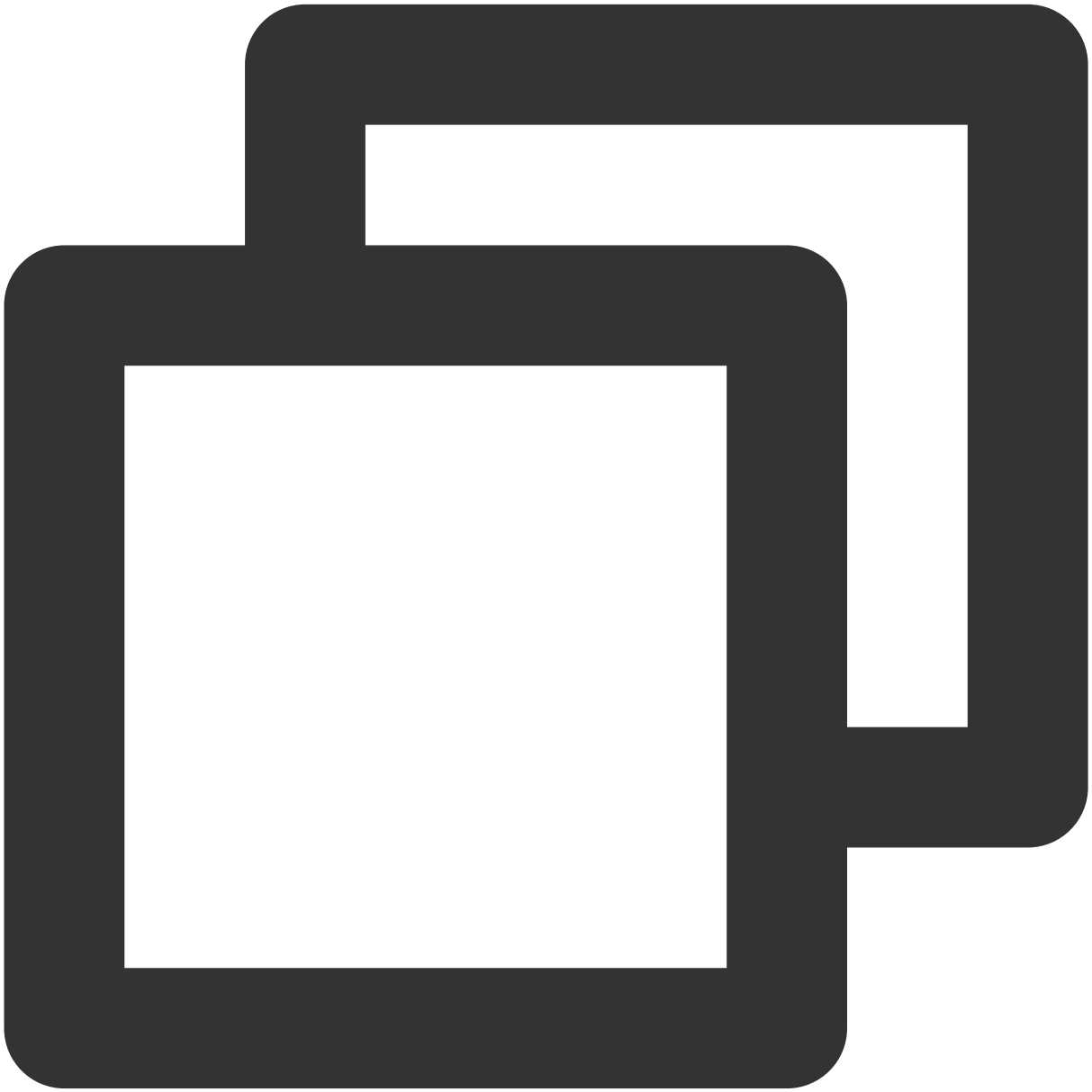
参数名称	参数描述
OriginProtectResult	查询原图保护的结果

关闭原图保护

功能说明

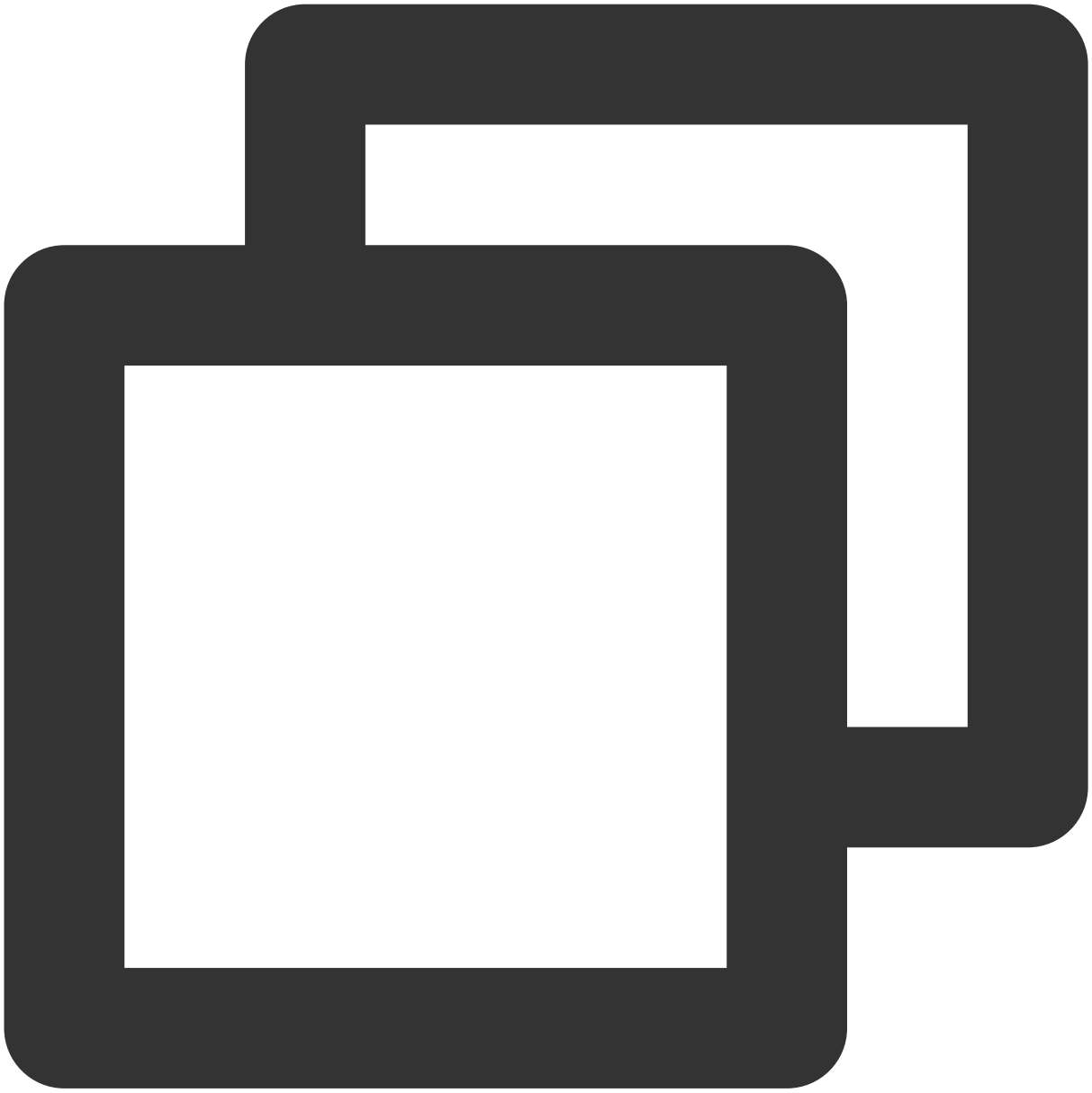
用于关闭原图保护功能。

方法原型



```
func (s *CIService) CloseOriginProtect(ctx context.Context) (*Response, error)
```

请求示例



```
_, err := c.CI.CloseOriginProtect(context.Background())
```

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体 bucket 接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

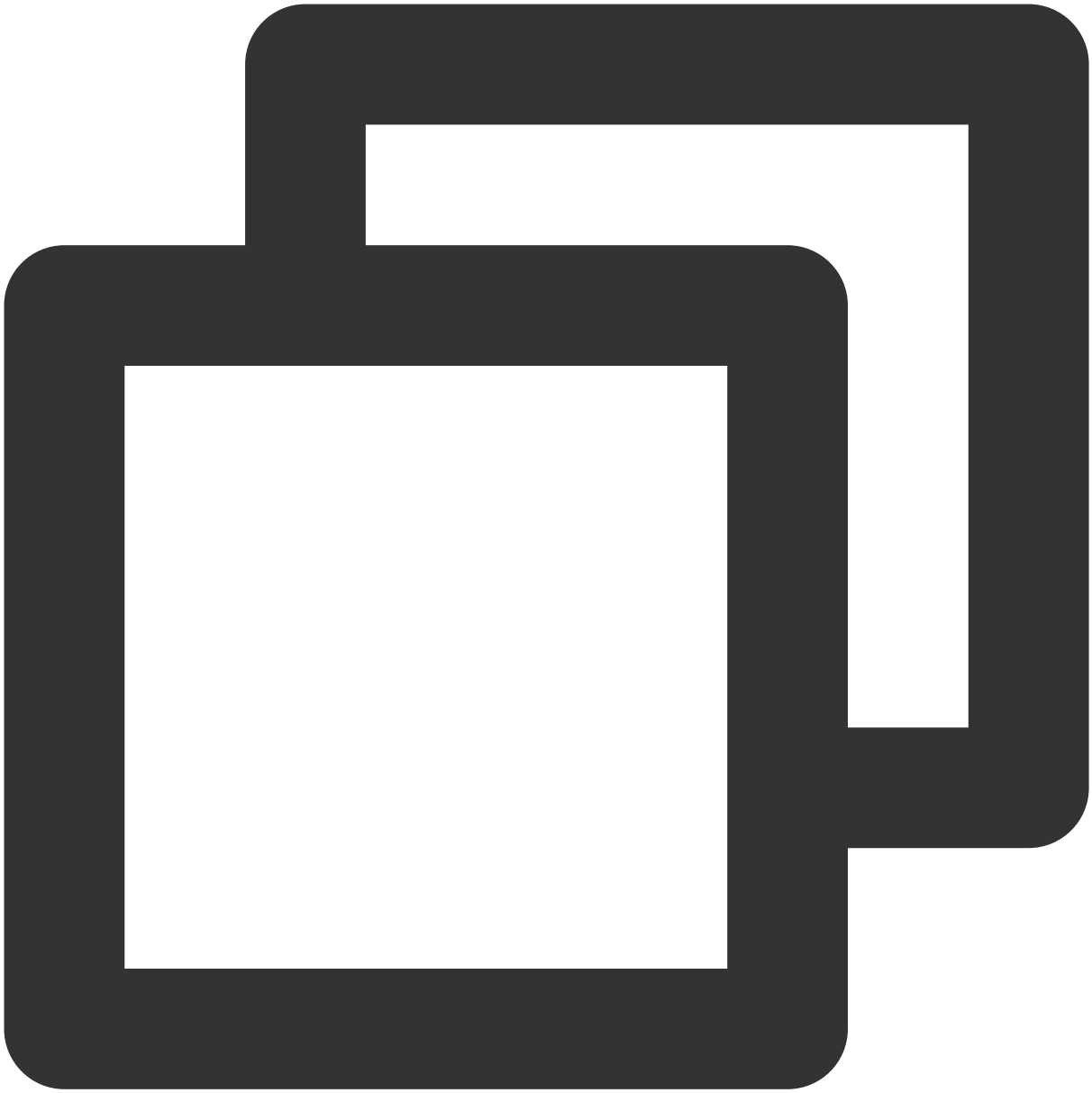
功能说明

用于查询已经开通媒体处理功能的存储桶。

注意：

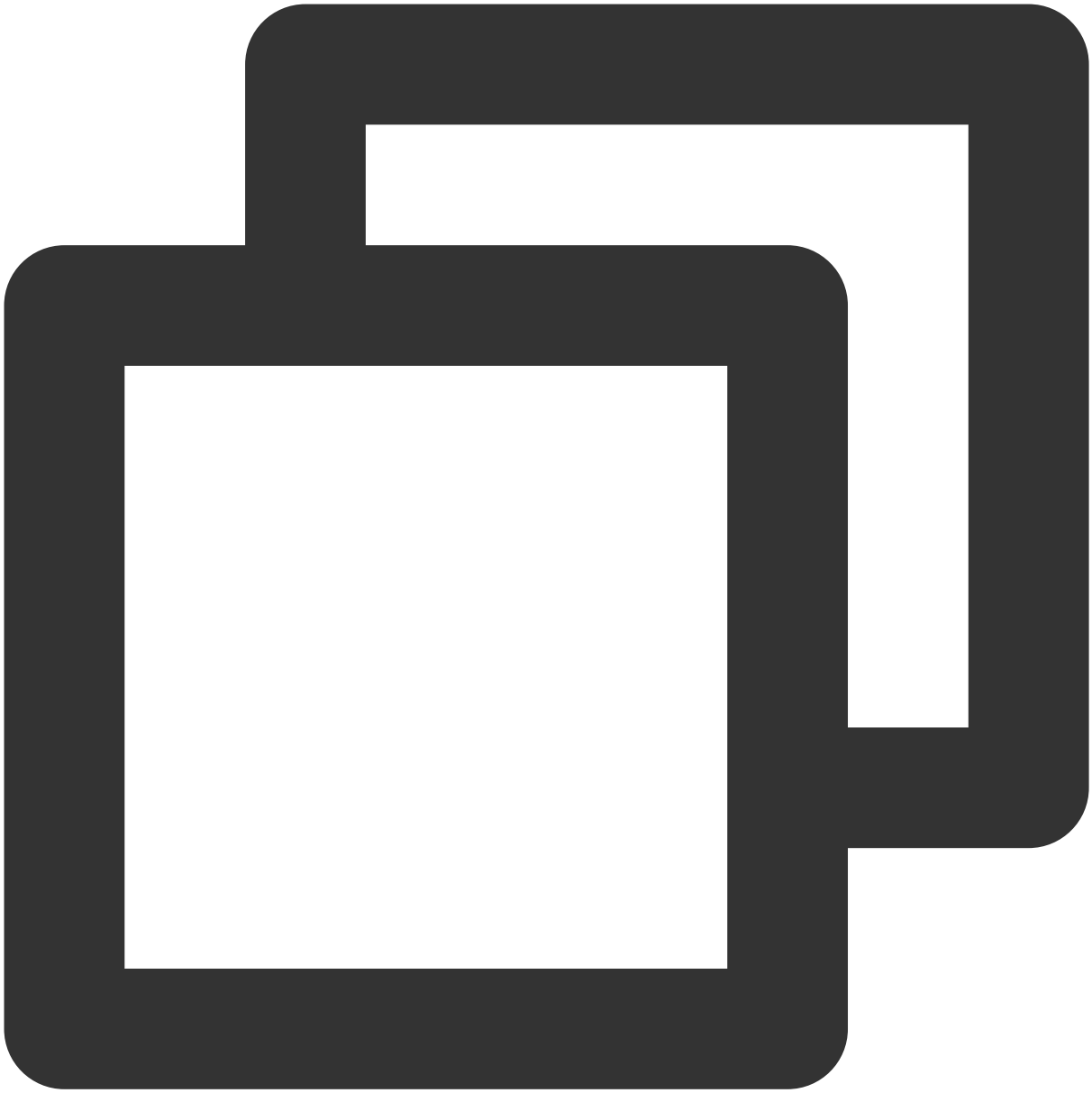
COS Go SDK 版本需要大于等于 v0.7.32。

方法原型



```
func (s *CIService) DescribeMediaProcessBuckets(ctx context.Context, opt *DescribeM
```

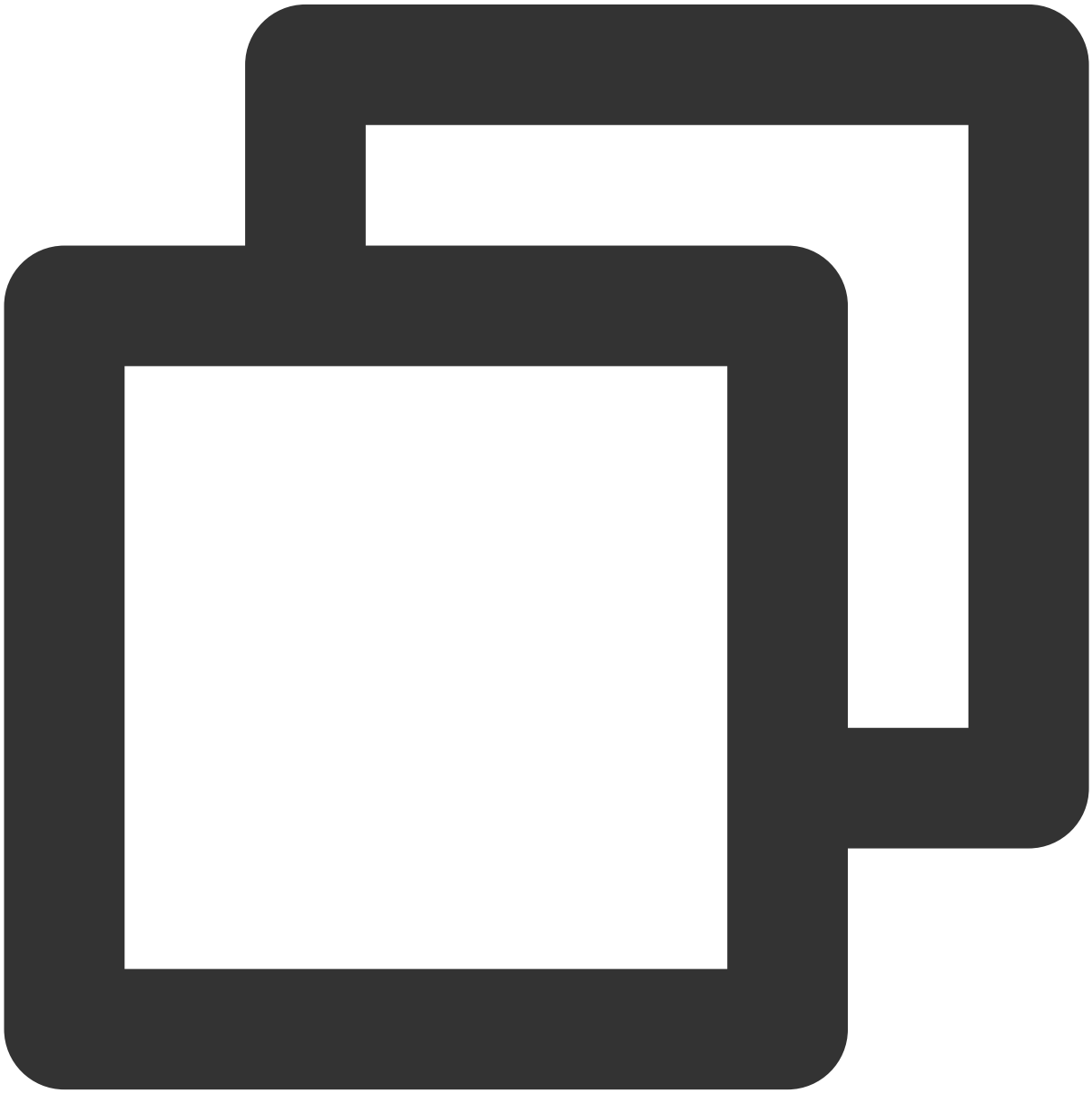
请求示例



```
// 需要设置 请求 URL 为 ci.<Region>.myqcloud.com
cu, _ := url.Parse("https://ci.ap-guangzhou.myqcloud.com")
b := &cos.BaseURL{CIURL: cu}
c := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        SecretID:  "SECRETID",
        SecretKey: "SECRETKEY",
    })

opt := &cos.DescribeMediaProcessBucketsOptions{
    Regions: "ap-guangzhou",
```

```
}
res, _, err := c.CI.DescribeMediaProcessBuckets(context.Background(), opt)
if err != nil {
    // ERROR
}
fmt.Printf("res: %+v\\n", res)
```

参数说明

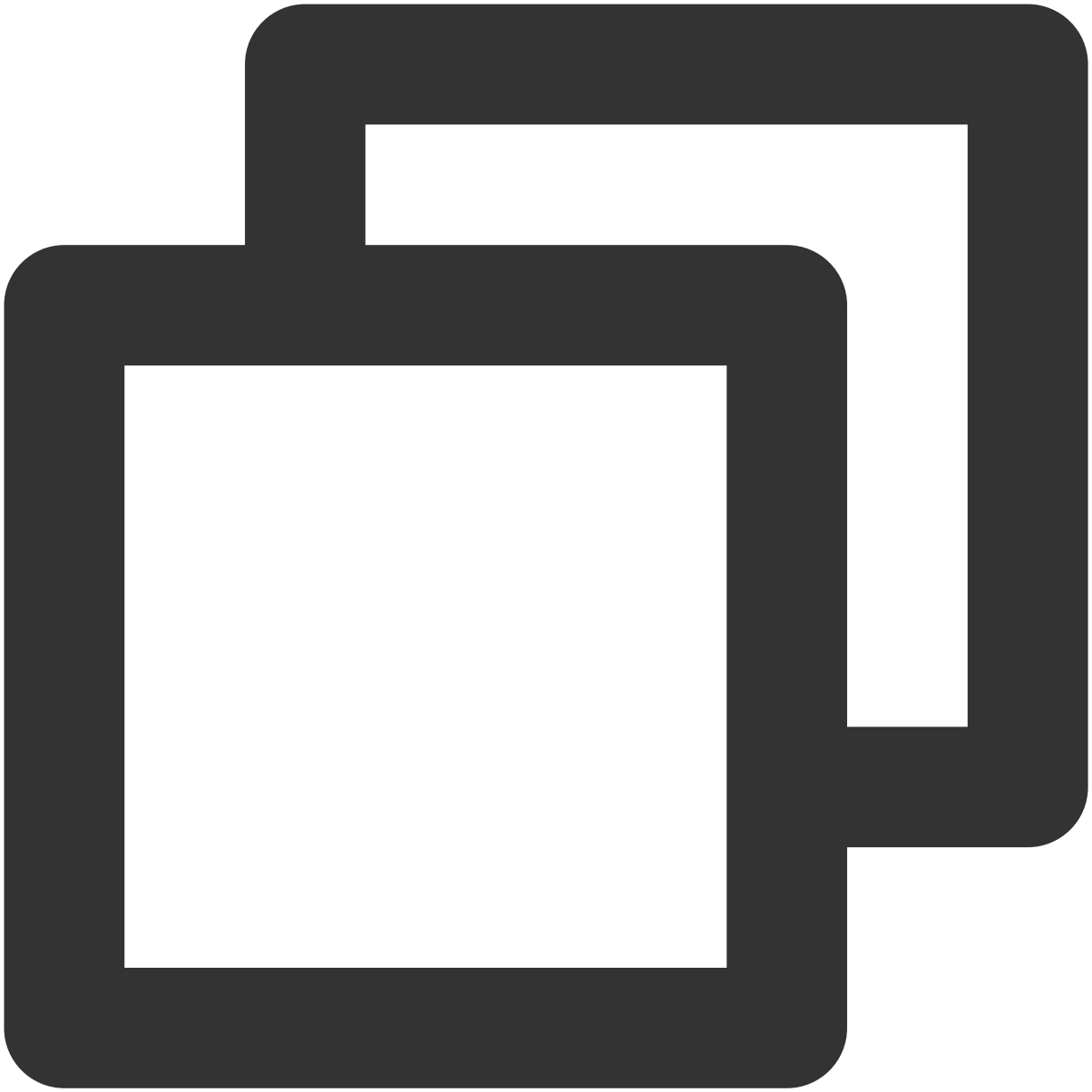


```
type DescribeMediaProcessBucketsOptions struct {
        Regions     string
```

```
        BucketNames string
        BucketName  string
        PageNumber  int
        PageSize    int
}
```

参数名称	参数描述	类型	是否必填
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	string	否
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string	否
BucketName	存储桶名称前缀，前缀搜索	string	否
PageNumber	第几页	string	否
PageSize	每页个数	string	否

返回结果说明



```
type DescribeMediaProcessBucketsResult struct {
        RequestId       string
        TotalCount      int
        PageNumber      int
        PageSize        int
        MediaBucketList []MediaProcessBucket
}
type MediaProcessBucket struct {
        BucketId   string
        Region     string
        CreateTime string
```

```
}
```

具体的数据内容如下：

节点名称（关键字）	父节点	描述	类型
RequestId	Response	请求的唯一 ID	String
TotalCount	Response	媒体 Bucket 总数	Int
PageNumber	Response	当前页数，同请求中的 pageNumber	Int
PageSize	Response	每页个数，同请求中的 pageSize	Int
MediaBucketList	Response	媒体 Bucket 列表	Container

Container 节点 MediaBucketList 的内容：

节点名称（关键字）	父节点	描述	类型
BucketId	Response.MediaBucketList	存储桶 ID	String
Name	Response.MediaBucketList	存储桶名称，同 BucketId	String
Region	Response.MediaBucketList	所在的地域	String
CreateTime	Response.MediaBucketList	创建时间	String

版权所有：腾讯云计算（北京）有限责任公司

媒体信息接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

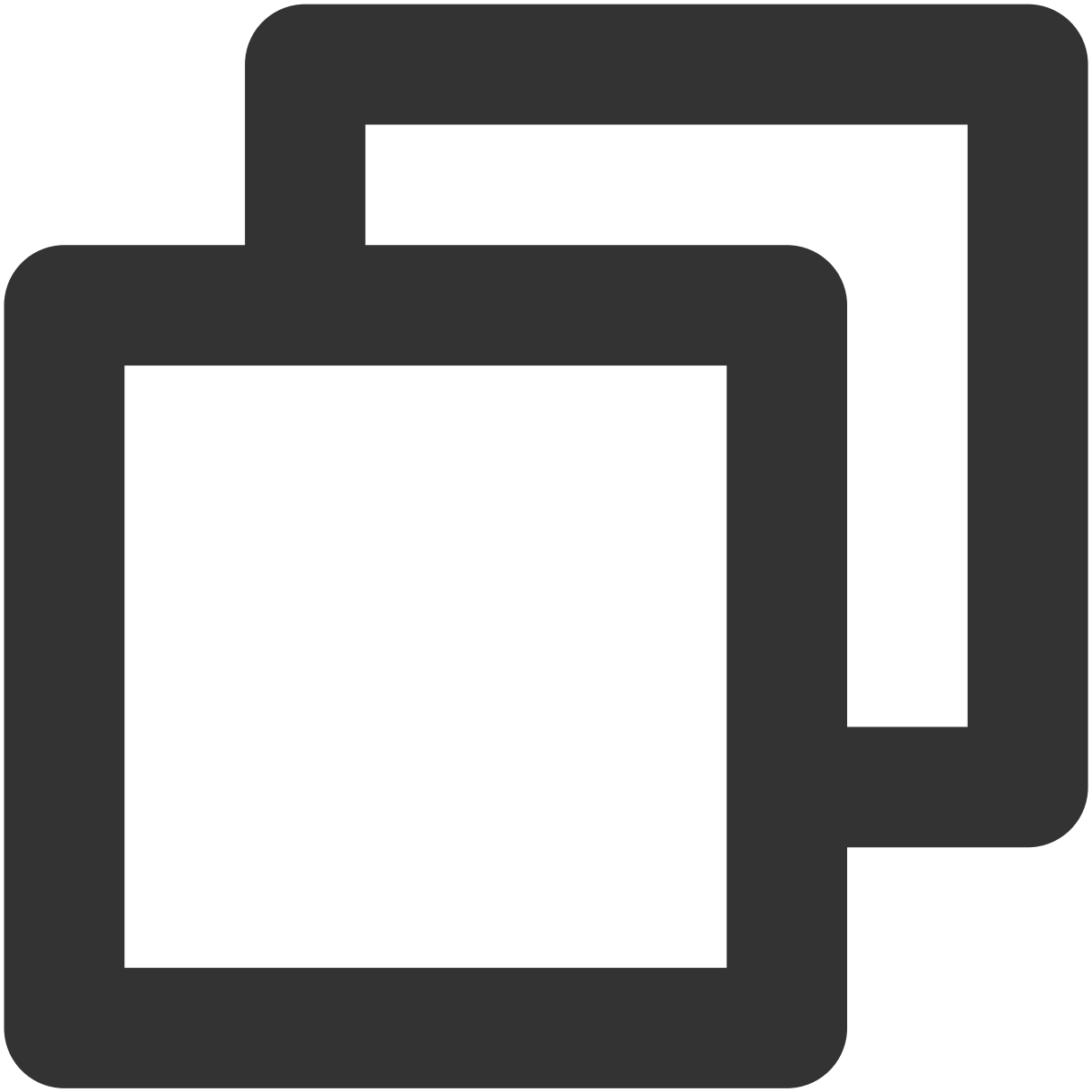
功能说明

用于查询媒体文件的信息。

注意：

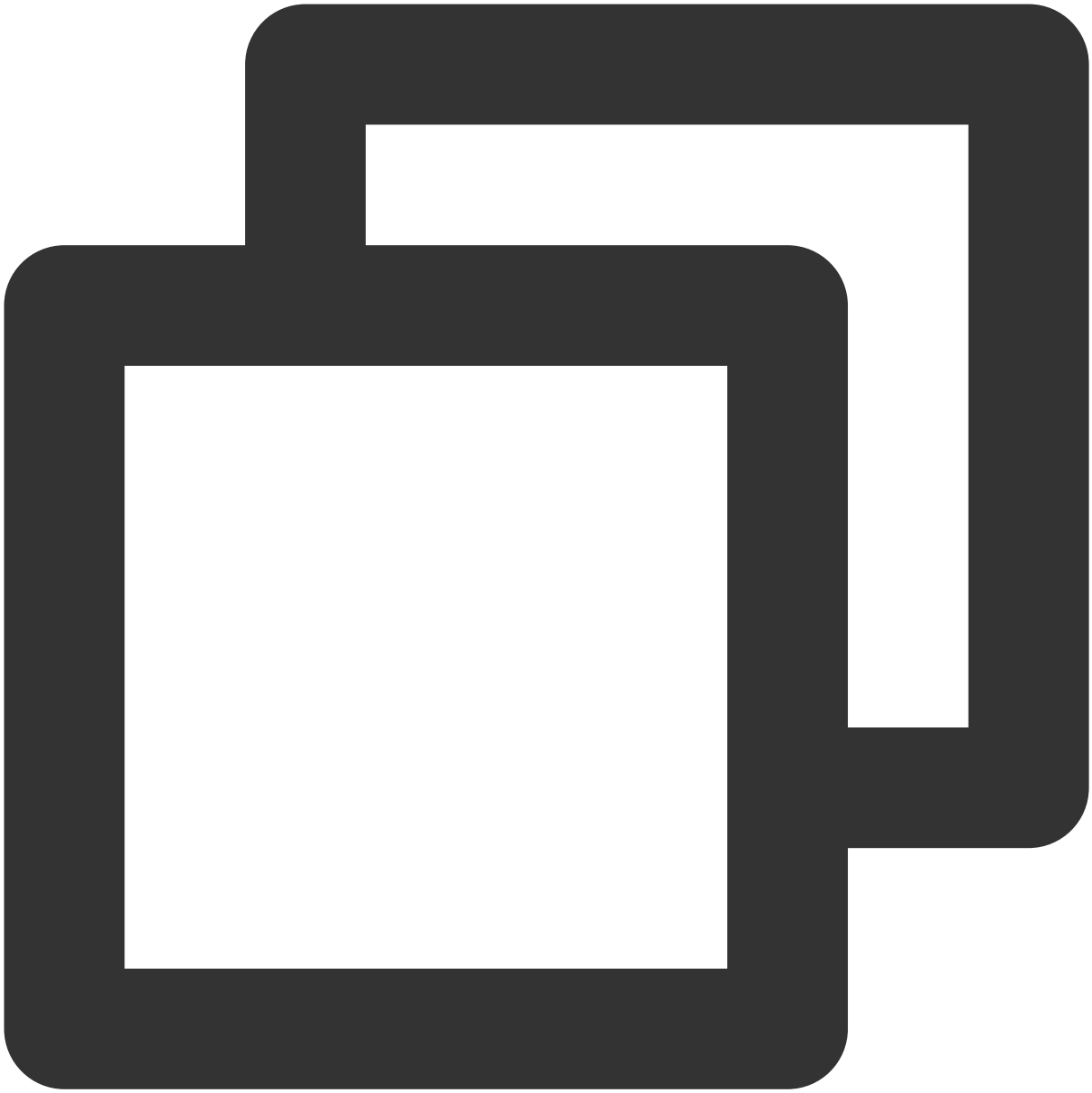
COS Go SDK 版本需要大于等于 v0.7.32。

方法原型



```
func (s *CIService) GetMediaInfo(ctx context.Context, name string, opt *ObjectGetOp
```

请求示例



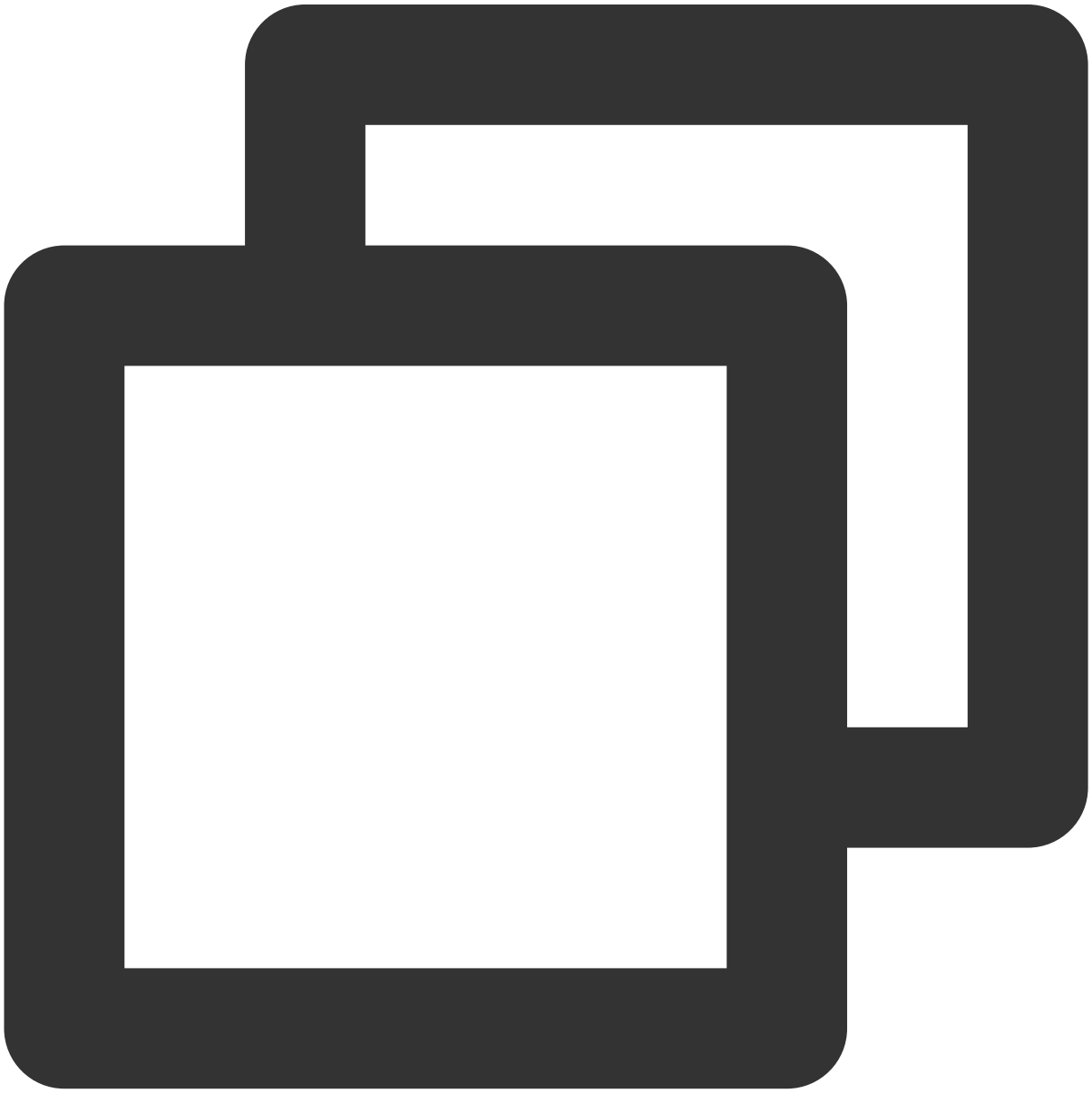
```
res, _, err := c.CI.GetMediaInfo(context.Background(), "test.mp4", nil)
if err != nil {
    // ERROR
}
fmt.Printf("res: %+v\\n", res)
```

参数说明

参数名称	参数描述	类型	是否必填

name	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	name	是
opt	公共请求头部和参数，详见 Get Object	struct	否
id	针对版本控制的对象 VersionId	string	否

返回结果说明



```
type GetMediaInfoResult struct {
    XMLName   xml.Name
    MediaInfo struct {
        Format struct {
            Bitrate        float32
            Duration       float32
            FormatLongName string
            FormatName     string
            NumProgram     int
            NumStream      int
            Size           int
            StartTime      float32
        }
        Stream struct {
            Audio []struct {
                Index          int
                CodecName      string
                CodecLongName  string
                CodecTimeBase  string
                CodecTagString string
                CodecTag       string
                SampleFmt      string
                SampleRate     int
                Channel        int
                ChannelLayout  string
                Timebase       string
                StartTime      float32
                Duration       float32
                Bitrate        float32
                Language       string
            }
            Subtitle struct {
                Index    int
                Language string
            }
            Video struct {
                Index          int
                CodecName      string
                CodecLongName  string
                CodecTimeBase  string
                CodecTagString string
                CodecTag       string
                Profile        string
                Height         int
                Width          int
                HasBFrame      int
                RefFrames      int
```

```
                Sar             string
                Dar             string
                PixFormat       string
                FieldOrder      string
                Level           int
                Fps             float32
                AvgFps          string
                Timebase        string
                StartTime       float32
                Duration        float32
                Bitrate         float32
                NumFrames       int
                Language        string
            }
        }
    }
}
```

具体的数据内容如下：

节点名称（关键字）	父节点	描述	类型
Response	无	保存结果的容器	Container

Container 节点 Response 的内容：

节点名称（关键字）	父节点	描述	类型
MediaInfo	Response	媒体的详细信息	Container

Container 节点 MediaInfo 的内容：

节点名称（关键字）	父节点	描述	类型
Stream	Response.MediaInfo	流信息	Container
Format	Response.MediaInfo	格式信息	Container

Container 节点 Stream 的内容：

节点名称（关键字）	父节点	描述	类型
Video	Response.MediaInfo. Stream	视频信息	Container
Audio	Response.MediaInfo. Stream	音频信息	Container
Subtitle	Response.MediaInfo. Stream	字幕信息	Container

版权所有：腾讯云计算（北京）有限责任公司

Container 节点 Format 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
NumStream	Response.MediaInfo.Format	Stream（包含 Video、Audio、Subtitle）的数量	Int
NumProgram	Response.MediaInfo.Format	节目的数量	Int
FormatName	Response.MediaInfo.Format	容器格式名字	String
FormatLongName	Response.MediaInfo.Format	容器格式的详细名字	String
StartTime	Response.MediaInfo.Format	起始时间，单位为秒	Float
Duration	Response.MediaInfo.Format	时长，单位为秒	Float
Bitrate	Response.MediaInfo.Format	比特率，单位为 kbps	Int
Size	Response.MediaInfo.Format	大小，单位为 Byte	Int

Container 节点 Video 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
Index	Response.MediaInfo. Stream.Video	该流的编号	Int
CodecName	Response.MediaInfo. Stream.Video	编解码格式名字	String
CodecLongName	Response.MediaInfo. Stream.Video	编解码格式的详细名字	String
CodecTimeBase	Response.MediaInfo. Stream.Video	编码时基	String
CodecTagString	Response.MediaInfo. Stream.Video	编码标签名	String
CodecTag	Response.MediaInfo. Stream.Video	编码标签	String
Profile	Response.MediaInfo. Stream.Video	视频编码档位	String
Height	Response.MediaInfo. Stream.Video	视频高，单位 px	Int

版权所有：腾讯云计算（北京）有限责任公司

Width	Response.MediaInfo. Stream.Video	视频宽，单位 px	Int
HasBFrame	Response.MediaInfo. Stream.Video	是否有B帧。1表示有，0表示无	Int
RefFrames	Response.MediaInfo. Stream.Video	视频编码的参考帧个数	Int
Sar	Response.MediaInfo. Stream.Video	采样宽高比	String
Dar	Response.MediaInfo. Stream.Video	显示宽高比	String
PixFormat	Response.MediaInfo. Stream.Video	像素格式	String
FieldOrder	Response.MediaInfo. Stream.Video	场的顺序	String
Level	Response.MediaInfo. Stream.Video	视频编码等级	Int
Fps	Response.MediaInfo. Stream.Video	视频帧率	Int
AvgFps	Response.MediaInfo. Stream.Video	平均帧率	String
Timebase	Response.MediaInfo. Stream.Video	时基	String
StartTime	Response.MediaInfo. Stream.Video	视频开始时间，单位为秒	Float
Duration	Response.MediaInfo. Stream.Video	视频时长，单位为秒	Float
Bitrate	Response.MediaInfo. Stream.Video	比特率，单位为 kbps	Float
NumFrames	Response.MediaInfo. Stream.Video	总帧数	Int
Language	Response.MediaInfo. Stream.Video	语言	String

Container 节点 Audio 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
Index	Response.MediaInfo. Stream.Audio	该流的编号	Int
CodecName	Response.MediaInfo. Stream.Audio	编解码格式名字	String
CodecLongName	Response.MediaInfo. Stream.Audio	编解码格式的详细名字	String
CodecTimeBase	Response.MediaInfo. Stream.Audio	编码时基	String
CodecTagString	Response.MediaInfo. Stream.Audio	编码标签名	String
CodecTag	Response.MediaInfo. Stream.Audio	编码标签	String
SampleFmt	Response.MediaInfo. Stream.Audio	采样格式	String

版权所有：腾讯云计算（北京）有限责任公司

SampleRate	Response.MediaInfo. Stream.Audio	采样率	Int
Channel	Response.MediaInfo. Stream.Audio	通道数量	Int
ChannelLayout	Response.MediaInfo. Stream.Audio	通道格式	String
Timebase	Response.MediaInfo. Stream.Audio	时基	String
StartTime	Response.MediaInfo. Stream.Audio	音频开始时间，单位秒	Float
Duration	Response.MediaInfo. Stream.Audio	音频时长，单位秒	Float
Bitrate	Response.MediaInfo. Stream.Audio	比特率，单位 kbps	Float
Language	Response.MediaInfo. Stream.Audio	语言	String

Container 节点 Subtitle 的内容：

节点名称（关键字）	父节点	描述	类型
Index	Response.MediaInfo. Stream.Subtitle	该流的编号	Int
Language	Response.MediaInfo. Stream.Subtitle	语言，und 表示无查询结果	String

版权所有：腾讯云计算（北京）有限责任公司

媒体截图接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

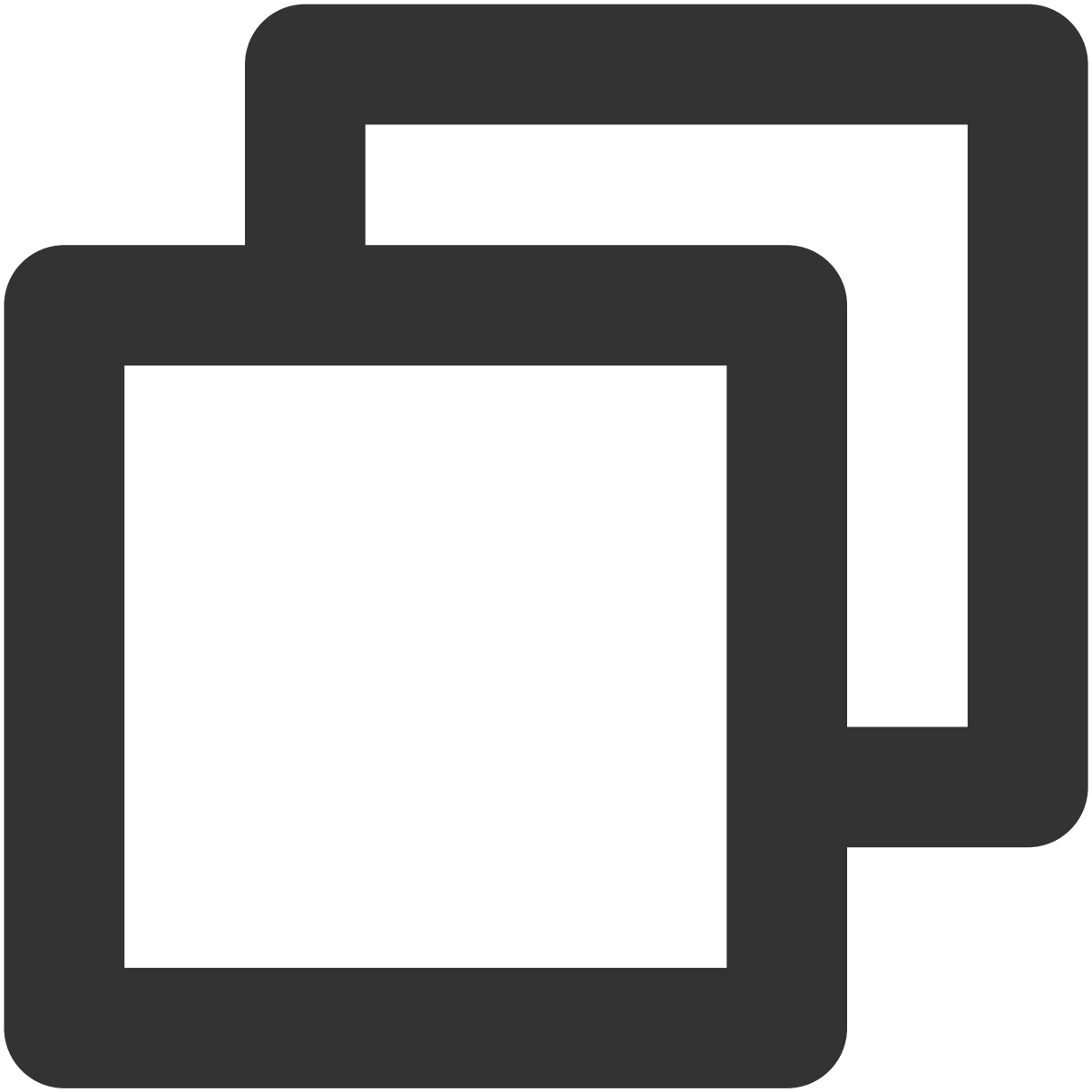
功能说明

用于查询媒体文件在某个时间的截图。

注意：

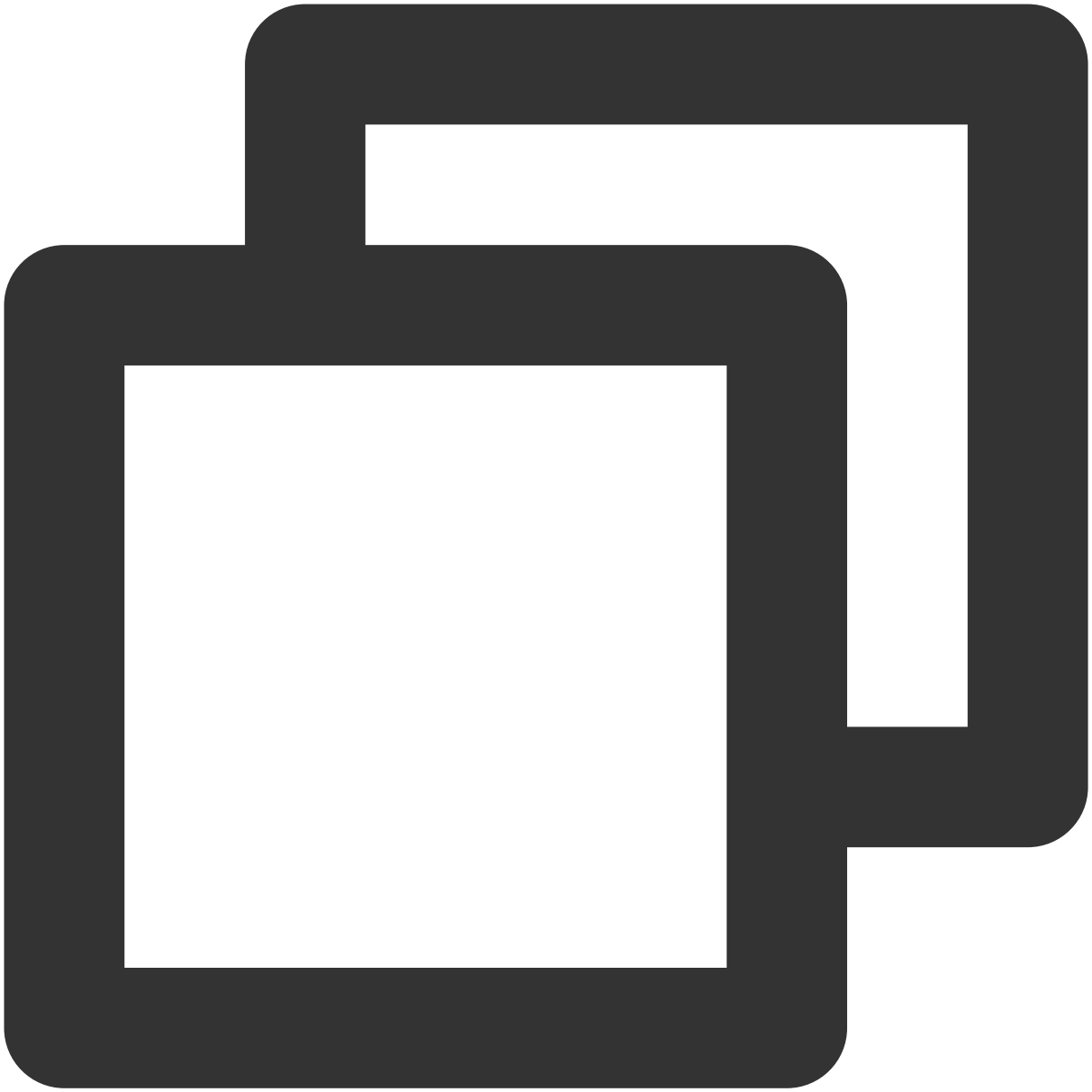
COS Go SDK 版本需要大于等于 v0.7.32。

方法原型



```
func (s *CIService) GetSnapshot(ctx context.Context, name string, opt *GetSnapshotO
```

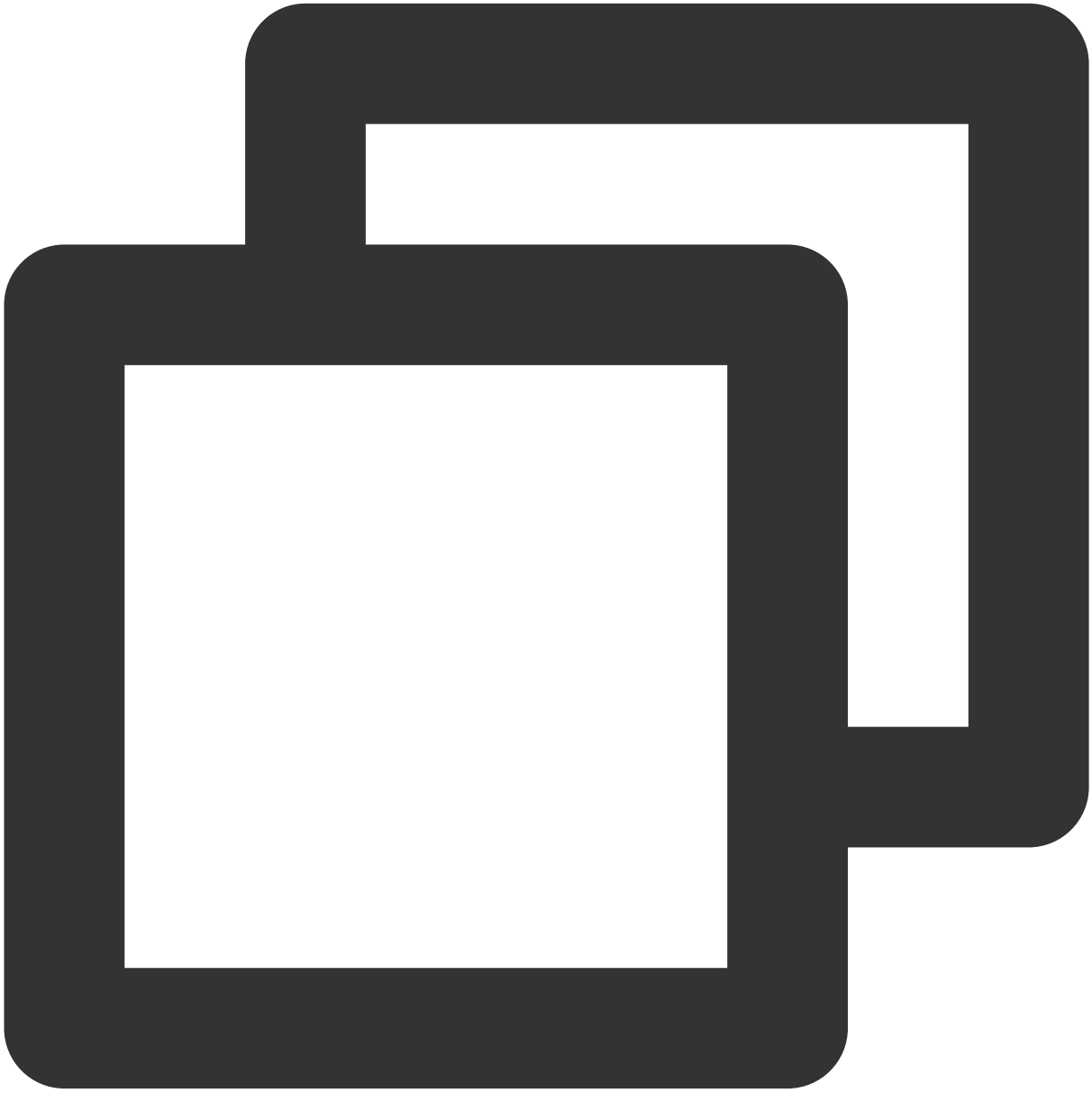
请求示例



```
opt := &cos.GetSnapshotOptions{
    Time: 1,
}
resp, err := c.CI.GetSnapshot(context.Background(), "test.mp4", opt)
if err != nil {
    // ERROR
}

fd, err := os.OpenFile("test.jpg", os.O_WRONLY|os.O_CREATE|os.O_TRUNC, 0660)
if err != nil {
    // ERROR
```

```
}
_, err = io.Copy(fd, resp.Body)
fd.Close()
```

参数说明



```
type GetSnapshotOptions struct {
    Time   float32
    Height int
    Width  int
    Format string
```

```
    Rotate string
    Mode   string
}
```

参数名称	参数描述	是否必填	类型
name	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	是	name
opt	截图参数	否	struct
id	针对版本控制的对象 VersionId	否	String
Time	截图的时间点，单位为秒	是	float
Width	截图的宽。默认为0	否	Int
Height	截图的高。默认为0 当 width 和 height 都为0时，表示使用视频的宽高；如果单个为0，则以另外一个值按视频宽高比例自动适应	否	Int
Format	截图的格式，支持 jpg 和 png，默认 jpg	否	String
Rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转off：不旋转默认值为 auto	否	String
Mode	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值为 exactframe	否	String

返回结果说明

参数名称	参数描述	是否必填	类型
Response	http响应	是	Struct
Response.Header	http响应头部	是	Struct
Response.Body	http响应数据	是	Struct

版权所有：腾讯云计算（北京）有限责任公司

文档预览
查询文档预览开通状态

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于查询文档预览开通状态的 API 概览以及 SDK 示例代码。

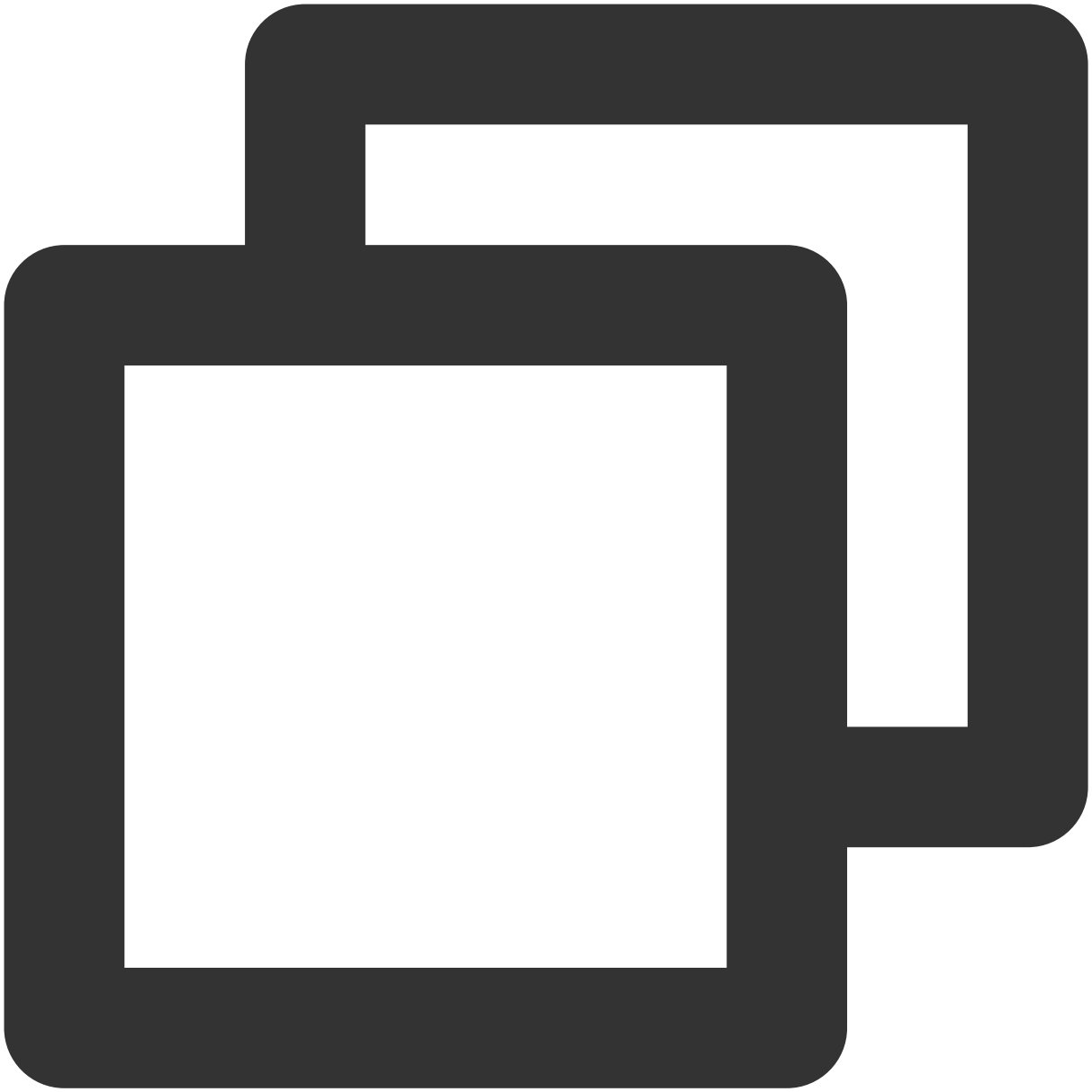
API	说明
查询文档预览开通状态	用于查询存储桶是否已开通文档预览功能

查询文档预览开通状态

功能说明

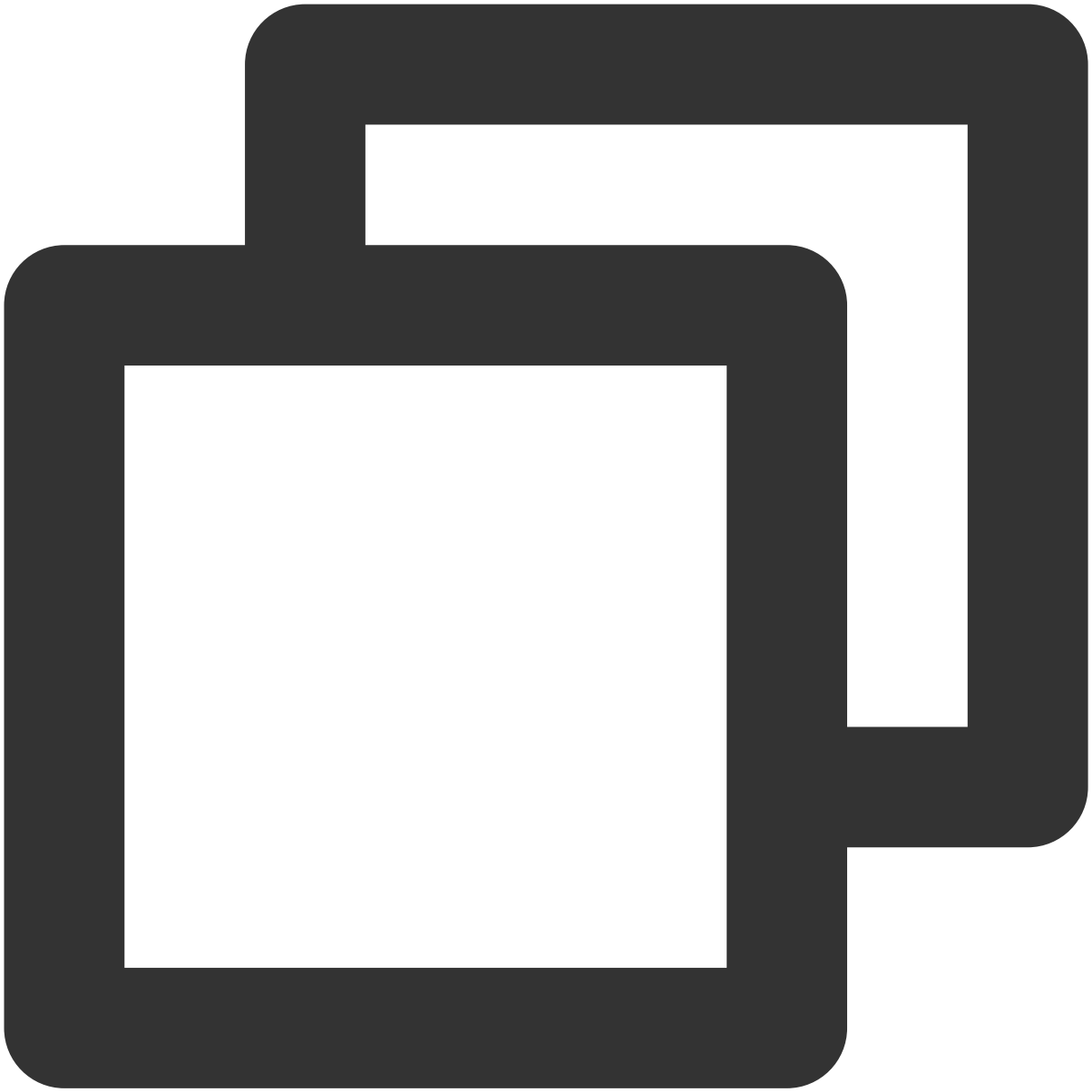
DescribeDocProcessBuckets 接口用于查询存储桶是否已开通文档预览功能。

方法原型



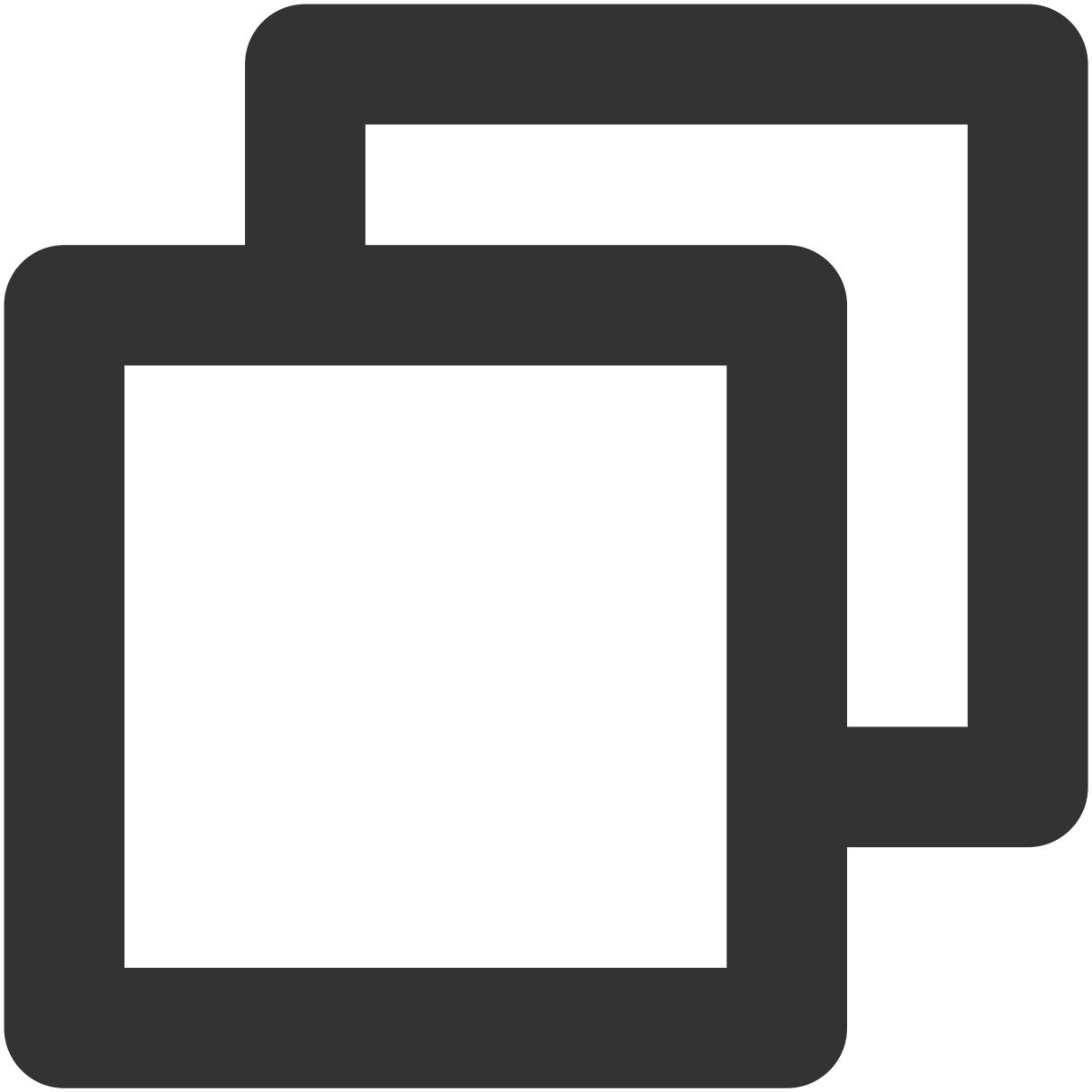
```
func (s *CIService) DescribeDocProcessBuckets(ctx context.Context, opt *DescribeDoc
```

请求示例



```
BucketsOpt := &cos.DescribeDocProcessBucketsOptions{
    Regions: "ap-shanghai",
}
res, _, err := c.CI.DescribeDocProcessBuckets(context.Background(), BucketsOpt)
```

参数说明

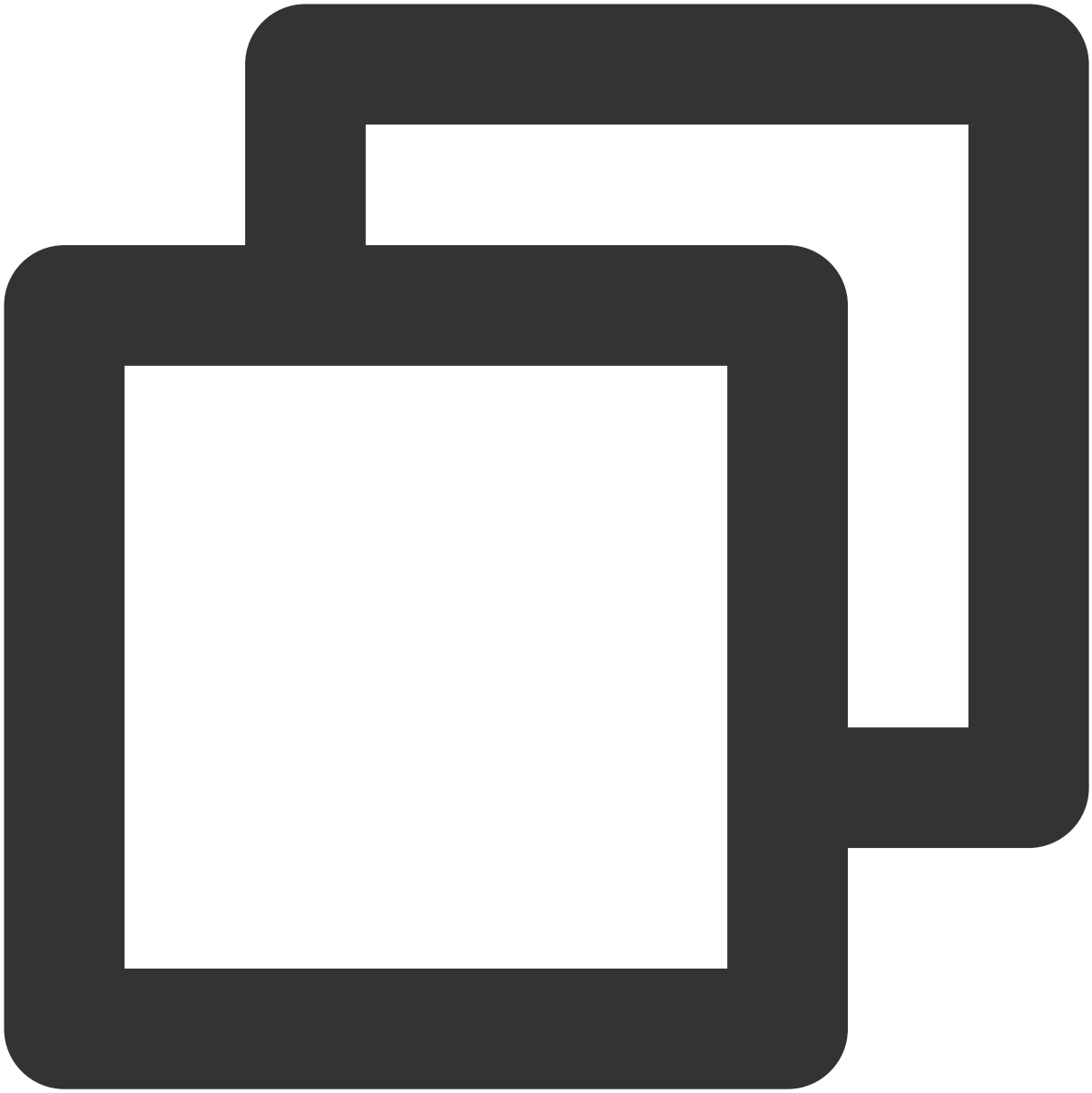


```
type DescribeDocProcessBucketsOptions struct {
    Regions     string
    BucketNames string
    BucketName  string
    PageNumber  int
    PageSize    int
}
```

参数名称	描述	类型

Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	string
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string
BucketName	存储桶名称前缀，前缀搜索	string
PageNumber	第几页	int
pageSize	每页个数	int

返回结果说明



```
type DescribeDocProcessBucketsResult struct {
    RequestId     string
    TotalCount    int
    PageNumber    int
    PageSize      int
    DocBucketList []DocProcessBucket
}
type DocProcessBucket struct {
    BucketId      string
    Name          string
    Region        string
```

```
    CreateTime    string
    AliasBucketId string
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	string
TotalCount	文档预览 Bucket 总数	int
PageNumber	当前页数，同请求中的 pageNumber	int
PageSize	每页个数，同请求中的 pageSize	int
DocBucketList	文档预览 Bucket 列表	Container
BucketId	存储桶 ID	string
Name	存储桶名称，同 BucketId	string
Region	所在的地域	string
CreateTime	创建时间	string
AliasBucketId	存储桶别名	string

同步请求接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于文档预览同步请求的 API 概览以及 SDK 示例代码。

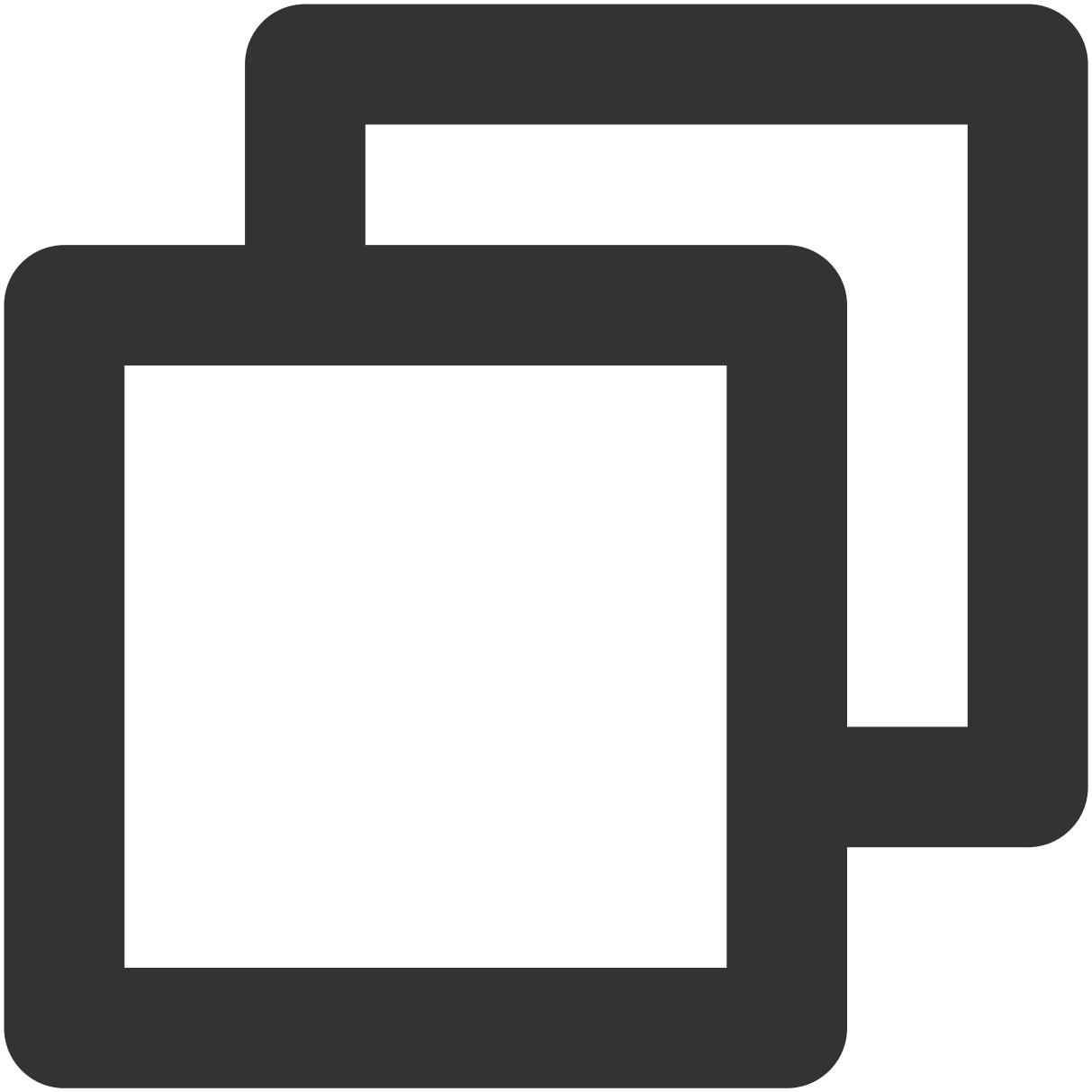
API	说明
同步请求接口	用于同步请求文档预览功能

同步请求接口

功能说明

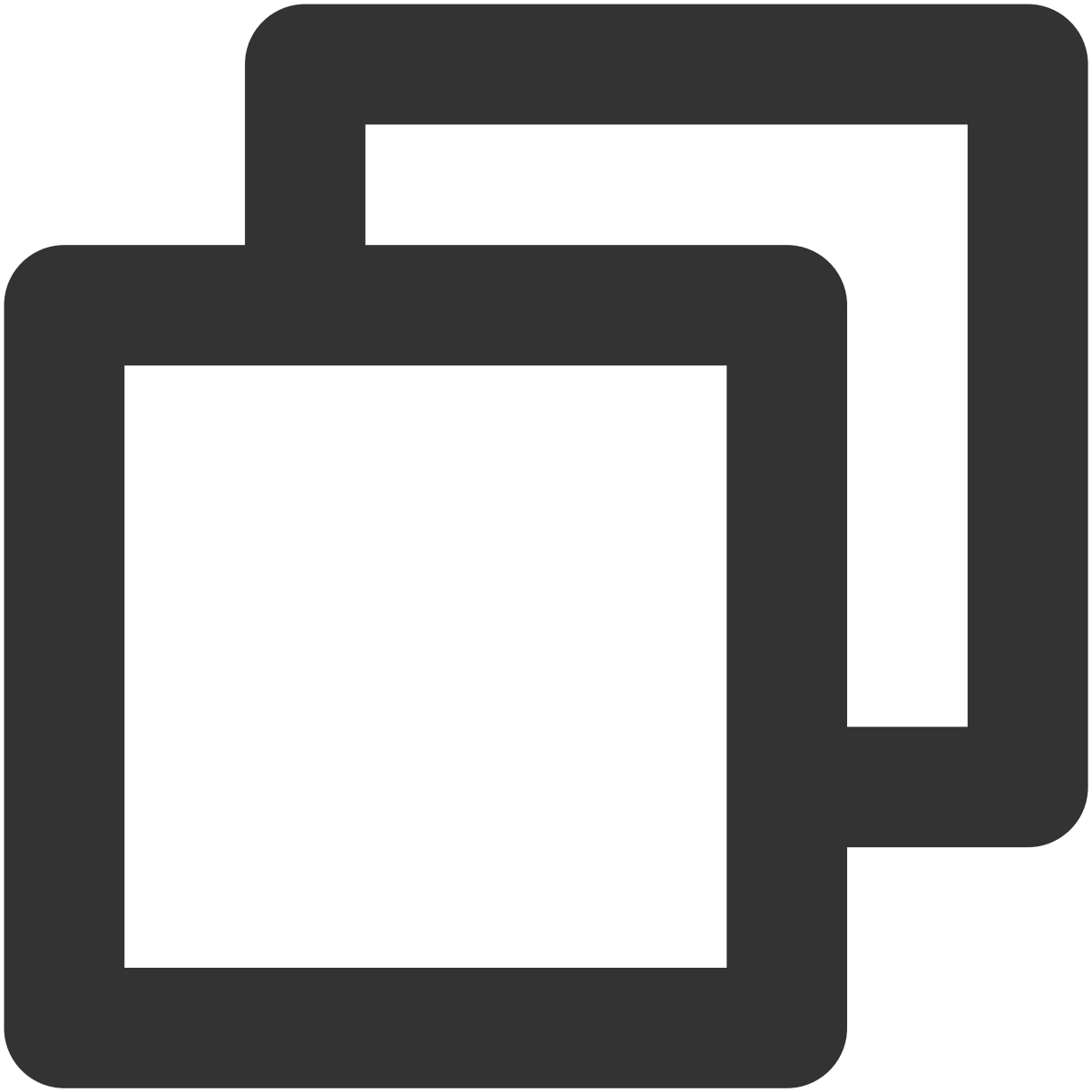
文档预览功能同步请求。

方法原型



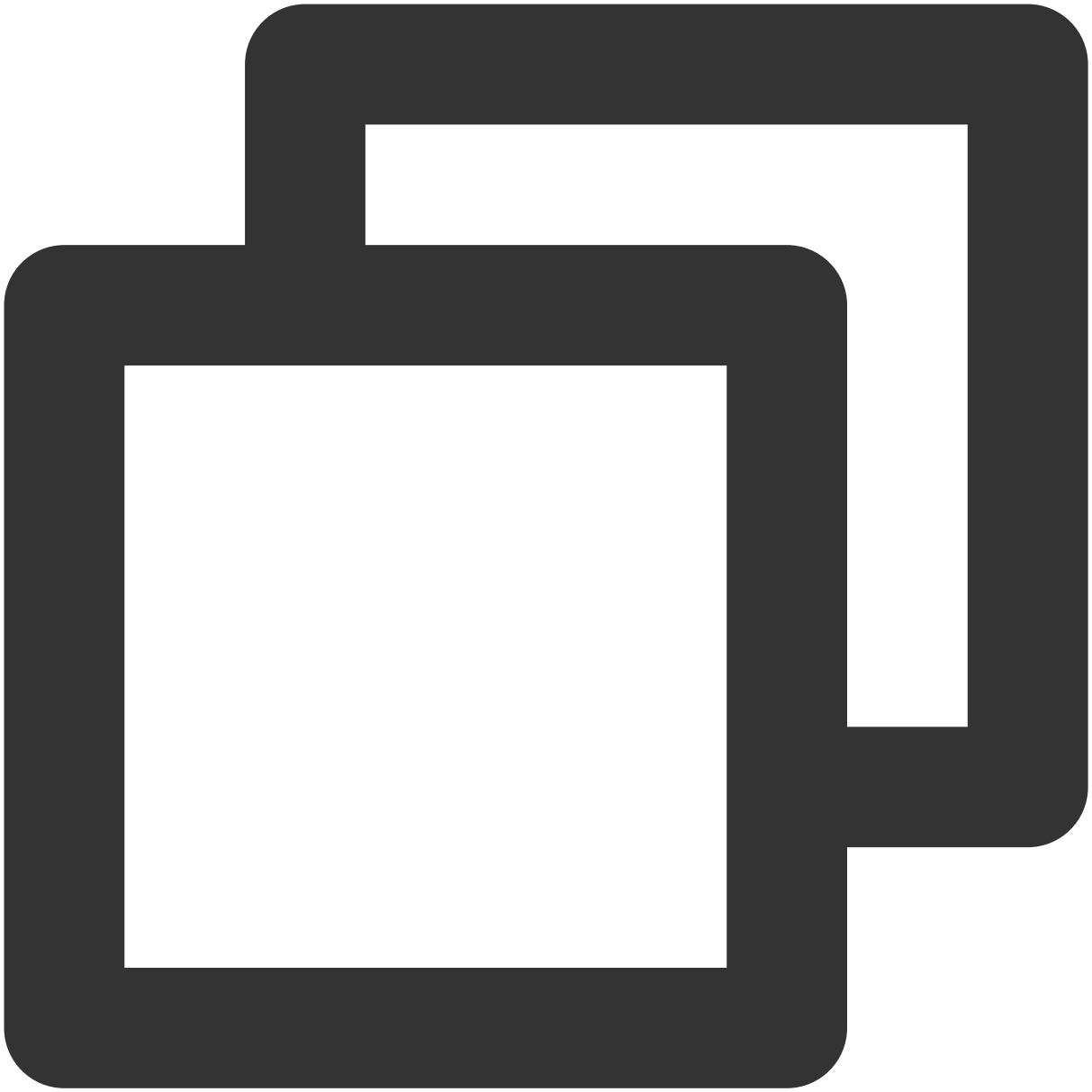
```
func (s *CIService) DocPreview(ctx context.Context, name string, opt *DocPreviewOpt
```

请求示例



```
opt := &cos.DocPreviewOptions{
    Page: 1,
}
resp, err := c.CI.DocPreview(context.Background(), "form.pdf", opt)
```

参数说明



```
type DocPreviewOptions struct {
    SrcType             string
    Page                int
    ImageParams         string
    Sheet               int
    DstType             string
    Password            string
    Comment             int
    ExcelPaperDirection int
    Quality             int
    Zoom                int
```

```
}
```

参数名称	描述	类型
name	对象文件名，例如 folder/document.pdf	String
SrcType	源数据的后缀类型，当前文档转换根据 COS 对象的后缀名来确定源数据类型。当 COS 对象没有后缀名时，可以设置该值	String
page	需转换的文档页码，默认从1开始计数； 表格文件中 page 表示转换的第 x 个 sheet 的第 x 张图	int
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String
Sheet	表格文件参数，转换第 x 个表，默认为1	int
DstType	转换输出目标文件类型 png 表示转换成 png 格式的图片文件 jpg 表示转换成 jpg 格式的图片文件 pdf 表示转换成 pdf 格式的图片文件。 如果传入的格式未能识别，默认使用 jpg 格式	String
Password	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String
Comment	是否隐藏批注和应用修订，默认为 0。 0：隐藏批注，应用修订 1：显示批注和修订	int
ExcelPaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	int
Quality	生成预览图的图片质量，取值范围：[1-100]，默认值100。 例如：值为100，代表生成图片质量为100%	int
Zoom	预览图片的缩放参数，取值范围：[10-200]， 默认值100。 例如：值为200，代表图片缩放比例为200%，即放大两倍	int

异步处理队列接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于文档预览队列接口的 API 概览以及 SDK 示例代码。

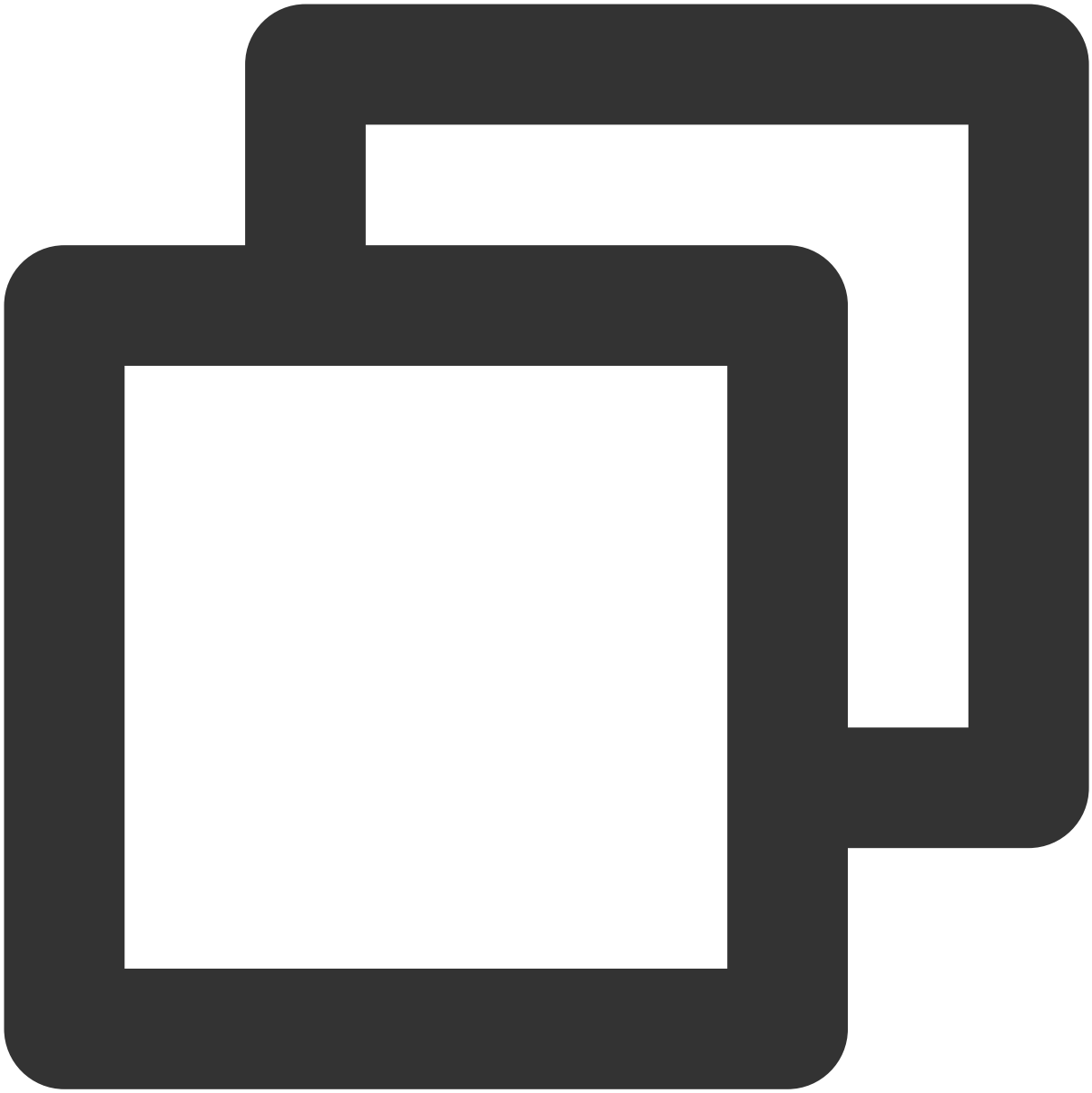
API	操作名	操作描述
DescribeDocProcessQueues	查询文档预览队列	查询文档预览队列信息
UpdateDocProcessQueue	更新文档预览队列	更新文档预览队列信息

查询文档预览队列

功能说明

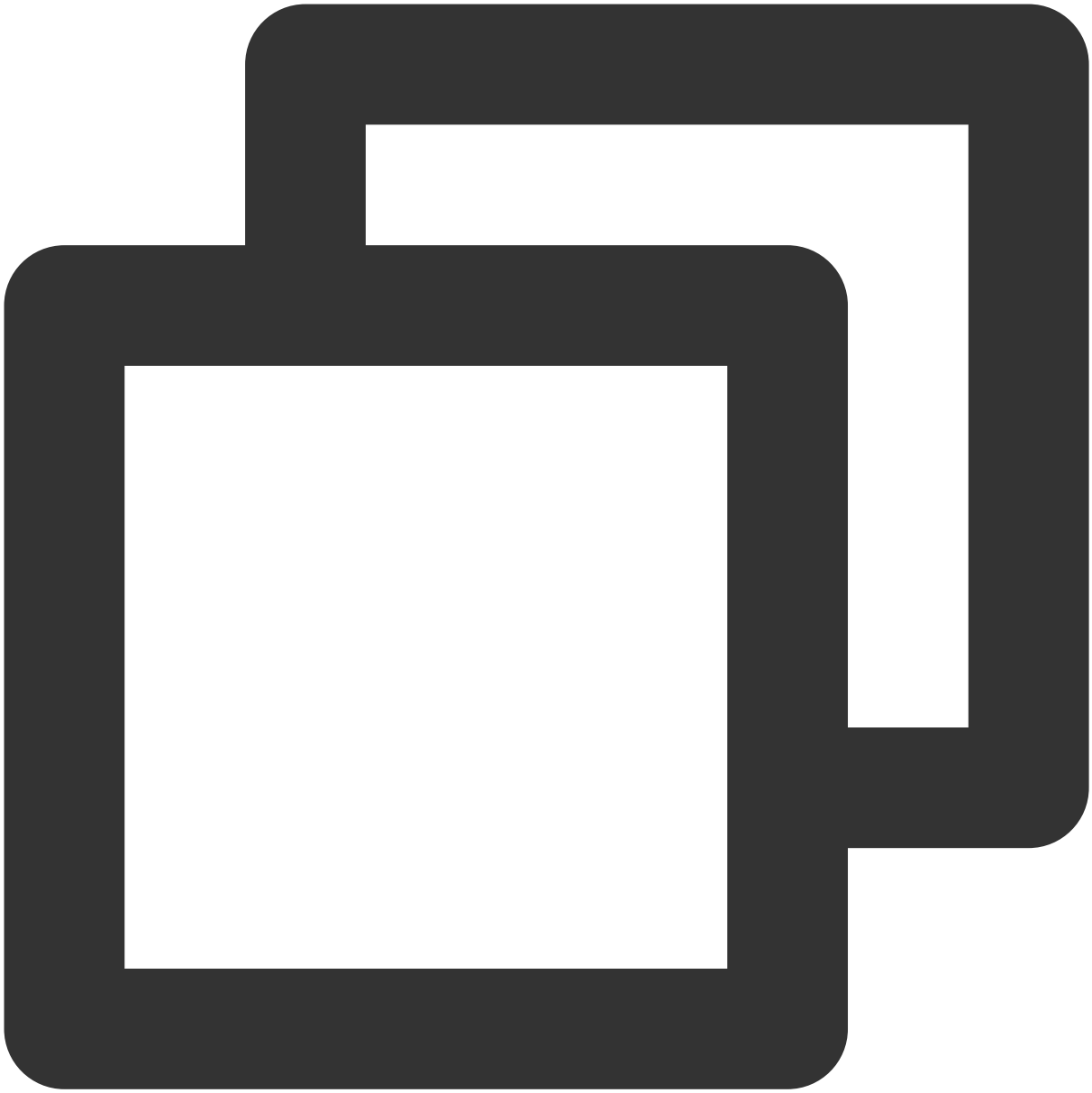
DescribeDocProcessQueues 接口用于查询文档预览队列。

方法原型



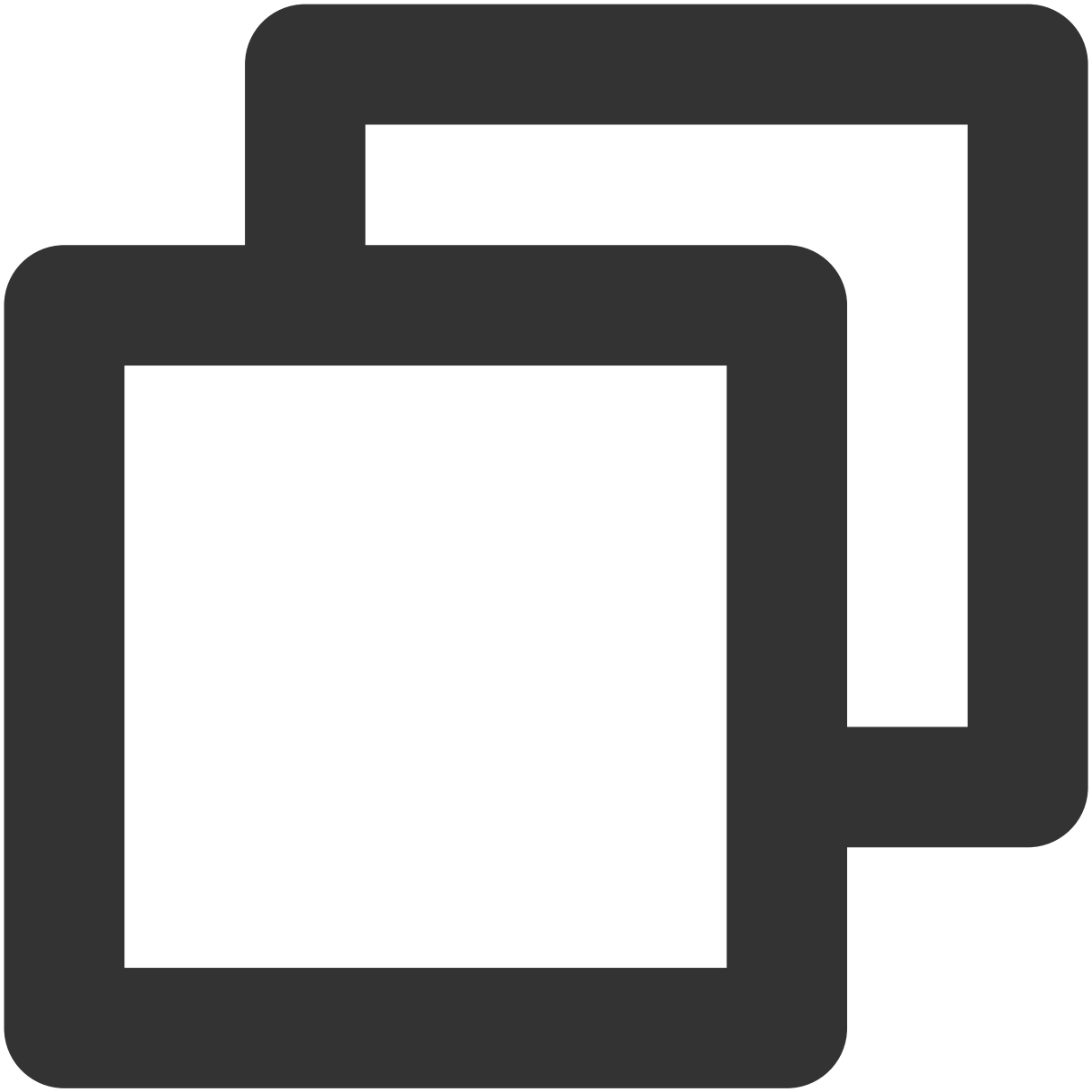
```
func (s *CIService) DescribeDocProcessQueues(ctx context.Context, opt *DescribeDocP
```

请求示例



```
DescribeQueueOpt := &cos.DescribeDocProcessQueuesOptions{
    QueueIds: "p111a8dd208104ce3b11c78398f658ca8,p4318f85d2aa14c43b1dba6f9b78be9b3,
    PageNumber: 1,
    PageSize:   2,
}
res, _, err := c.CI.DescribeDocProcessQueues(context.Background(), DescribeQueueOpt
```

参数说明

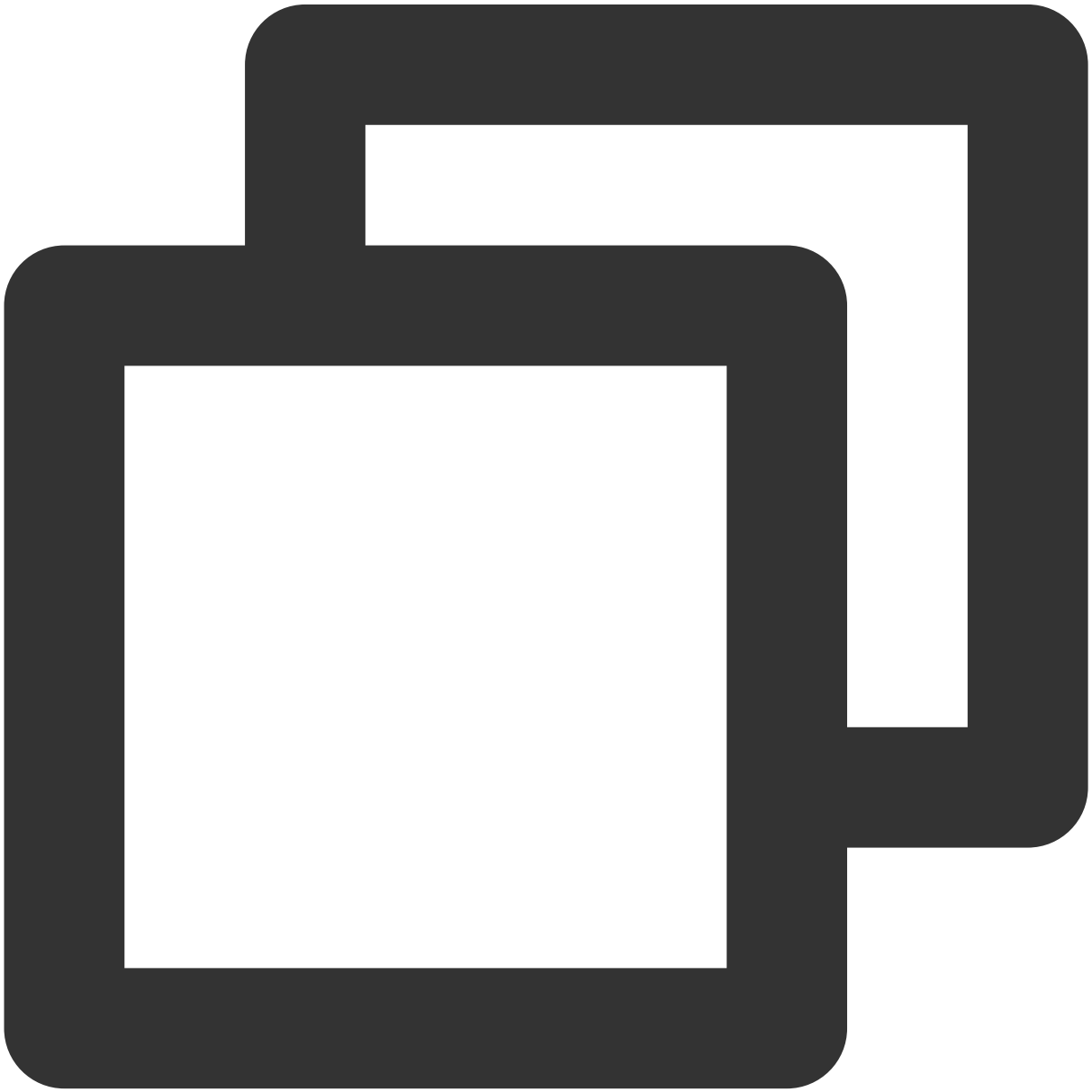


```
type DescribeDocProcessQueuesOptions struct {
    QueueIds   string `url:"queueIds,omitempty"`
    State      string `url:"state,omitempty"`
    PageNumber int    `url:"pageNumber,omitempty"`
    PageSize   int    `url:"pageSize,omitempty"`
}
```

参数名称	描述	类型
QueueIds	队列 ID，以“,”符号分割字符串	String

State	1. Active 表示队列内的作业会被文档预览服务调度执行 2. Paused 表示队列暂停，作业不再会被文档预览服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于执行中的任务将继续执行，不受影响	String
PageNumber	第几页	int
PageSize	每页个数	int

返回结果说明



```
type DescribeDocProcessQueuesResult struct {
```

```
    XMLName      xml.Name
    RequestId    string
    TotalCount   int
    PageNumber   int
    PageSize     int
    QueueList    []DocProcessQueue
    NonExistPIDs []string
}
type DocProcessQueue struct {
    QueueId       string
    Name          string
    State         string
    MaxSize       int
    MaxConcurrent int
    UpdateTime    string
    CreateTime    string
    NotifyConfig  *DocProcessQueueNotifyConfig
}
type DocProcessQueueNotifyConfig struct {
    Url   string
    State string
    Type  string
    Event string
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	String
TotalCount	队列总数	Int
PageNumber	当前页数，同请求中的 pageNumber	Int
PageSize	每页个数，同请求中的 pageSize	Int
QueueList	队列数组	Container
NonExistPIDs	不存在的队列 ID 列表	String 数组
QueueId	队列 ID	String
Name	队列名字	String
State	当前状态，Active 或者 Paused	String
NotifyConfig	回调配置	Container
MaxSize	队列最大长度	Int

版权所有：腾讯云计算（北京）有限责任公司

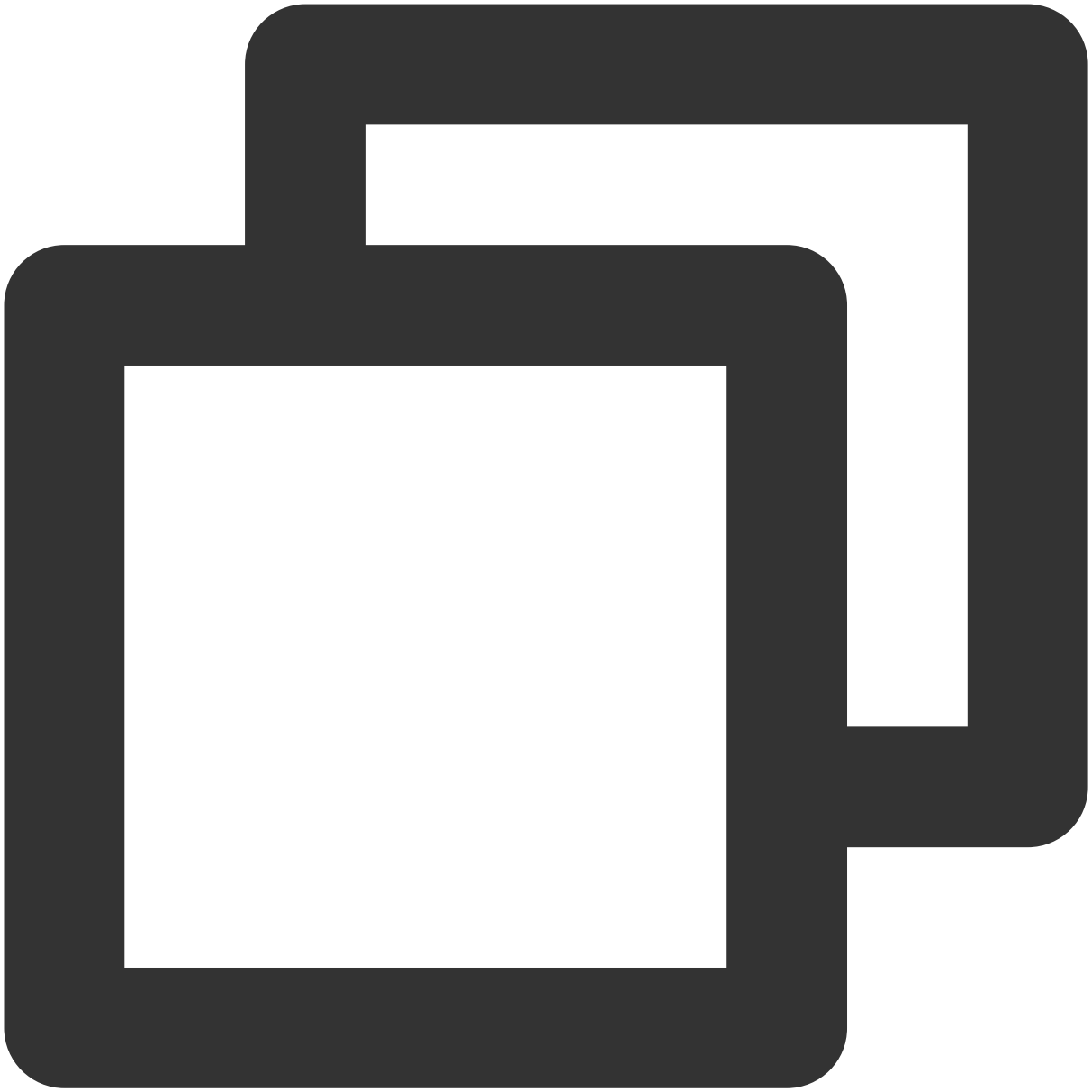
MaxConcurrent	当前队列最大并行执行的任务数	Int
UpdateTime	更新时间	String
CreateTime	创建时间	String
Url	回调地址	String
State	开关状态，On 或者 Off	String
Type	回调类型，Url	String
Event	触发回调的事件	String

更新文档预览队列

功能说明

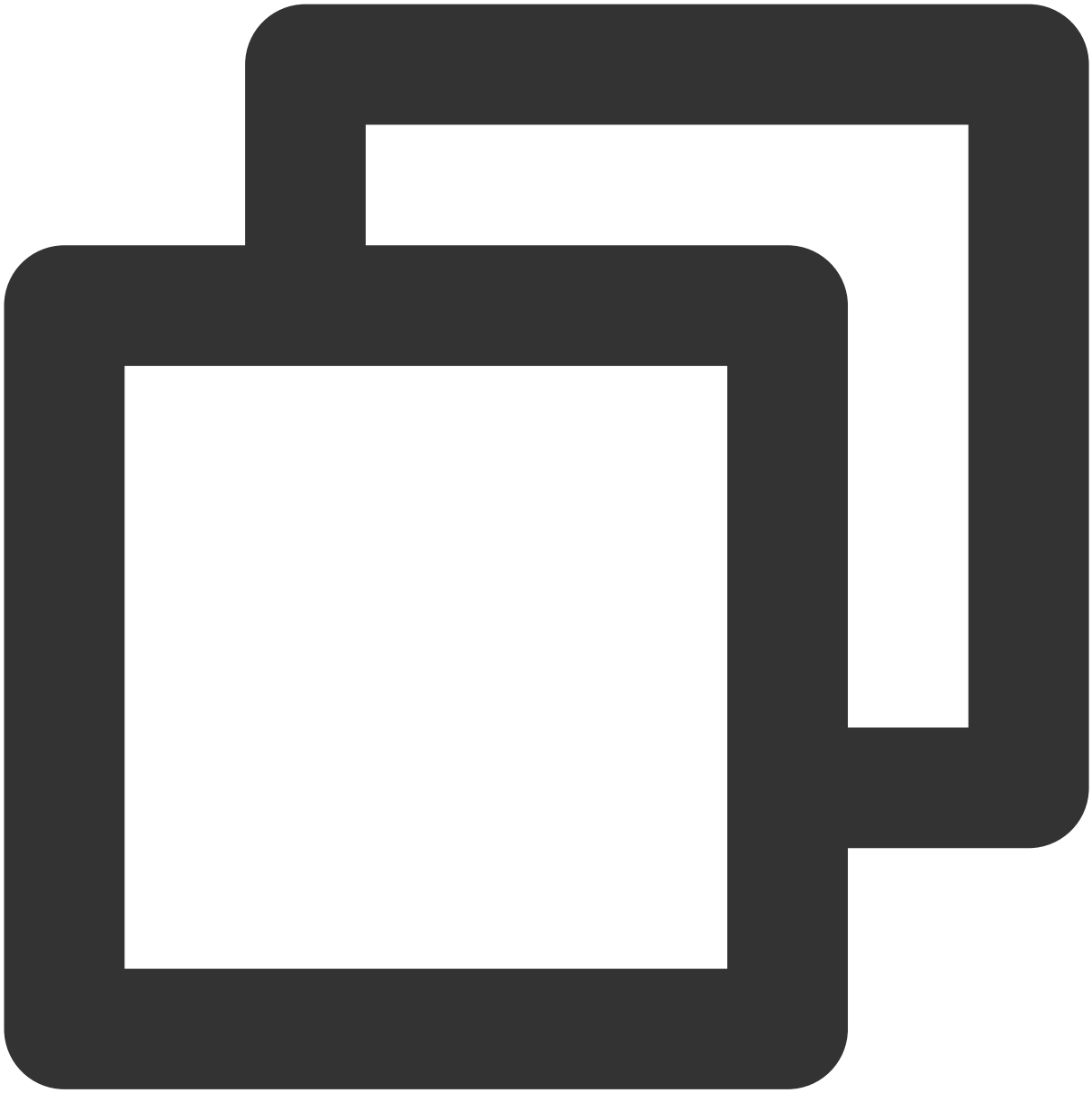
UpdateDocProcessQueue 接口用于更新文档预览队列。

方法原型



```
func (s *CIService) UpdateDocProcessQueue(ctx context.Context, opt *UpdateDocProces
```

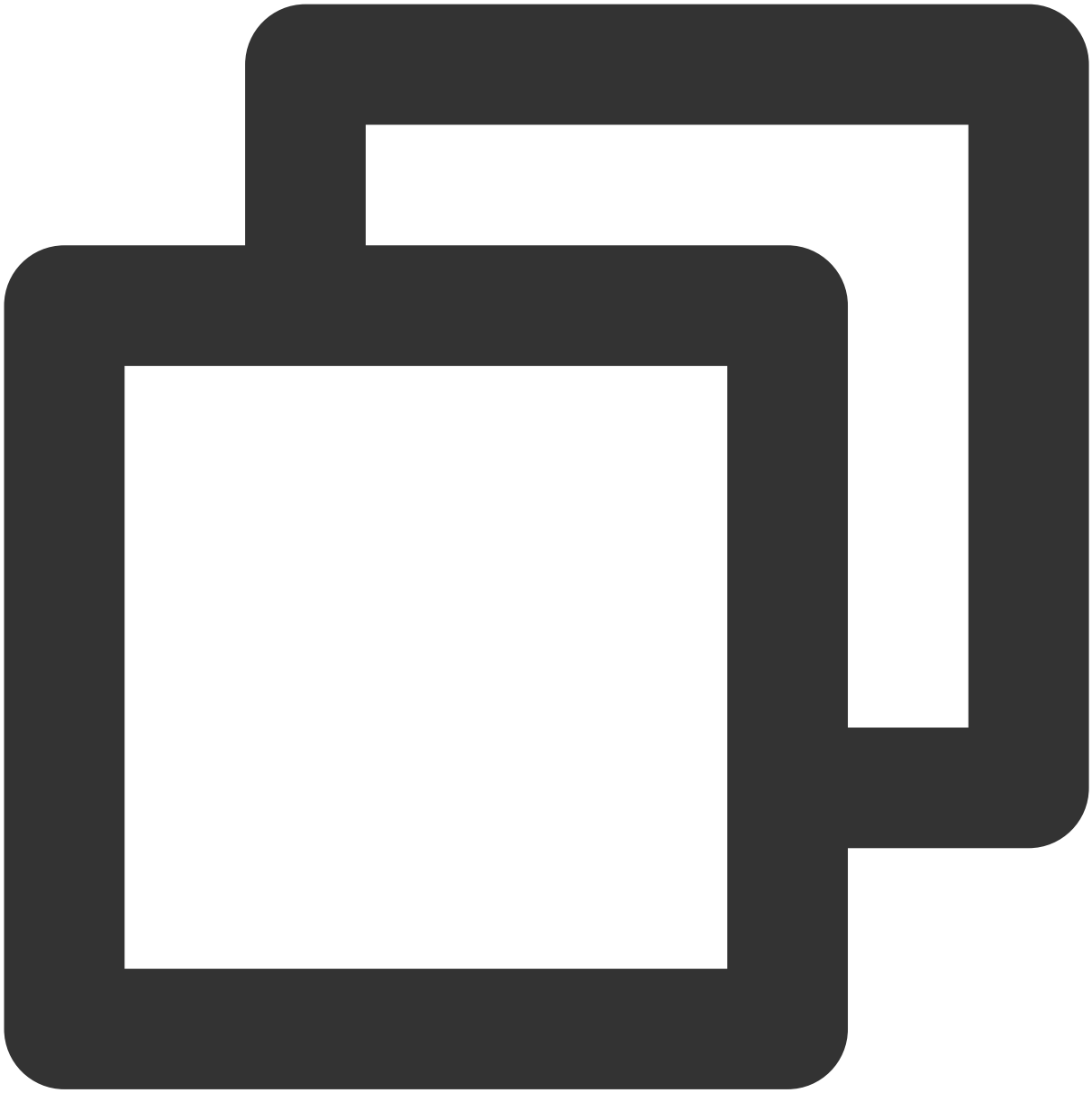
请求示例



```
updateQueueOpt := &cos.UpdateDocProcessQueueOptions{
    Name:     "queue-doc-process-1",
    QueueID: "p111a8dd208104ce3b11c78398f658ca8",
    State:    "Active",
    NotifyConfig: &cos.DocProcessQueueNotifyConfig{
        State: "Off",
    },
}
res, _, err := c.CI.UpdateDocProcessQueue(context.Background(), updateQueueOpt)
```

版权所有：腾讯云计算（北京）有限责任公司

参数说明

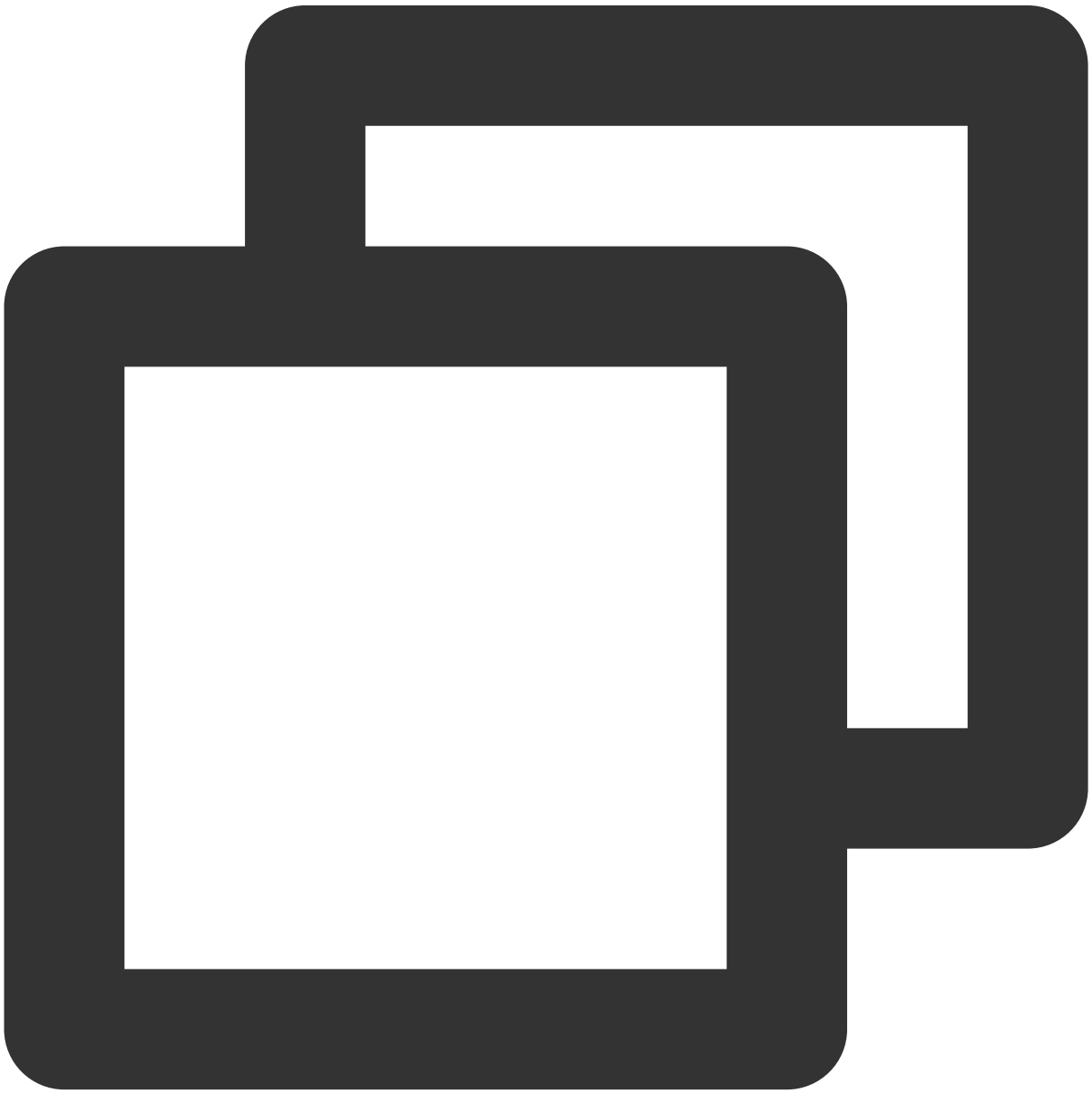


```
type UpdateDocProcessQueueOptions struct {
    XMLName      xml.Name
    Name         string
    QueueID      string
    State        string
    NotifyConfig *DocProcessQueueNotifyConfig
}
```

参数名称	描述	类型

Name	队列名称	String
QueueID	队列 ID	String
State	队列状态	String
NotifyConfig	通知渠道，详情请参见 DocProcessQueue.NotifyConfig	Container

返回结果说明



```
type UpdateDocProcessQueueResult struct {
    XMLName   xml.Name
    RequestId string
    Queue     *DocProcessQueue
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	Struct
Queue	队列信息，详情请参见 DescribeDocProcessQueuesResult.QueueList	Struct

异步处理任务接口

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于文档预览任务接口的 API 概览以及 SDK 示例代码。

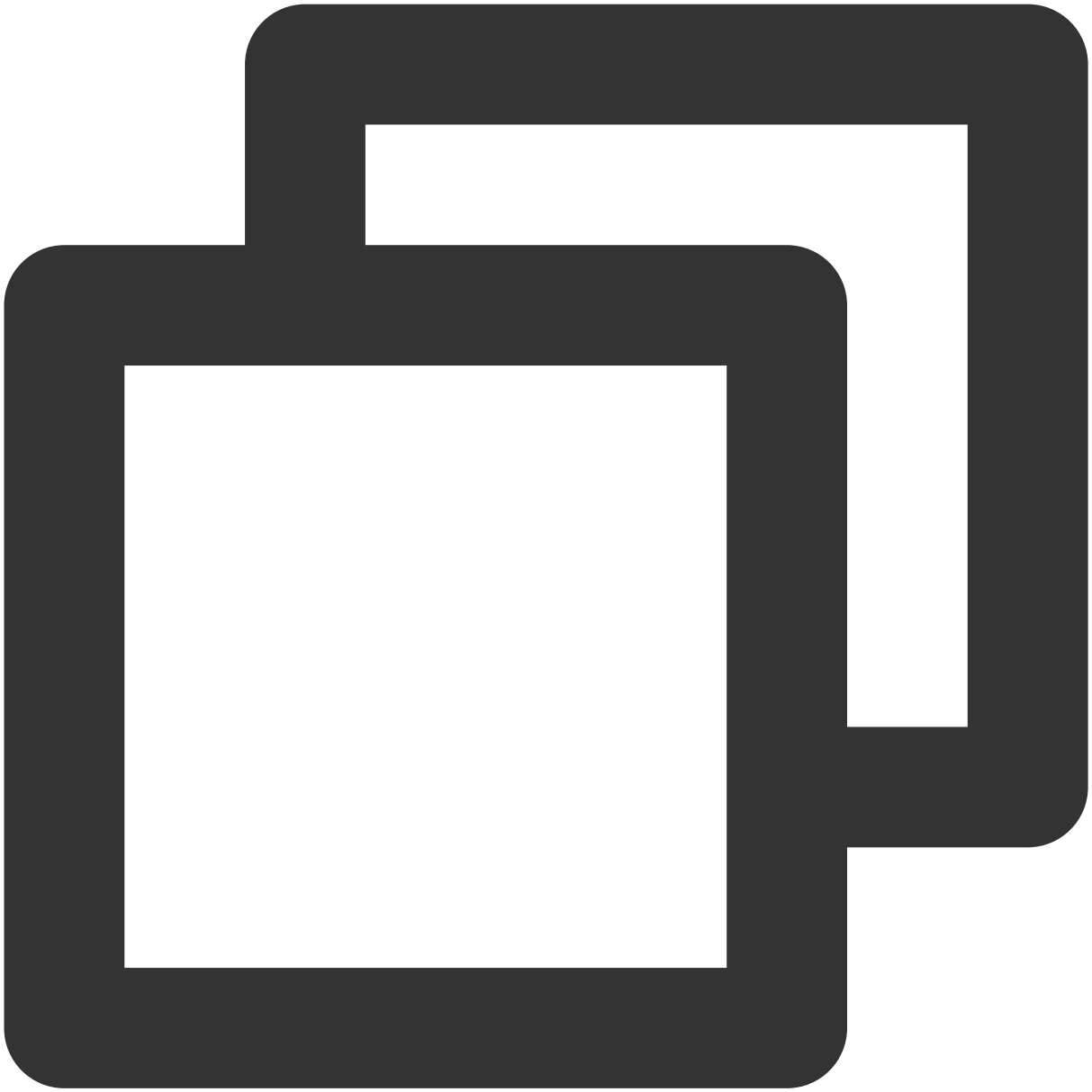
API	操作名	操作描述
CreateDocProcessJobs	提交文档预览任务	用于提交一个文档预览任务
DescribeDocProcessJob	查询指定的文档预览任务	用于查询指定的文档预览任务
DescribeDocProcessJobs	拉取符合条件的文档预览任务	用于拉取符合条件的文档预览任务

提交文档预览任务

功能说明

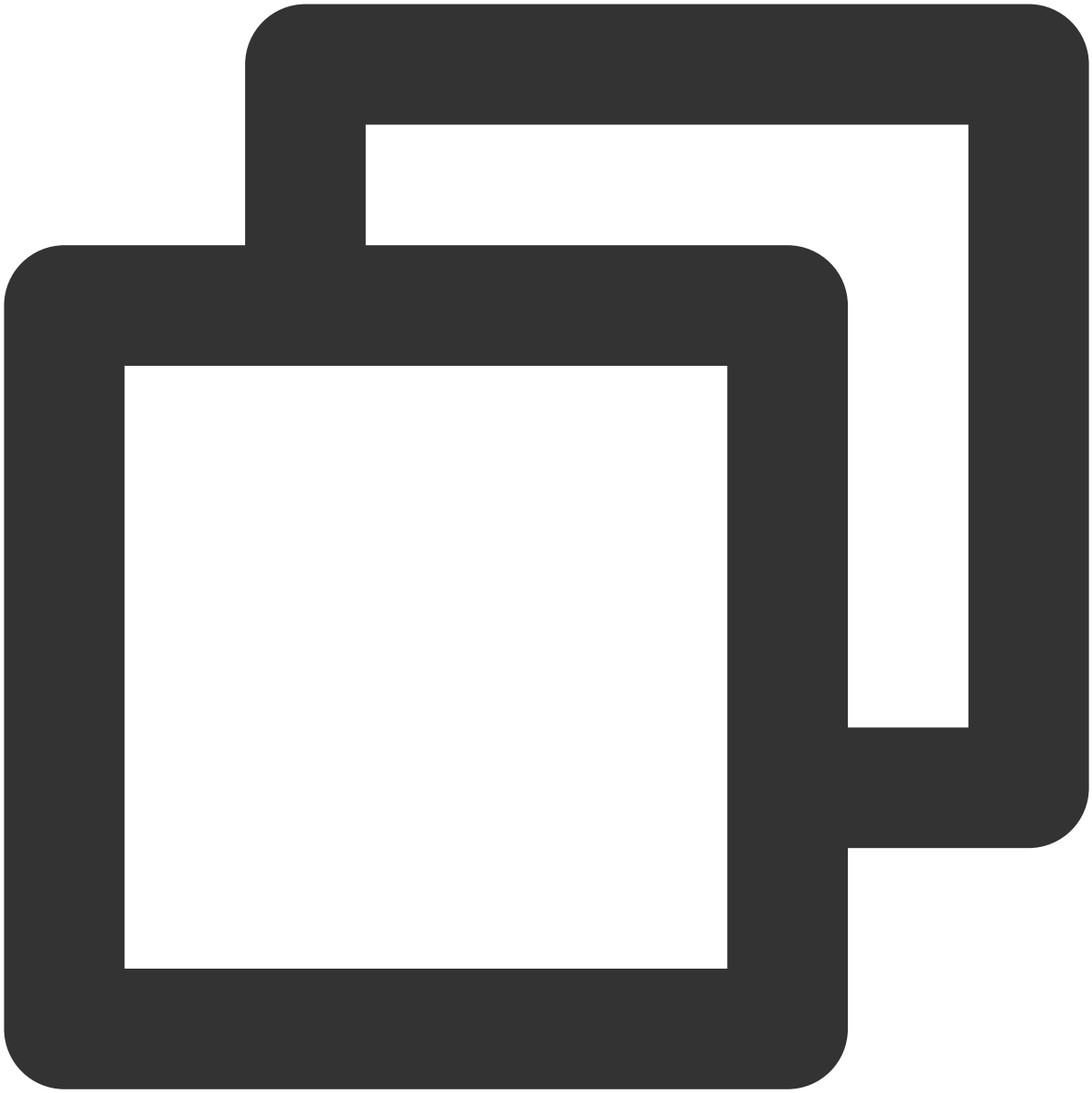
CreateDocProcessJobs 接口用于提交一个文档预览任务。

方法原型



```
func (s *CIService) CreateDocProcessJobs(ctx context.Context, opt *CreateDocProcess
```

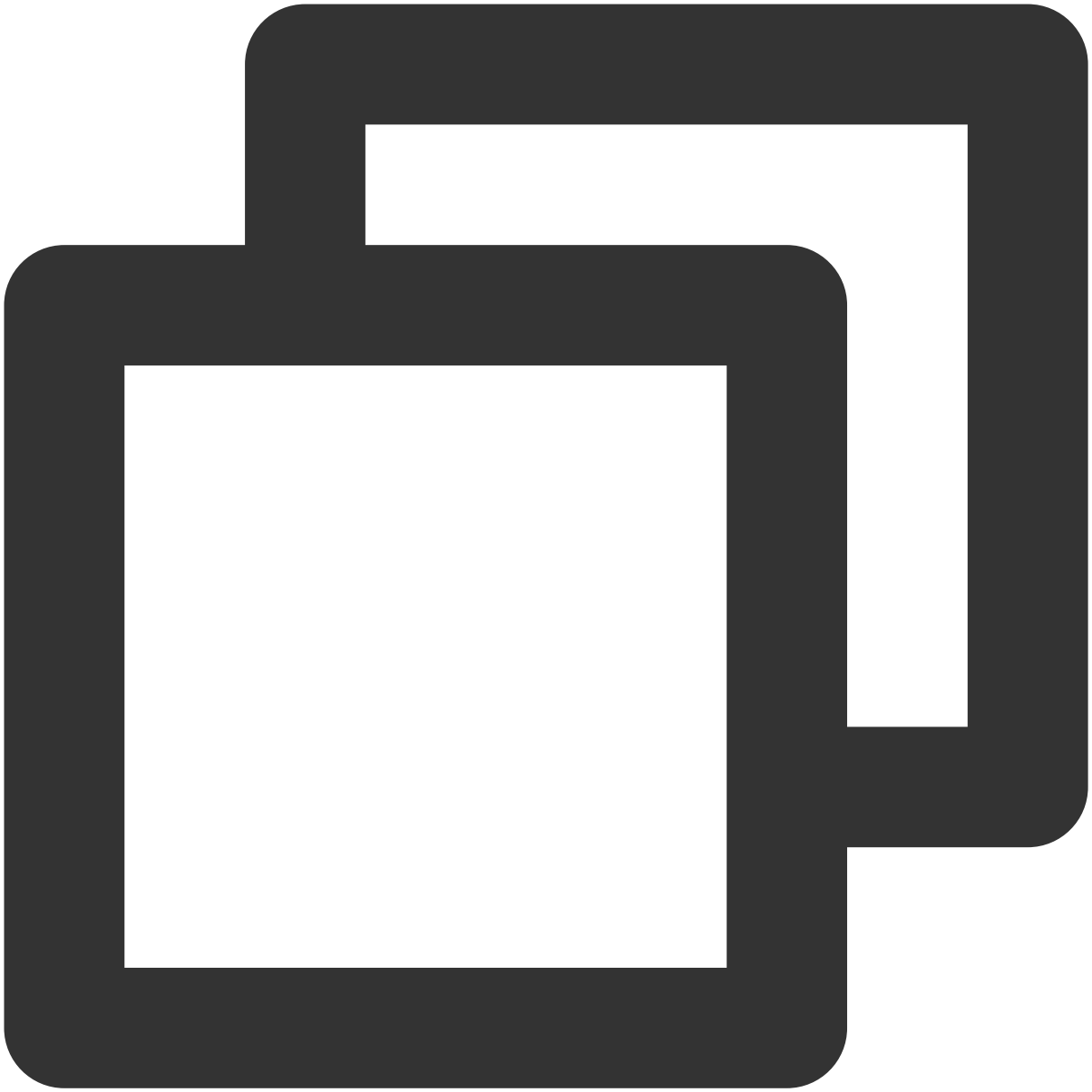
请求示例



```
createJobOpt := &cos.CreateDocProcessJobsOptions{
    Tag: "DocProcess",
    Input: &cos.DocProcessJobInput{
        Object: "form.pdf",
    },
    Operation: &cos.DocProcessJobOperation{
        Output: &cos.DocProcessJobOutput{
            Region: "ap-guangzhou",
            Object: "test-doc${Number}",
            Bucket: "examplebucket-1250000000",
        },
```

```
        DocProcess: &cos.DocProcessJobDocProcess{
            TgtType:     "png",
            StartPage:   1,
            EndPage:     -1,
            ImageParams: "watermark/1/image/aHR0cDovL3Rlc3QwMDUtMTI1MTcwNDcwOC5jb3M
        },
    },
    QueueId: "p111a8dd208104ce3b11c78398f658ca8",
}
res, _, err := c.CI.CreateDocProcessJobs(context.Background(), createJobOpt)
```

参数说明



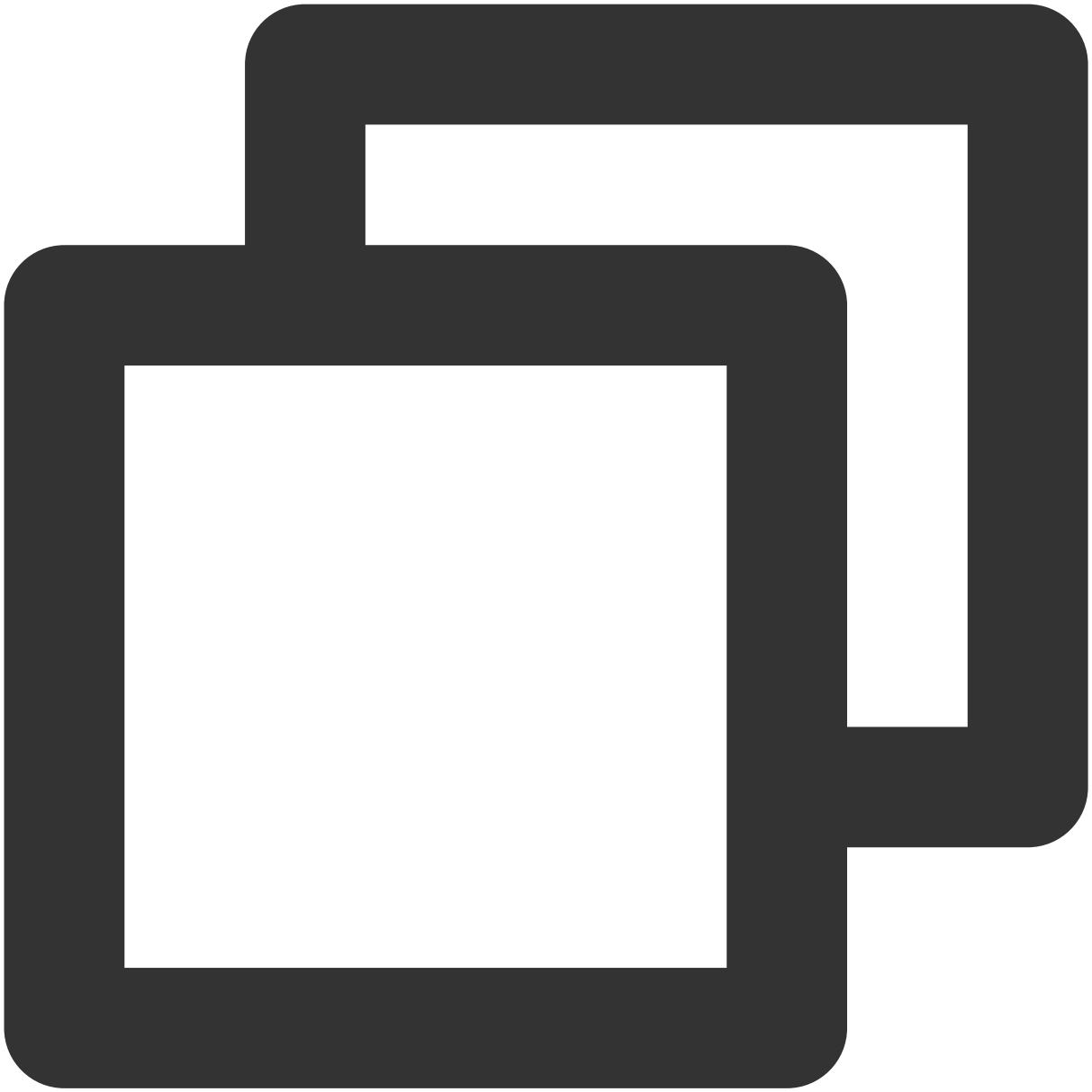
```
type CreateDocProcessJobsOptions struct {
    XMLName   xml.Name
    Tag       string
    Input     *DocProcessJobInput
    Operation *DocProcessJobOperation
    QueueId   string
}
type DocProcessJobInput struct {
    Object string
}
type DocProcessJobOperation struct {
```

```
    Output           *DocProcessJobOutput
    DocProcess       *DocProcessJobDocProcess
    DocProcessResult *DocProcessJobDocProcessResult
}
type DocProcessJobOutput struct {
    Region string
    Bucket string
    Object string
}
type DocProcessJobDocProcess struct {
    SrcType        string
    TgtType        string
    SheetId        int
    StartPage      int
    EndPage        int
    ImageParams    string
    DocPassword    string
    Comments       int
    PaperDirection int
    Quality        int
    Zoom           int
}
```

参数名称	描述	类型
Tag	创建任务的 Tag，目前仅支持：DocProcess	String
Input	待操作的文件对象	Container
Operation	操作规则	Container
QueueId	任务所在的队列 ID	String
Object	文件在 COS 上的文件路径，Bucket 由 Host 指定	String
Output	结果输出地址	Container
DocProcess	当 Tag 为 DocProcess 时有效，指定该任务的参数	Container
Region	存储桶的地域	String
Bucket	存储结果的存储桶	String
Object	输出文件路径。**非表格文件输出文件名需包含${Number}或${Page}参数。**多个输出文件，${Number} 表示序号从1开始，${Page}表示序号与预览页码一致。${Number} 表示多个输出文件，序号从1开始，例如输入 abc_${Number}.jpg，预览某文件5-6页，则输出文件名为 abc_1.jpg，abc_2.jpg${Page}表示多个输出文件，序号与预览页码一致，例如输入	String

	abc_${Page}.jpg，预览某文件5-6页，则输出文件名为 abc_5.jpg，abc_6.jpg。**表格文件输出路径需包含${SheetID}占位符，输出文件名必须包含${Number}参数。**例如 /${SheetID}/ abc_${Number}.jpg，先根据 excel 转换的表格数，生成对应数量的文件夹，再在对应的文件夹下，生成对应数量的图片文件	
SrcType	源数据的后缀类型。当前文档转换根据 COS 对象的后缀名来确定源数据类型，当 COS 对象没有后缀名时，可以设置该值	String
TgtType	转换输出目标文件类型，png 表示转换成 png 格式的图片文件；jpg 表示转换成 jpg 格式的图片文件；pdf 表示转换成 pdf 格式的图片文件。如果传入的格式未能识别，默认使用 jpg 格式	String
SheetId	表格文件参数，转换第 x 个表，默认为1；设置 SheetId 为0，即转换文档中全部表格	Int
StartPage	从第 x 页开始转换；在表格文件中，一张表可能分割为多页转换，生成多张图片。StartPage 表示从指定 SheetId 的第 x 页开始转换。默认为1	Int
EndPage	转换至第 x 页；在表格文件中，一张表可能分割为多页转换，生成多张图片。EndPage 表示转换至指定 SheetId 的第 x 页。默认为-1，即转换全部页	Int
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String
DocPassword	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String
Comments	是否隐藏批注和应用修订，默认为 0。 0：隐藏批注，应用修订；1：显示批注和修订	Int
PaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	Int
Quality	生成预览图的图片质量，取值范围：[1-100]，默认值100。 例如：值为100，代表生成图片质量为100%	Int
Zoom	预览图片的缩放参数，取值范围：[10-200]，默认值100。 例如：值为200，代表图片缩放比例为200% 即放大两倍	Int

返回结果说明



```
type CreateDocProcessJobsResult struct {
    JobsDetail DocProcessJobDetail
}
type DocProcessJobDetail struct {
    Code         string
    Message      string
    JobId        string
    Tag          string
    State        string
    CreationTime string
    QueueId      string
```

```
    Input         *DocProcessJobInput
    Operation     *DocProcessJobOperation
}
```

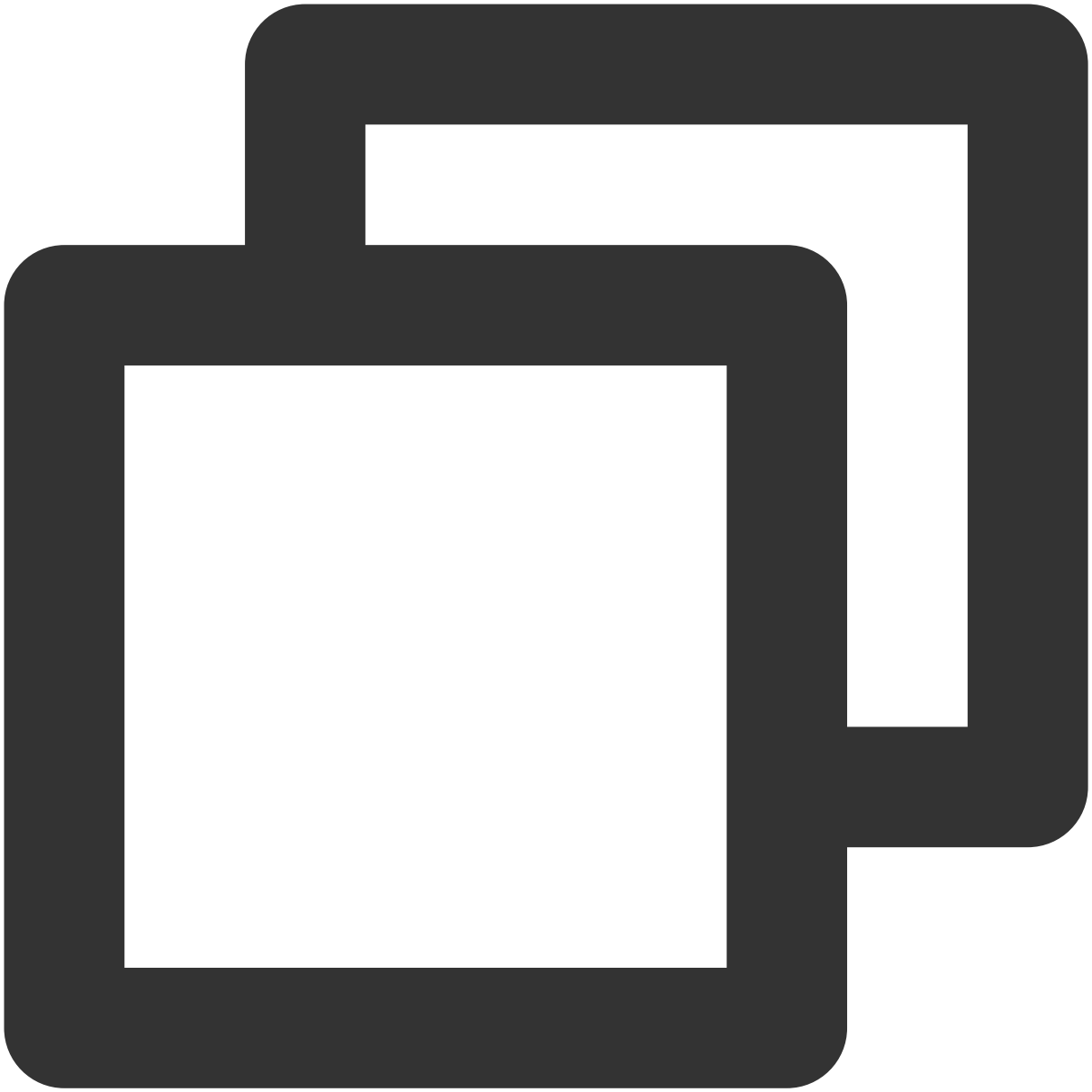
参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：DocProcess	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String
CreationTime	任务的创建时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径，详情请参见CreateDocProcessJobsOptions.Input	Container
Operation	该任务的规则，详情请参见CreateDocProcessJobsOptions.Operation	Container

查询指定文档预览任务

功能说明

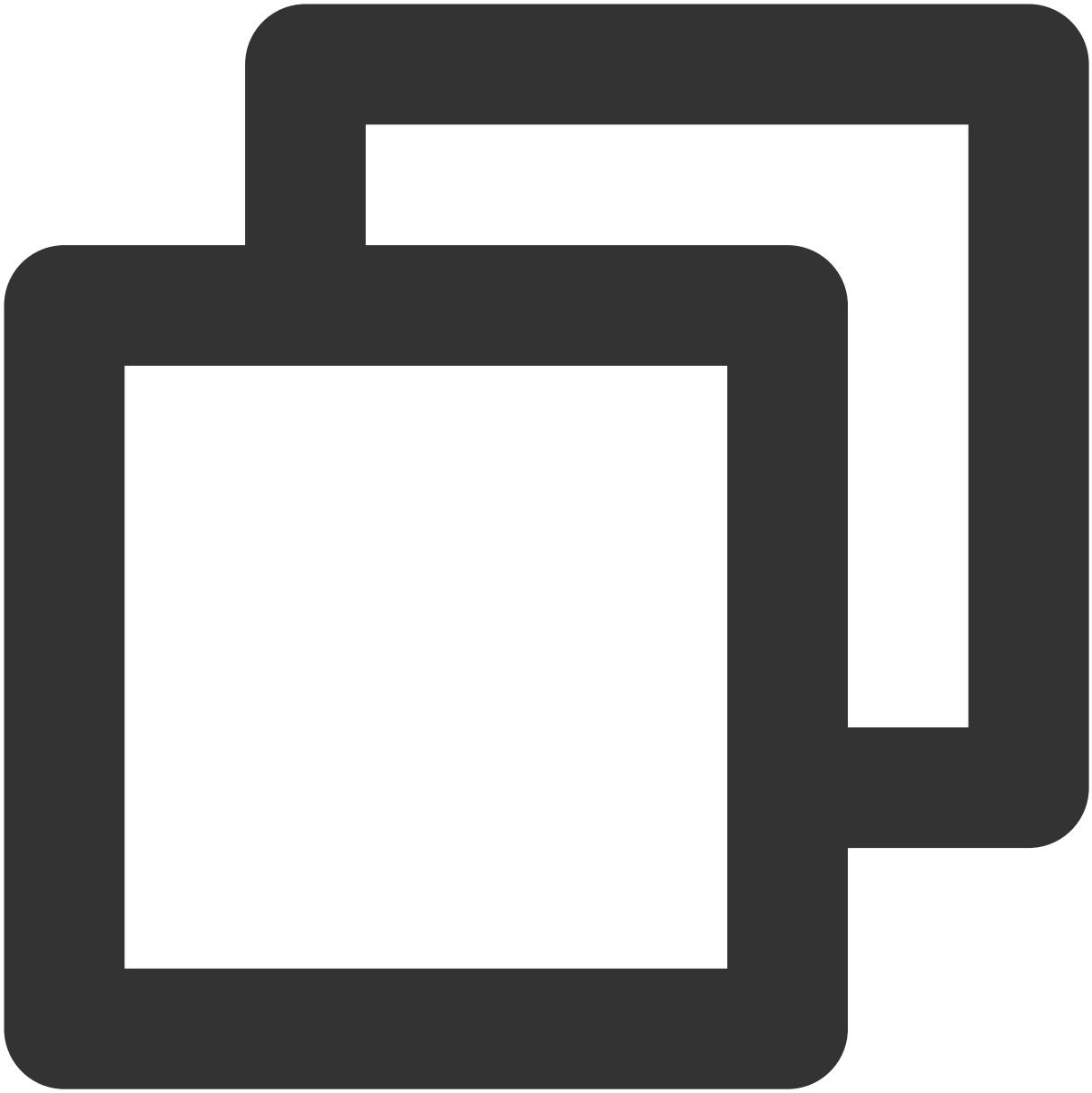
DescribeDocProcessJob 用于查询指定的文档预览任务。

方法原型



```
func (s *CIService) DescribeDocProcessJob(ctx context.Context, jobid string) (*Desc
```

请求示例

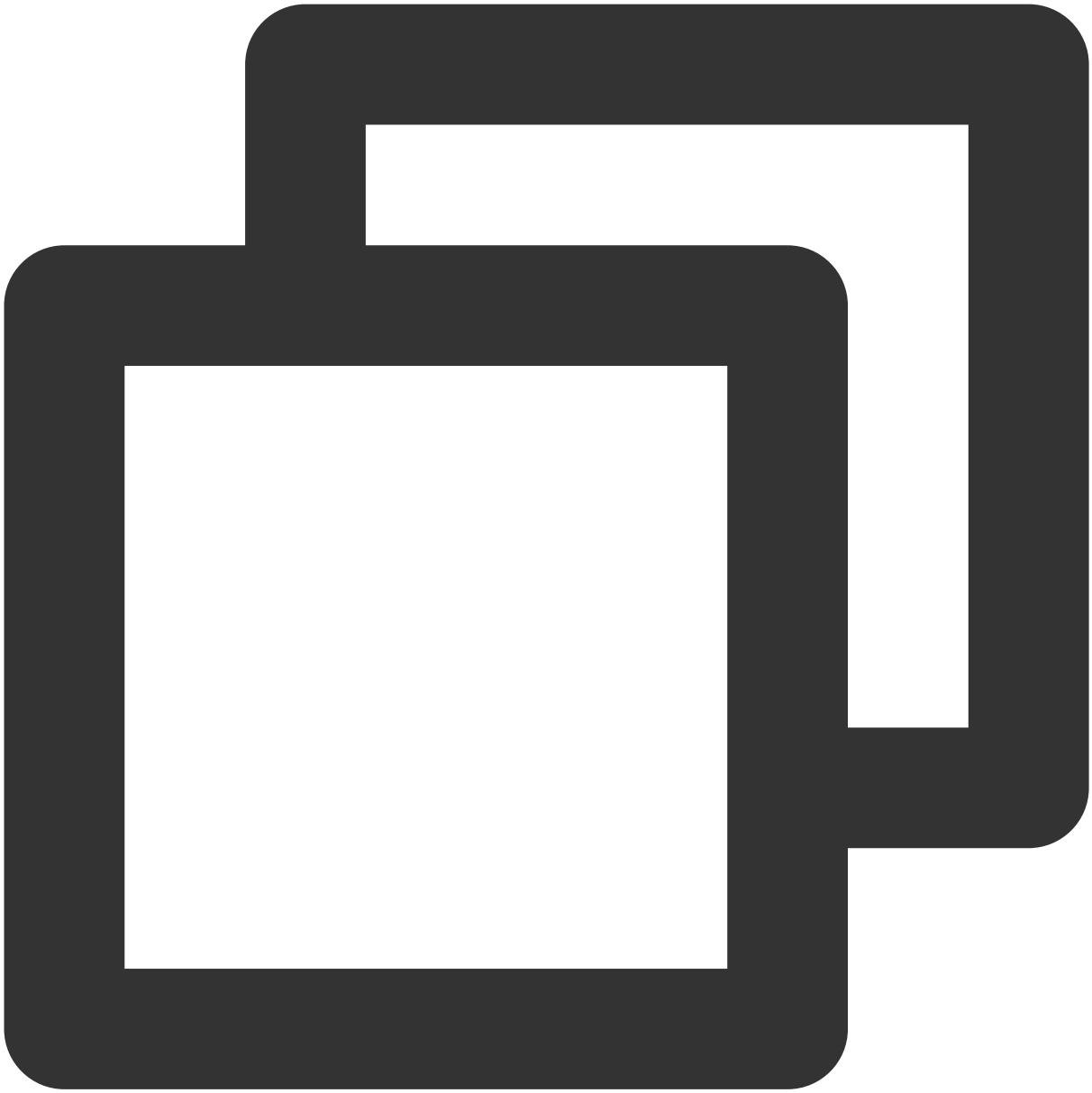


```
jobid := "<jobid>"
res, _, err := c.CI.DescribeDocProcessJob(context.Background(), jobid)
```

参数说明

参数名称	描述	类型
jobid	文档预览任务 ID	String

返回结果说明



```
type DescribeDocProcessJobResult struct {
    JobsDetail     *DocProcessJobDetail
    NonExistJobIds string
}
```

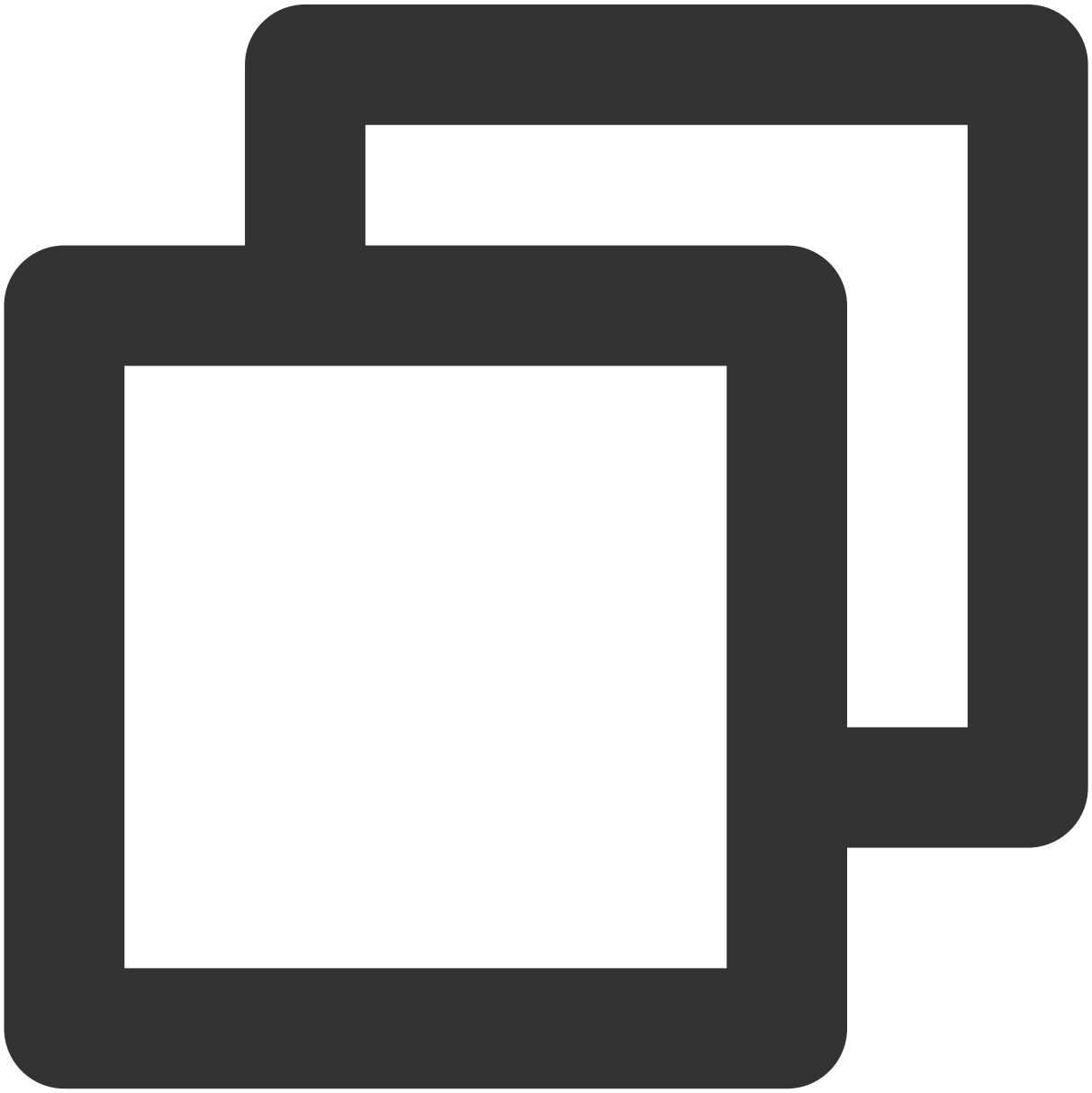
参数名称	描述	类型
JobsDetail	任务的详细信息， 同 CreateDocProcessJobs 接口的 Response.JobsDetail 节点	Container
NonExistJobIds	查询的 ID 中不存在的任务，所有任务都存在时不返回	String

拉取符合条件的文档预览任务

功能说明

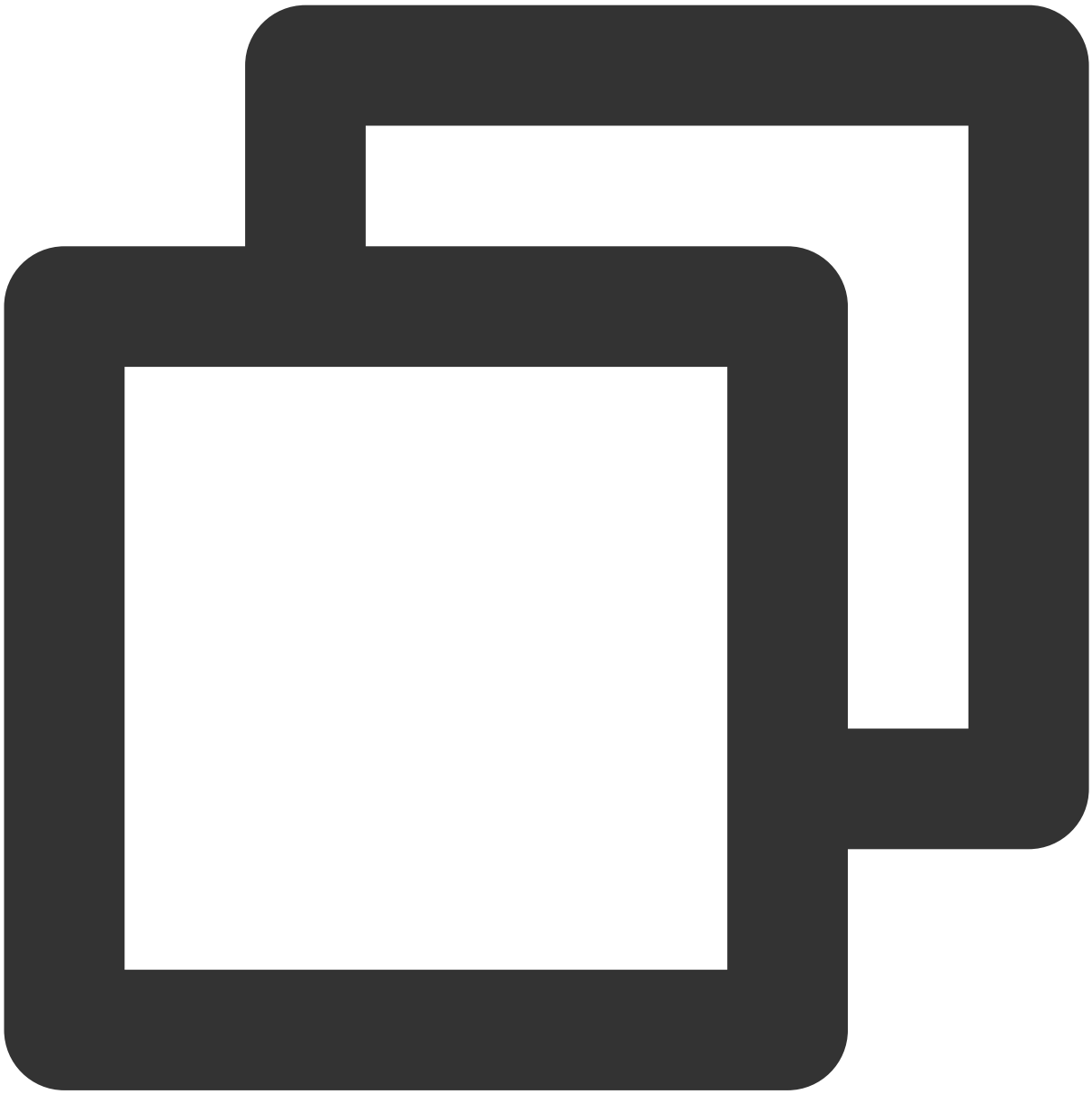
DescribeDocProcessJobs 用于拉取符合条件的文档预览任务。

方法原型



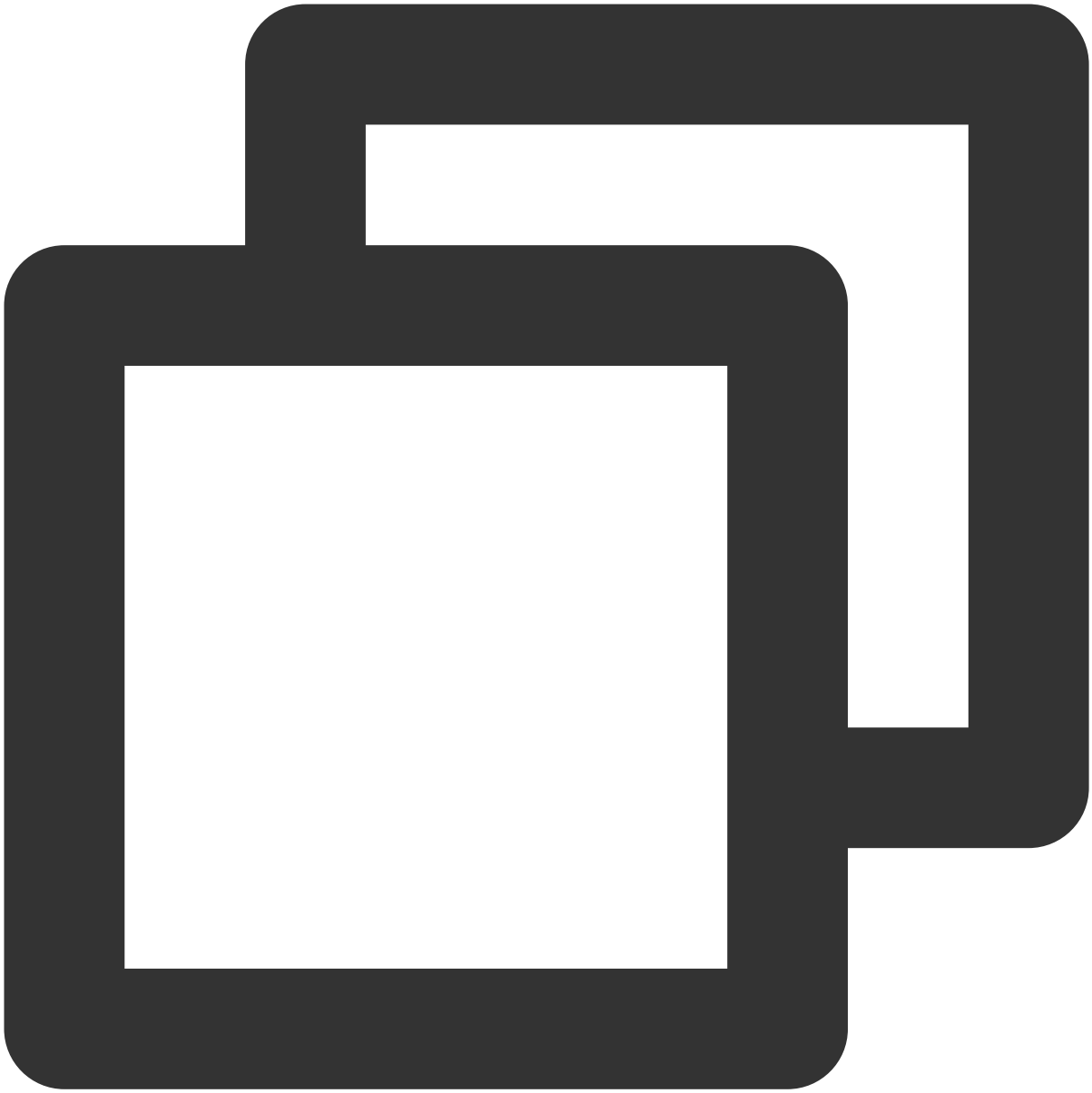
```
func (s *CIService) DescribeDocProcessJobs(ctx context.Context, opt *DescribeDocPro
```

请求示例



```
DescribeJobsOpt := &cos.DescribeDocProcessJobsOptions{
    QueueId: "p111a8dd208104ce3b11c78398f658ca8",
    Tag:     "DocProcess",
}
res, _, err := c.CI.DescribeDocProcessJobs(context.Background(), DescribeJobsOpt)
```

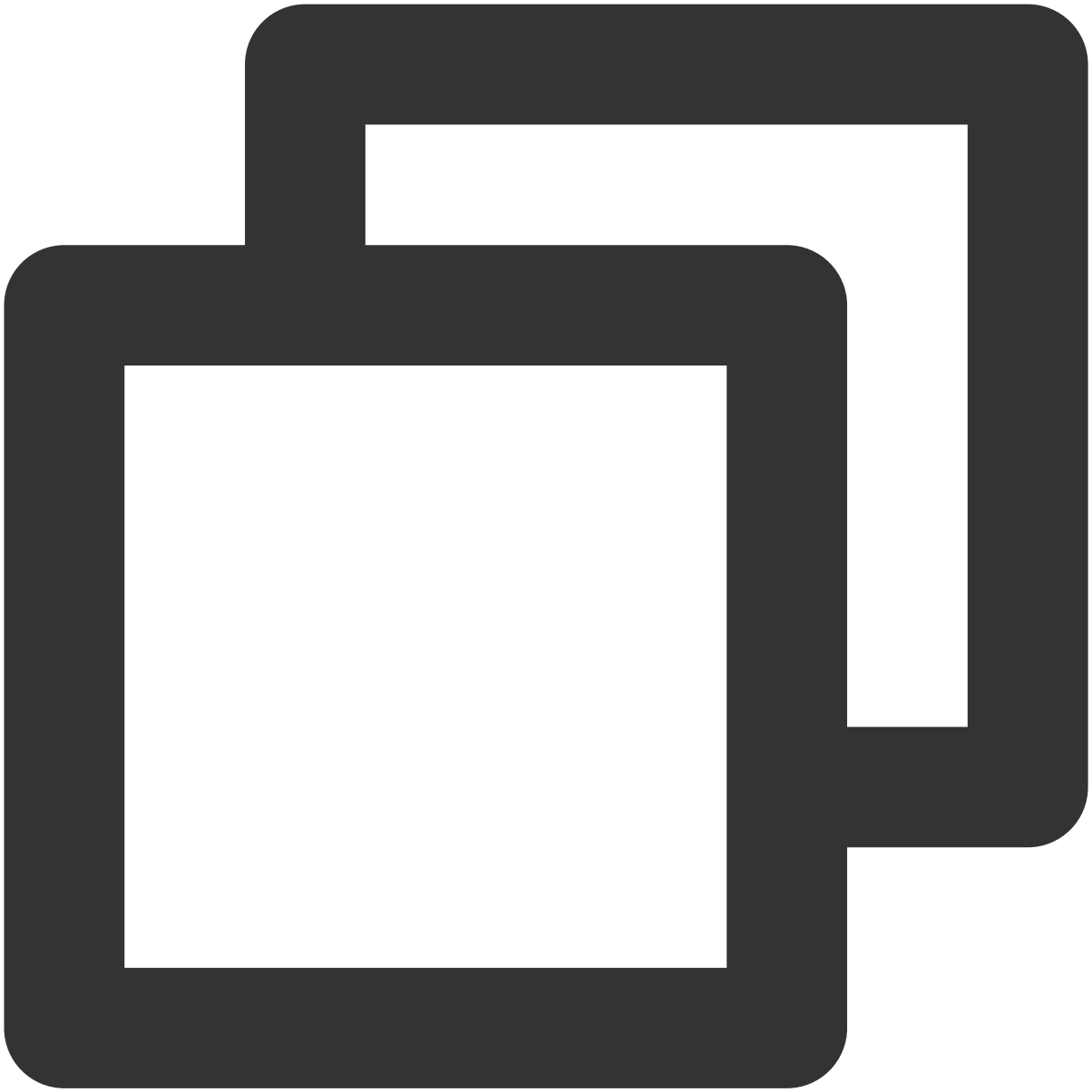
参数说明



```
type DescribeDocProcessJobsOptions struct {
    QueueId           string
    Tag               string
    OrderByTime       string
    NextToken         string
    Size              int
    States            string
    StartCreationTime string
    EndCreationTime   string
}
```

参数名称	描述	类型
QueueId	拉取该队列 ID 下的任务	String
Tag	任务的 Tag：DocProcess	String
OrderByTime	Desc 或者 Asc。默认为 Desc	String
NextToken	请求的上下文，用于翻页。上次返回的值	String
Size	拉取的最大任务数。默认为10，最大值为100	Int
States	拉取该状态的任务，以 `,` 分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel。默认为 All	String
StartCreationTime	拉取创建时间大于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String
EndCreationTime	拉取创建时间小于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String

返回结果说明



```
type DescribeDocProcessJobsResult struct {
    JobsDetail []DocProcessJobDetail
    NextToken  string
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息，同 CreateDocProcessJobs 接口中的 Response.JobsDetail 节点	Container
NextToken	翻页的上下文 Token	String

文档转 HTML

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于文档预览同步请求的 API 概览以及 SDK 示例代码。

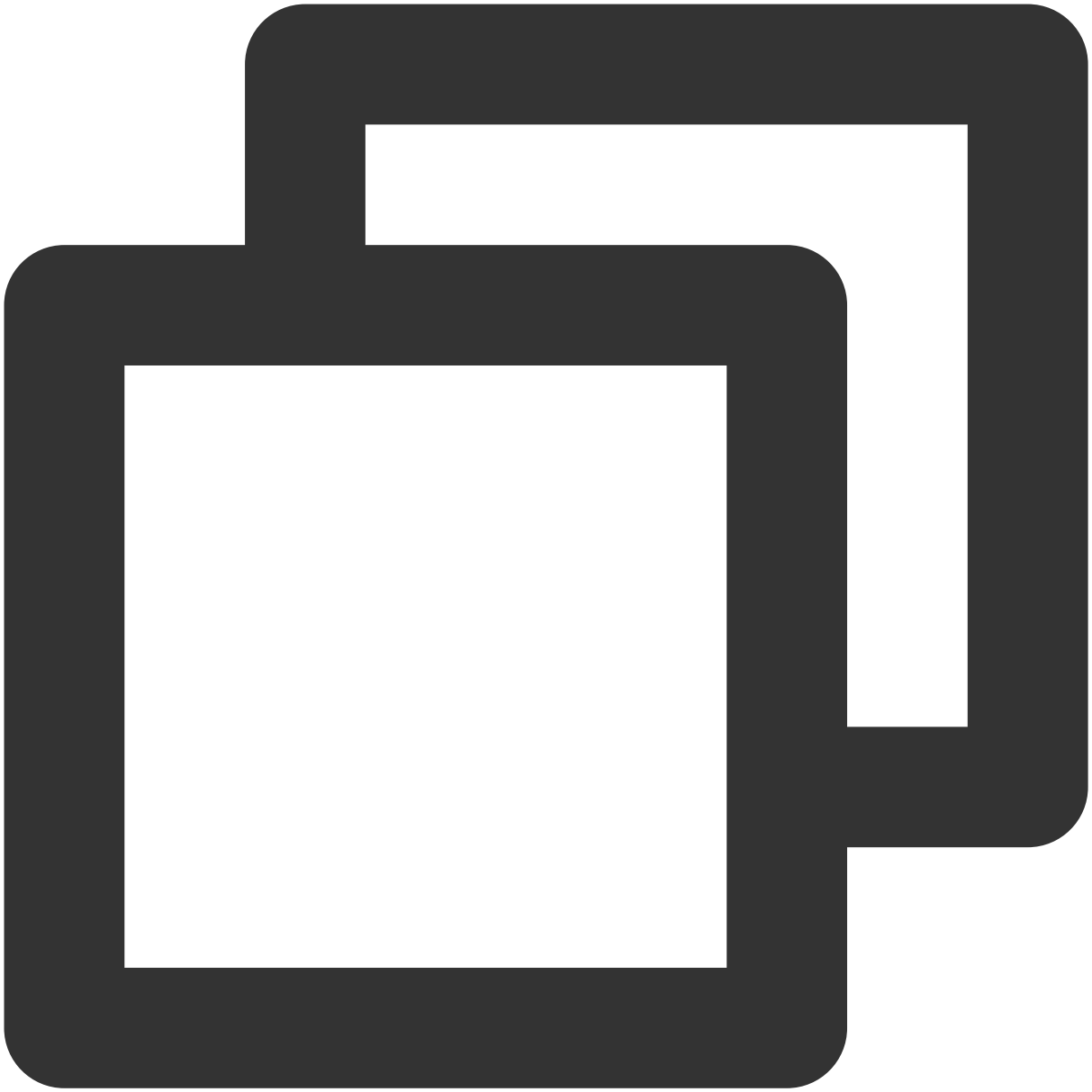
API	说明
文档转 HTML	文档转 HTML 功能支持对多种文档类型的文件生成 HTML 格式预览，满足 PC、App 等多个用户端的文档在线浏览需求，适用于在线教育、企业 OA、在线知识库、网盘文档预览等业务场景。

文档转 HTML

功能说明

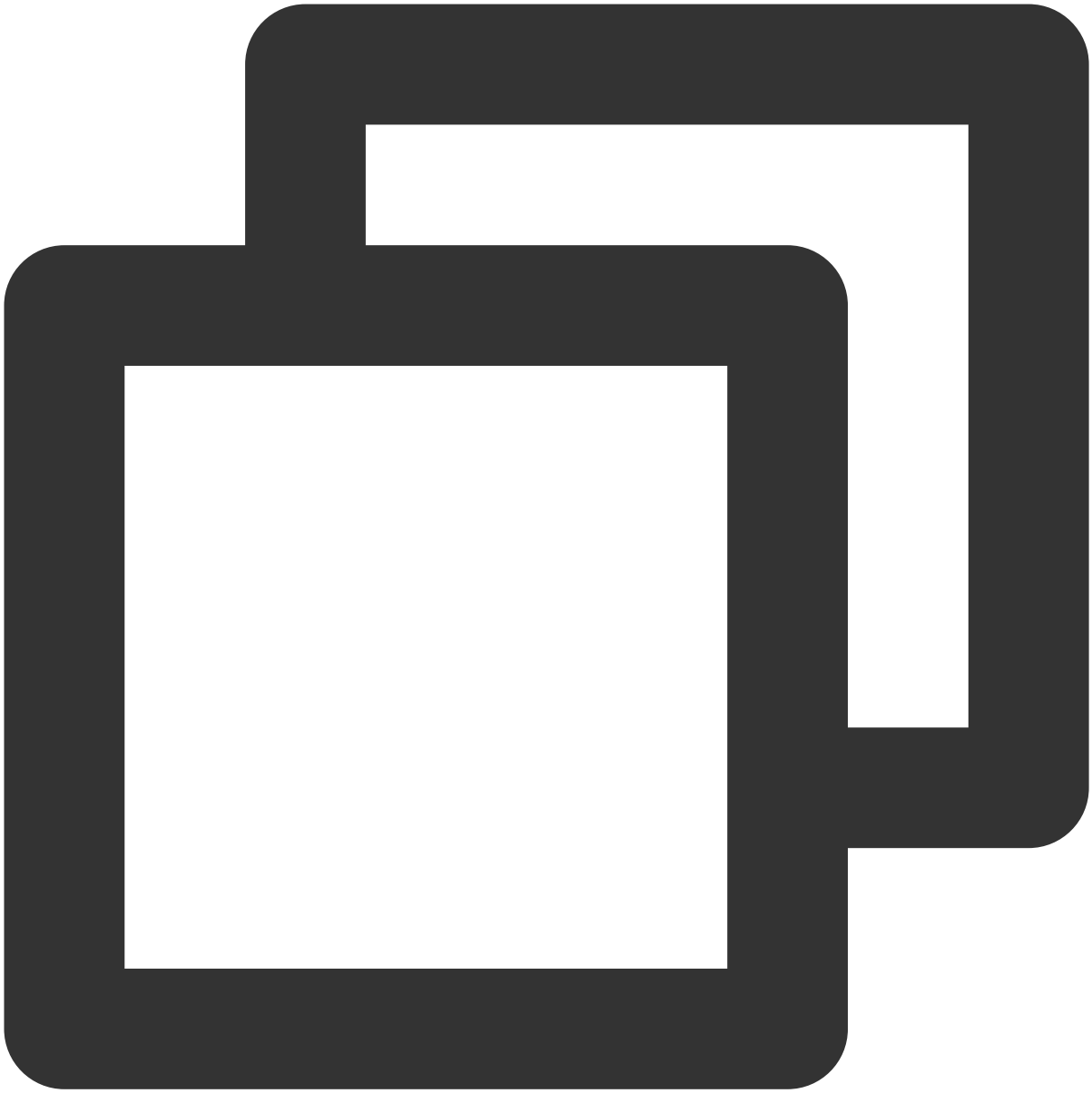
支持对多种文档类型的文件生成 HTML 格式预览。

方法原型



```
func (s *CIService) DocPreviewHTML(ctx context.Context, name string, opt *DocPrevie
```

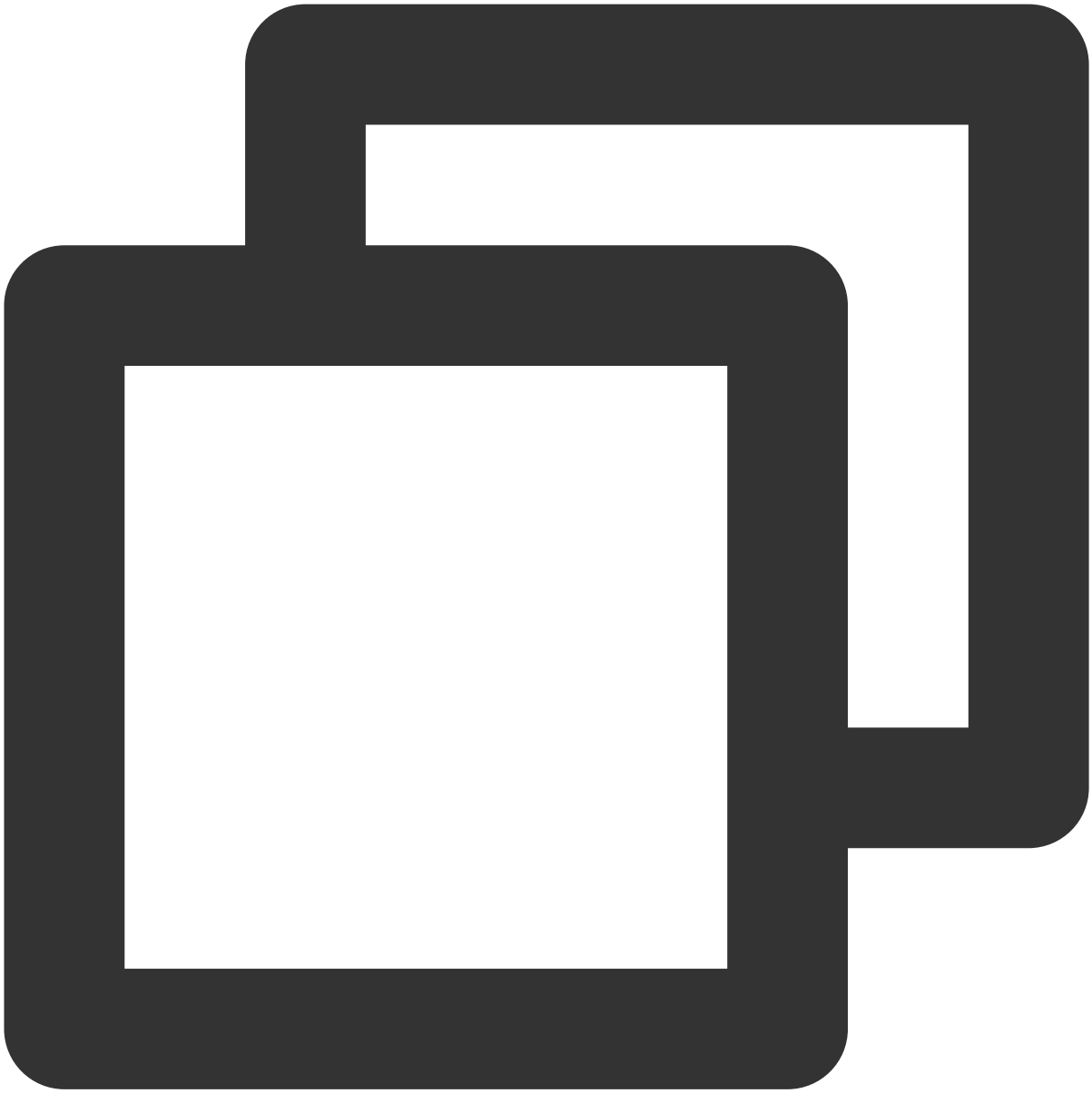
请求示例



```
opt := &cos.DocPreviewHTMLOptions{
    DstType:  "html",
    SrcType:  "ppt",
    Copyable: "1",
    HtmlParams: &cos.HtmlParams{
        CommonOptions: &cos.HtmlCommonParams{
            IsShowTopArea: false,
        },
        PptOptions: &cos.HtmlPptParams{
            IsShowBottomStatusBar: true,
        },
```

```
    },
    Htmlwaterword:  "5pWw5o2u5LiH6LGhLeaWh+aho+mihOiniA==",
    Htmlfillstyle:  "cmdiYSgxMDIsMjA0LDI1NSwwLjMp", // rgba(102,204,255,0.3)
    Htmlfront:      "Ym9sZCAyNXB4IFNlcmlm",         // bold 25px Serif
    Htmlrotate:     "315",
    Htmlhorizontal: "50",
    Htmlvertical:   "100",
}
resp, err := c.CI.DocPreviewHTML(context.Background(), "form.pdf", opt)
```

参数说明



```
type DocPreviewHTMLOptions struct {
    DstType        string      `url:"dstType,omitempty"`
    SrcType        string      `url:"srcType,omitempty"`
    Sign           string      `url:"sign,omitempty"`
    Copyable       string      `url:"copyable,omitempty"`
    HtmlParams     *HtmlParams `url:"htmlParams,omitempty"`
    Htmlwaterword  string      `url:"htmlwaterword,omitempty"`
    Htmlfillstyle  string      `url:"htmlfillstyle,omitempty"`
    Htmlfront      string      `url:"htmlfront,omitempty"`
    Htmlrotate     string      `url:"htmlrotate,omitempty"`
    Htmlhorizontal string      `url:"htmlhorizontal,omitempty"`
```

版权所有：腾讯云计算（北京）有限责任公司

```
    Htmlvertical    string      `url:"htmlvertical,omitempty"`
}

type HtmlParams struct {
    CommonOptions *HtmlCommonParams `json:"commonOptions,omitempty"`
    WordOptions   *HtmlWordParams   `json:"wordOptions,omitempty"`
    PdfOptions    *HtmlPdfParams    `json:"pdfOptions,omitempty"`
    PptOptions    *HtmlPptParams    `json:"pptOptions,omitempty"`
}

type HtmlCommonParams struct {
    IsShowTopArea           bool `json:"isShowTopArea"`
    IsShowHeader            bool `json:"isShowHeader"`
    IsBrowserViewFullscreen bool `json:"isBrowserViewFullscreen"`
    IsIframeViewFullscreen  bool `json:"isIframeViewFullscreen"`
}

type HtmlWordParams struct {
    IsShowDocMap          bool `json:"isShowDocMap"`
    IsBestScale           bool `json:"isBestScale"`
    IsShowBottomStatusBar bool `json:"isShowBottomStatusBar"`
}

type HtmlPdfParams struct {
    IsShowComment         bool `json:"isShowComment"`
    IsInSafeMode          bool `json:"isInSafeMode"`
    IsShowBottomStatusBar bool `json:"isShowBottomStatusBar"`
}

type HtmlPptParams struct {
    IsShowBottomStatusBar bool `json:"isShowBottomStatusBar"`
}
```

名称	参数说明	类型	是否必选
DstType	转换输出目标文件类型，文档 HTML 预览固定为 html（需为小写字母）	String	是
SrcType	指定目标文件类型，支持的文件类型请见 文档转 HTML 文档	String	否
Sign	对象下载签名，如果预览的对象为私有读时，需要传入签名，详情请参见 请求签名 文档</br>注意：需要进行 urlencode	String	否
Copyable	是否可复制。默认为可复制，填入值为1；不可复制，填入值为0	String	否

HtmlParams	自定义配置参数，json结构，需要经过 URL 安全 的 Base64 编码，默认配置为：{ commonOptions: { isShowTopArea: true, isShowHeader: true } }，支持的配置参考 自定义配置项说明	String	否
Htmlwaterword	水印文字，需要经过 URL 安全 的 Base64 编码，默认为空	String	否
Htmlfillstyle	水印 RGBA（颜色和透明度），需要经过 URL 安全 的 Base64 编码，默认为：rgba(192,192,192,0.6)	String	否
Htmlfront	水印文字样式，需要经过 URL 安全 的 Base64 编码，默认为：bold 20px Serif	String	否
Htmlrotate	水印文字旋转角度，0 - 360，默认315度	String	否
Htmlhorizontal	水印文字水平间距，单位 px，默认为50	String	否
Htmlvertical	水印文字垂直间距，单位 px，默认为100	String	否

版权所有：腾讯云计算（北京）有限责任公司

内容审核
图片审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核的相关的 API 概览以及 SDK 示例代码。

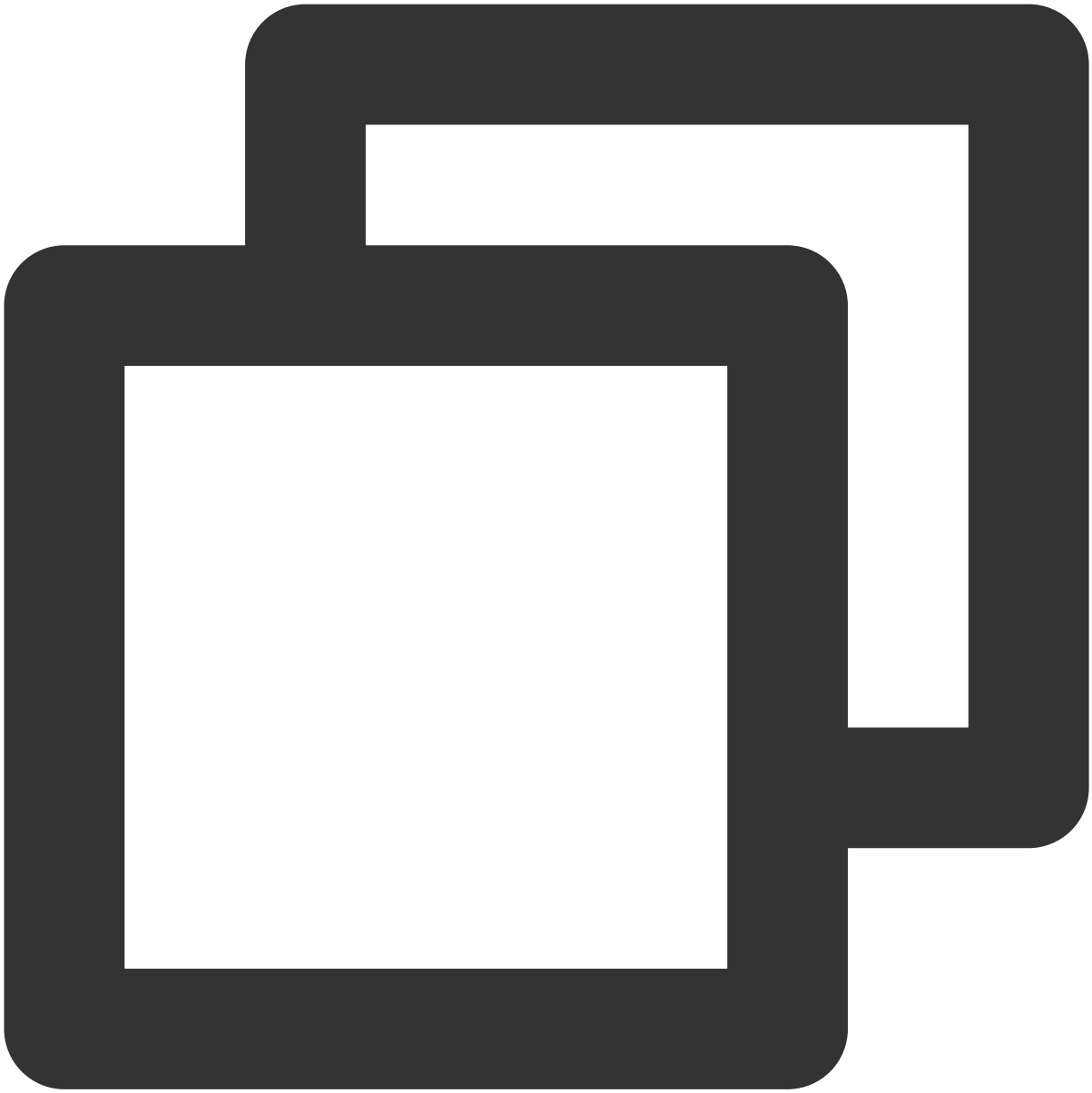
API	操作描述
图片单次审核	借助数据万象的内容审核接口，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描
图片批量审核	借助数据万象的内容审核接口，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描
查询图片审核任务结果	用于查询图片审核任务详情

图片单次审核

功能说明

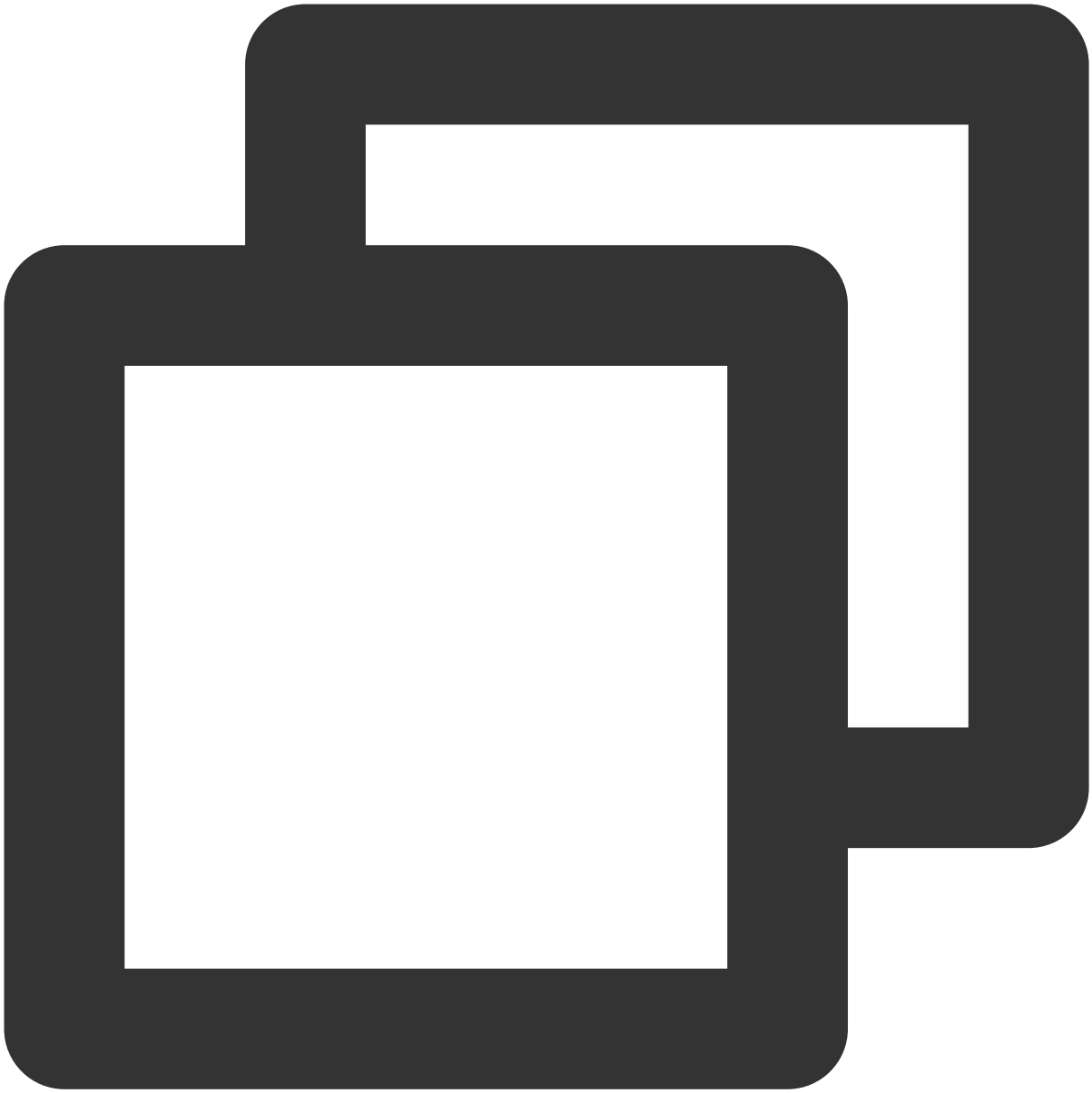
图片审核的存量扫描功能通过借助数据万象的持久化处理接口，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描。图片审核请求包属于 GET 请求。

方法原型



```
// 基本操作
func (s *CIService) ImageRecognition(ctx context.Context, name string, DetectType s
// 支持指定更多参数
func (s *CIService) ImageAuditing(ctx context.Context, name string, opt *ImageRecog
```

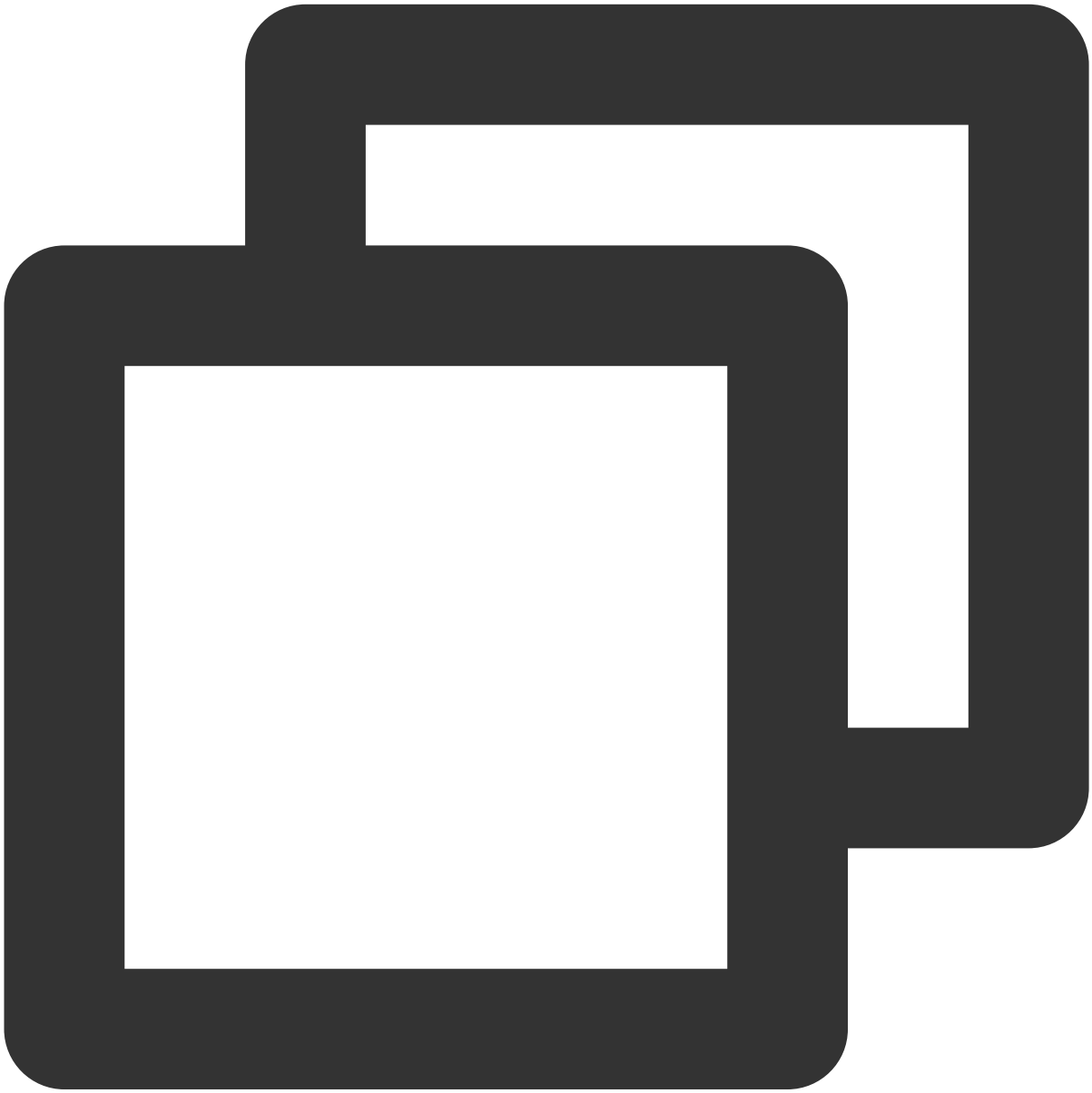
请求示例



```
DetectType := "porn,ads"
name := "test.jpg"
res, _, err := c.CI.ImageRecognition(context.Background(), name, DetectType)


opt := &cos.ImageRecognitionOptions{
    CIProcess: "sensitive-content-recognition",
    DetectType: "porn,ads",
    DetectUrl:   "http://www.example.com/test.jpg",
    BizType:     "ce25f391a72e11eb99f********",
}
```

```
res, _, err := c.CI.ImageAuditing(context.Background(), "", opt)
```

参数说明



```
type ImageRecognitionOptions struct {
    CIProcess        string
    DetectType       string
    DetectUrl        string
    Interval         int
    MaxFrames        int
    BizType          string
```

```
    LargeImageDetect int
    DataId           string
    Async            int
    Callback         string
}
```

参数名称	描述	类型	是否必选
name	COS 存储桶中的图片文件名称，COS 存储桶由Host指定，例如在北京的 examplebucket-1250000000 存储桶中的目录 test 下的文件 img.jpg，则 Host 填写 `examplebucket-1250000000.cos.ap-beijing.myqcloud.com`，ObjectKey填写 `test/img.jpg`。如使用 ImageAuditing 接口审核 url 图片链接，传空字符串。	String	是
CIProcess	标识数据处理功能的字段，内容审核的值为：`sensitive-content-recognition`。	String	是
BizType	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置审核策略。	String	否
DetectType	审核类型，拥有 porn（涉黄识别）、ads（广告识别）。用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核。如您有更多场景的审核需要，请使用 BizType 参数。	String	否
DetectUrl	您可以通过填写 detect-url 审核任意公网可访问的图片链接。不填写 detect-url 时，后台会默认审核 ObjectKey。填写了 detect-url 时，后台会审核 detect-url 链接，无需再填写 ObjectKey。detect-url 示例：`http://www.example.com/abc.jpg`。	String	否
Interval	审核 GIF 动图时，可使用该参数进行截帧配置，代表截帧的间隔。例如值设为5，则表示从第1帧开始截取，每隔5帧截取一帧，默认值5。	Integer	否
MaxFrames	针对 GIF 动图审核的最大截帧数量，需大于0。例如值设为5，则表示最大截取5帧，默认值为5。	Integer	否
LargeImageDetect	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。注：压缩最大支持32MB的图片，且会收取图片压缩费用。对于 GIF 等动态图过大时，压缩时间较长，可能会导致审核超时失败。	Integer	否
DataId	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否

Async	是否异步进行审核，取值 0：同步返回结果，1：异步进行审核，默认为0。	String	否
Callback	审核结果（Detail版本）以回调形式发送至您的回调地址，异步审核时生效，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否

结果说明

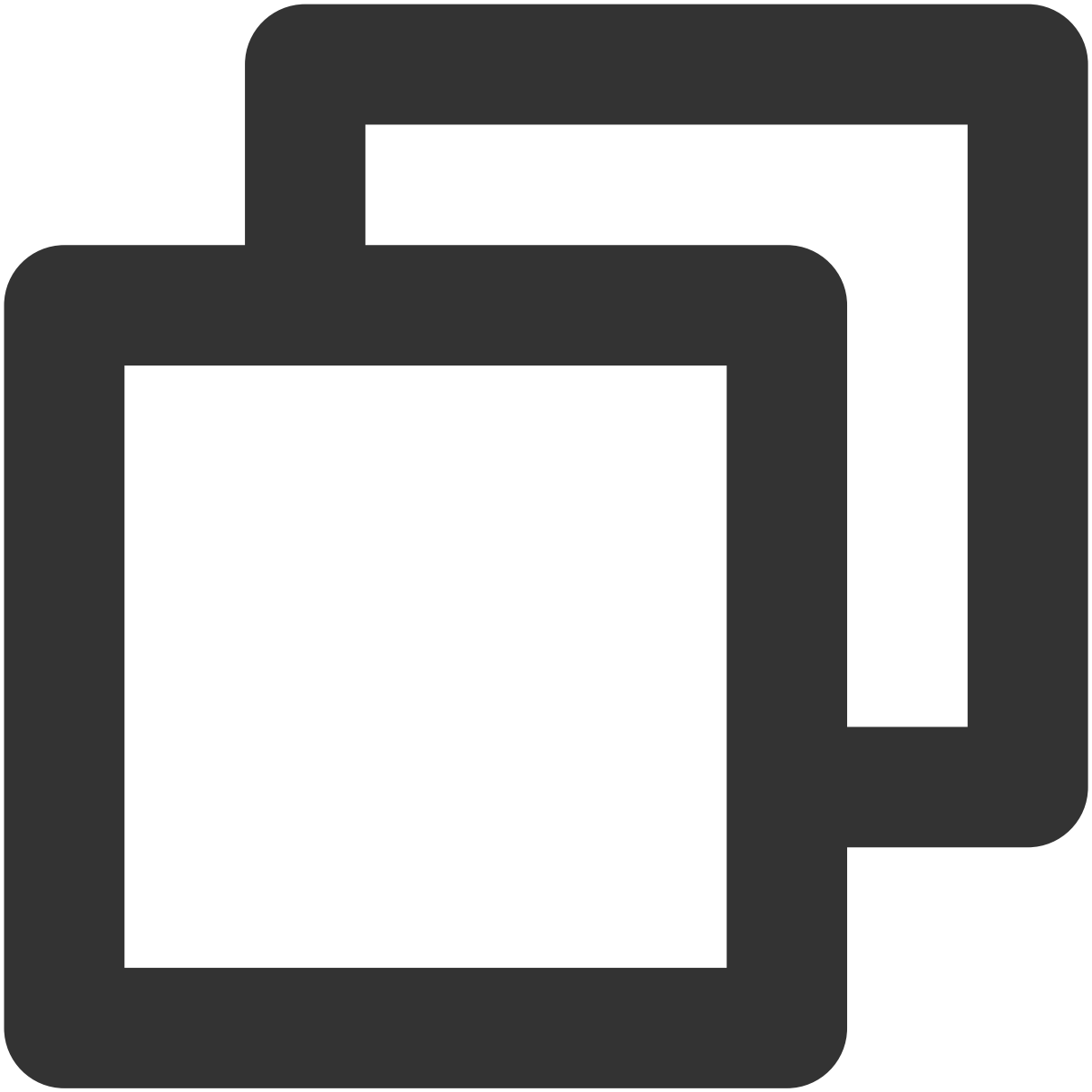
调用 ImageRecognition 或 ImageAuditing 函数，会解析 api 返回的 xml 内容到 ImageRecognitionResult 结构，具体返回参数可查看 图片单次审核 文档。

图片批量审核

功能说明

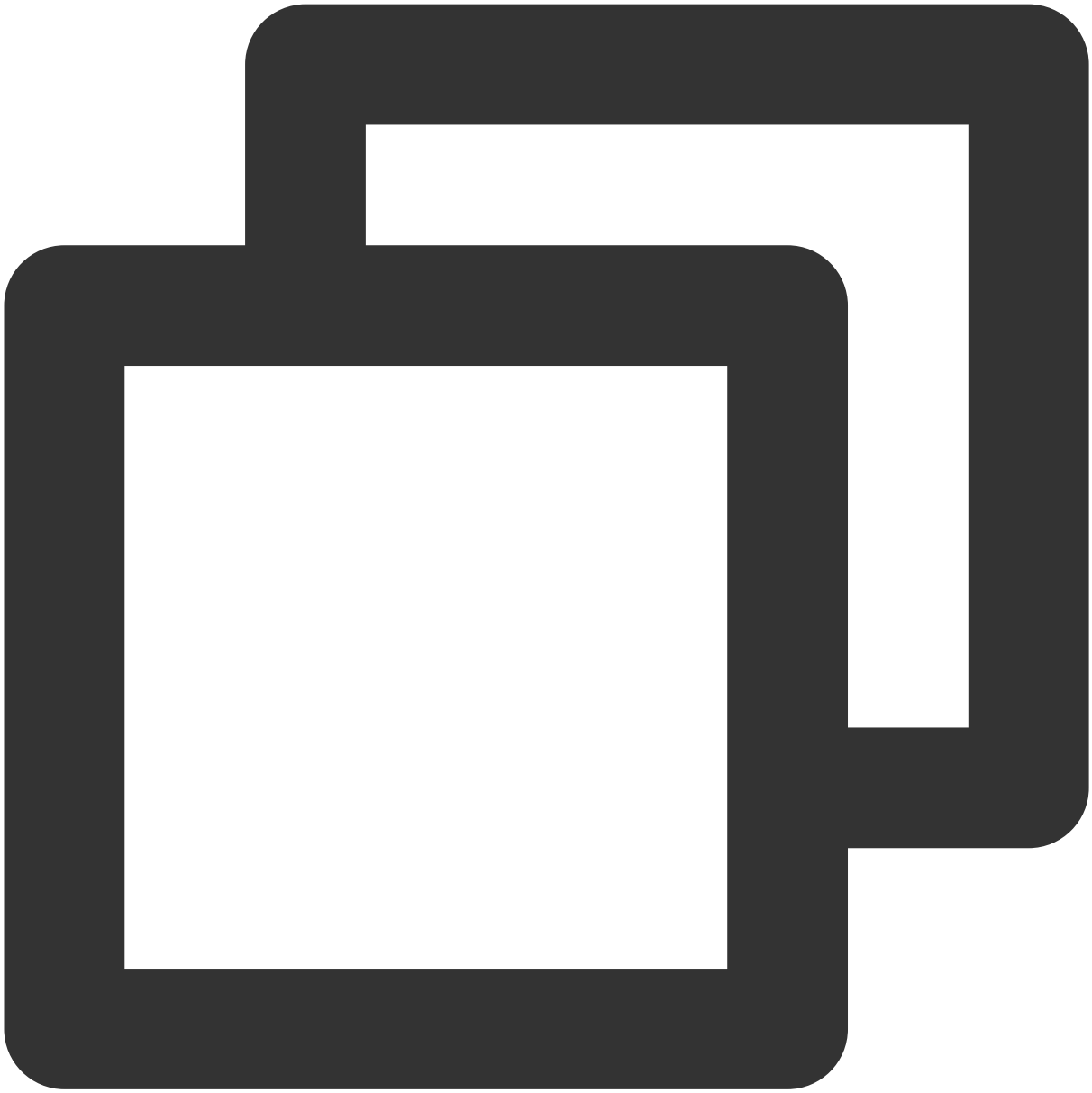
图片批量审核接口为同步请求方式，您可以通过本接口对多个图片文件进行内容审核。该接口属于 POST 请求。

方法原型



```
// 图片批量审核
func (s *CIService) BatchImageAuditing(ctx context.Context, opt *BatchImageAuditing
```

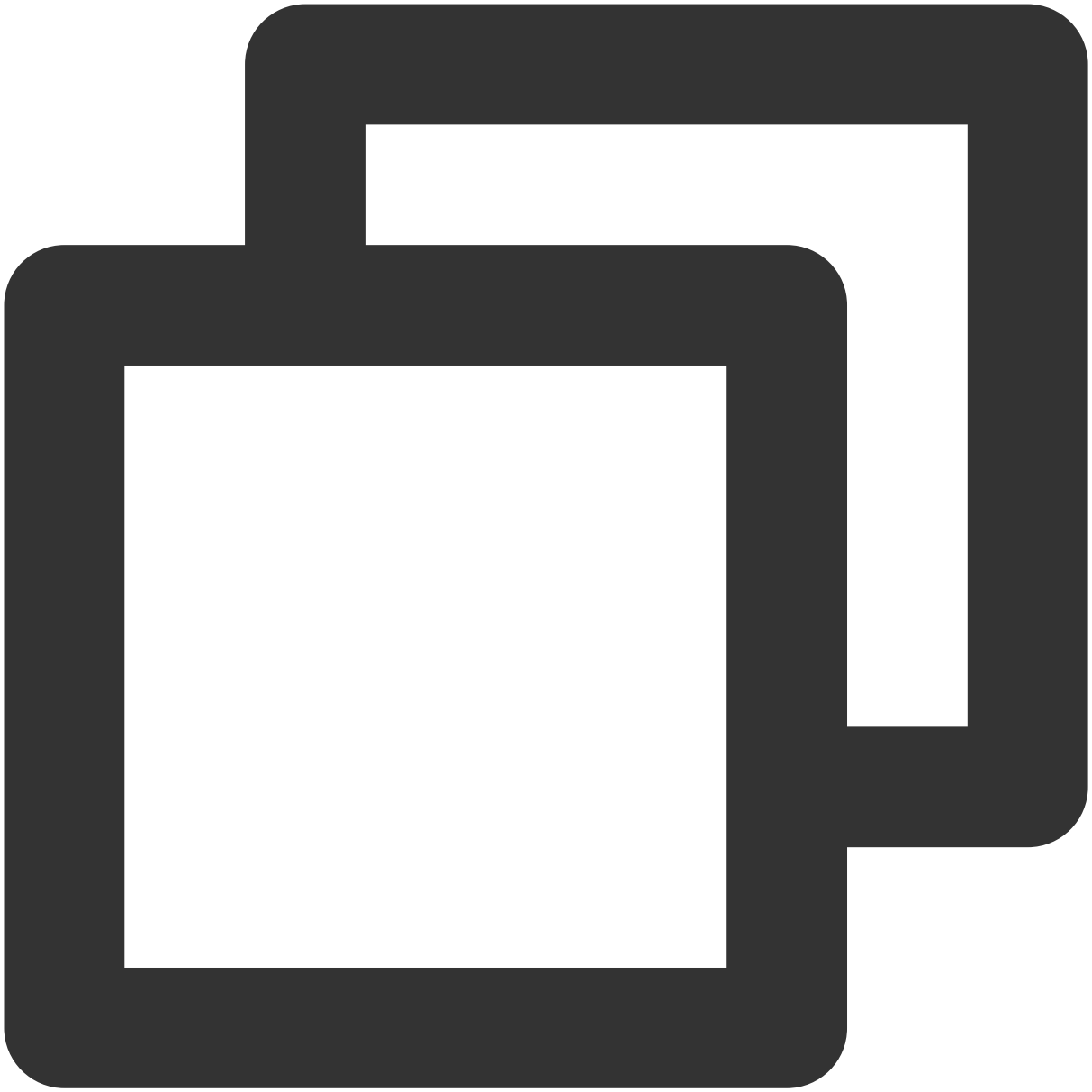
请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION 修改为真实的信息
// CI 任务需要提供 CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        },
})
```

```
opt := &cos.BatchImageAuditingOptions{
    Input: []cos.ImageAuditingInputOptions{
        cos.ImageAuditingInputOptions{
                DataId: "1",
                Object: "test.jpg",
        },
        cos.ImageAuditingInputOptions{
                DataId: "2",
                Url: "http://www.example.com/test.jpg",
        },
    },
    Conf: &cos.ImageAuditingJobConf{
        DetectType: "Porn,Ads",
    },
}
res, _, err := c.CI.BatchImageAuditing(context.Background(), opt)
```

参数说明



```
type ImageAuditingInputOptions struct {
    DataId           string
    Object           string
    Url              string
    Interval         int
    MaxFrames        int
    LargeImageDetect int
    UserInfo         *UserExtraInfo
}
type UserExtraInfo struct {
    TokenId  string
```

```
    Nickname string
    DeviceId string
    AppId    string
    Room     string
    IP       string
    Type     string
}
type ImageAuditingJobConf struct {
    DetectType string
    BizType    string
    Async      int
    Callback   string
}
type BatchImageAuditingOptions struct {
    Input   []ImageAuditingInputOptions
    Conf    *ImageAuditingJobConf
}
```

参数名称	描述	类型	是否必选
DataId	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否
Object	存储在 COS 存储桶中的图片文件名称，例如在目录 test 中的文件 image.jpg，则文件名称为 test/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Url	图片文件的链接地址，例如 http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Interval	截帧频率，GIF 图检测专用，默认值为5，表示从第一帧（包含）开始每隔5帧截取一帧。	Integer	否
MaxFrames	最大截帧数量，GIF 图检测专用，默认值为5，表示只截取 GIF 的5帧图片进行审核，必须大于0。	Integer	否
BizType	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否
DetectType	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以,分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	String	否
Async	是否异步进行审核，取值 0：同步返回结果，1：异步进行审核，默认为0。	String	否

Callback	审核结果（Detail版本）以回调形式发送至您的回调地址，异步审核时生效，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否
LargeImageDetect	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。注：压缩最大支持32MB的图片，且会收取图片压缩费用。对于 GIF 等动态图过大时，压缩时间较长，可能会导致审核超时失败。	Integer	否
UserInfo	用户业务字段。	Object	否

结果说明

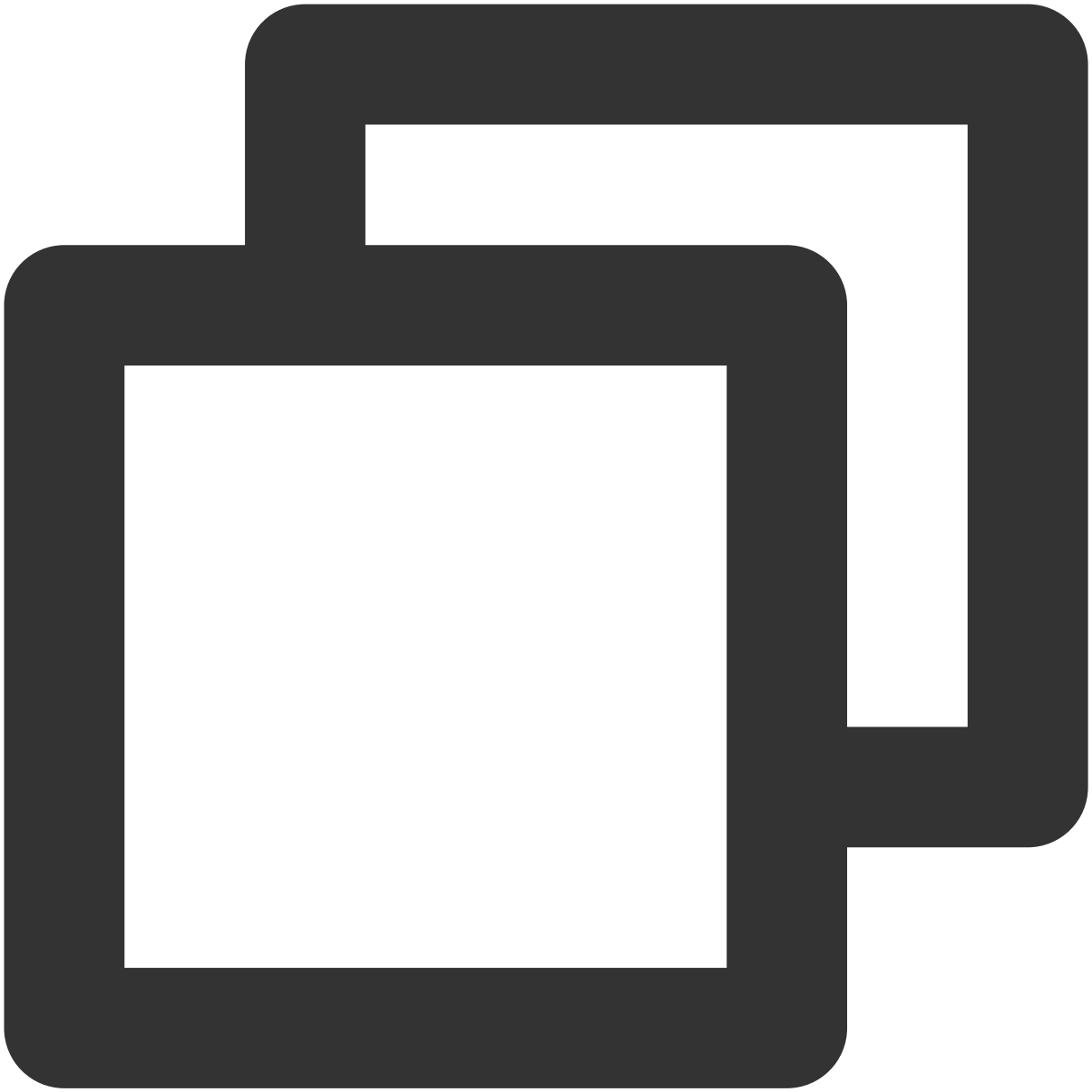
调用 BatchImageAuditing 函数，会解析 api 返回的 xml 内容到 BatchImageAuditingJobResult 结构，具体返回参数可查看 图片批量审核 文档。

查询图片审核任务结果

功能说明

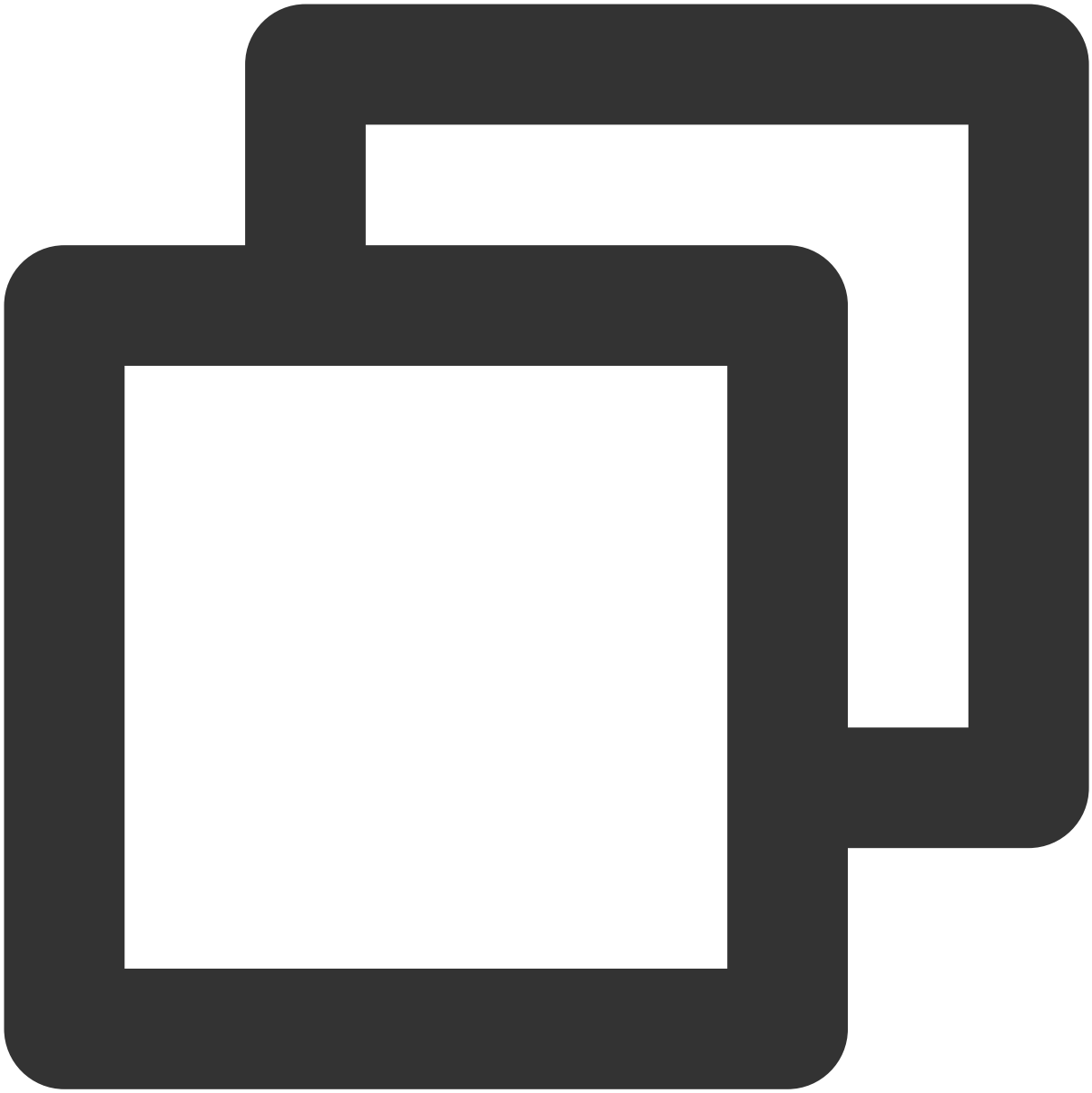
用于查询一个图片审核任务的结果详情。该接口属于 GET 请求。

方法原型



```
// 图片审核-查询任务
func (s *CIService) GetImageAuditingJob(ctx context.Context, jobid string) (*GetIma
```

请求示例



```
jobId := "iace25f391a72e11eb99f********"
res, _, err := c.CI.GetImageAuditingJob(context.Background(), jobId)
```

参数说明

参数名称	参数描述	类型
jobId	任务 ID	String

结果说明

调用 GetImageAuditingJob 函数，会解析 api 返回的 xml 内容到 GetImageAuditingJobResult 结构，具体返回参数可查看 查询图片审核任务结果 文档。

视频审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

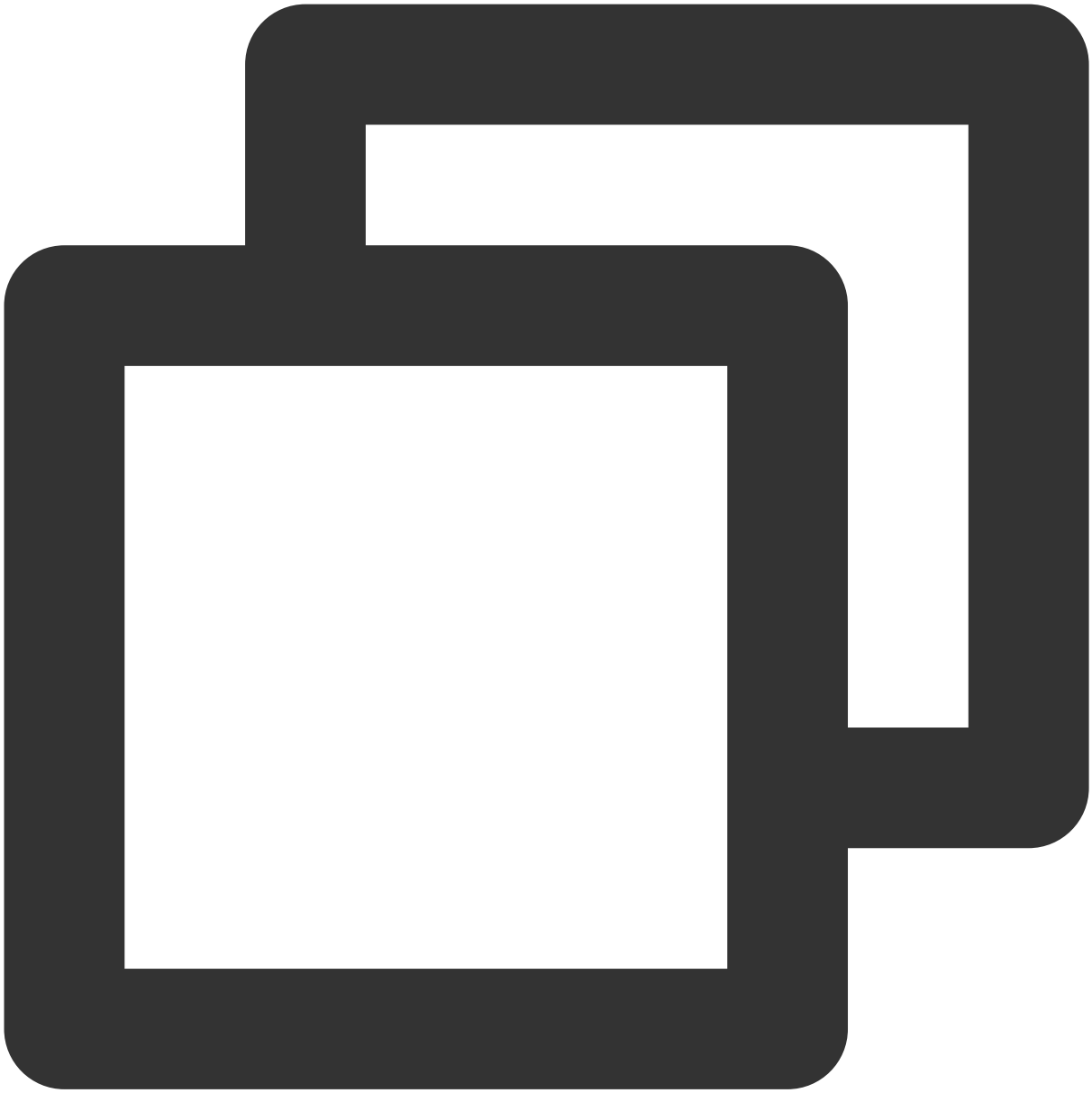
本文档提供关于视频审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
创建任务	用于提交一个视频审核任务
查询任务	用于查询指定的视频审核任务

创建任务

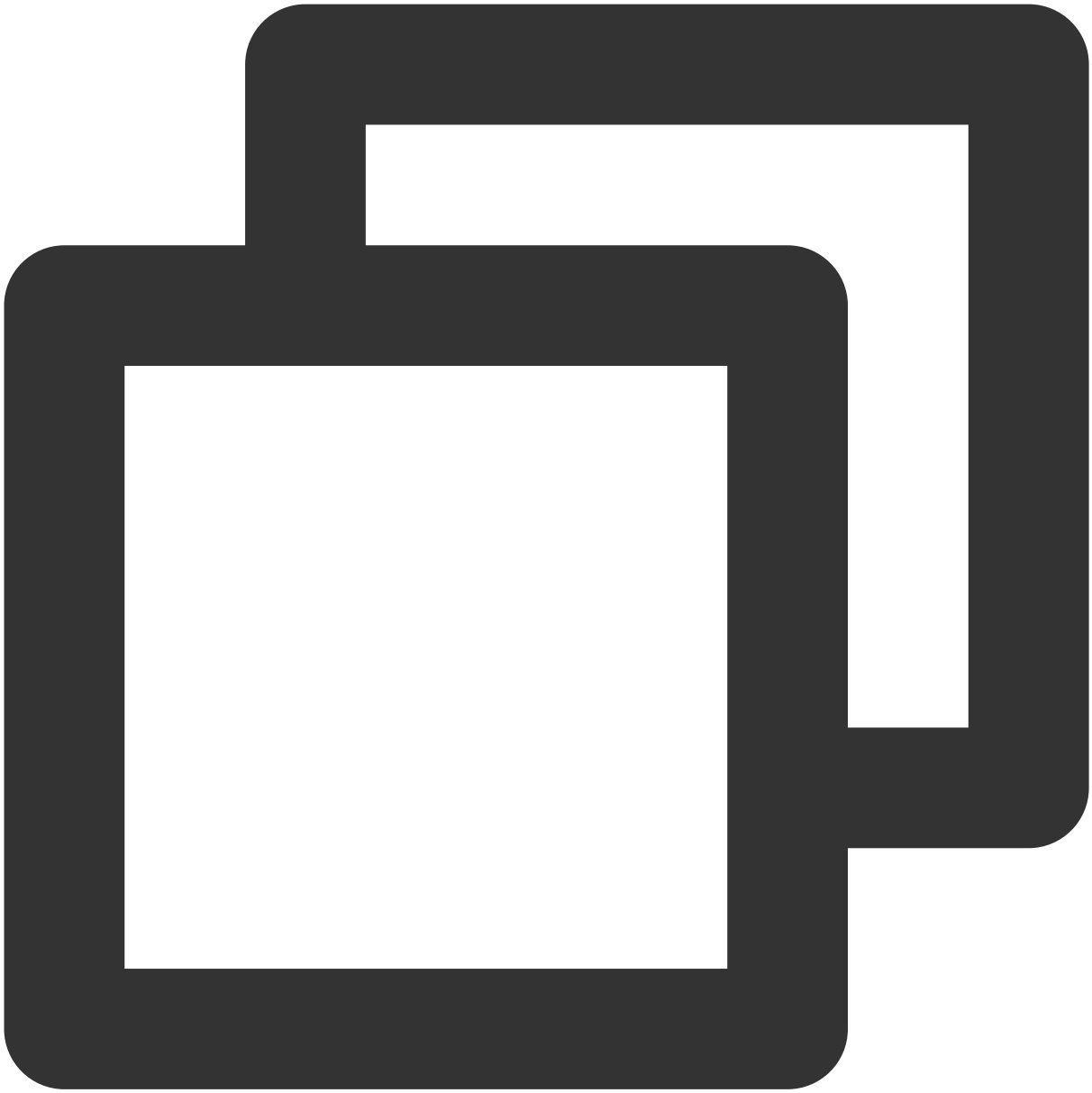
PutVideoAuditingJob 接口用于提交一个视频审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

方法原型



```
func (s *CIService) PutVideoAuditingJob(ctx context.Context, opt *PutVideoAuditingJ
```

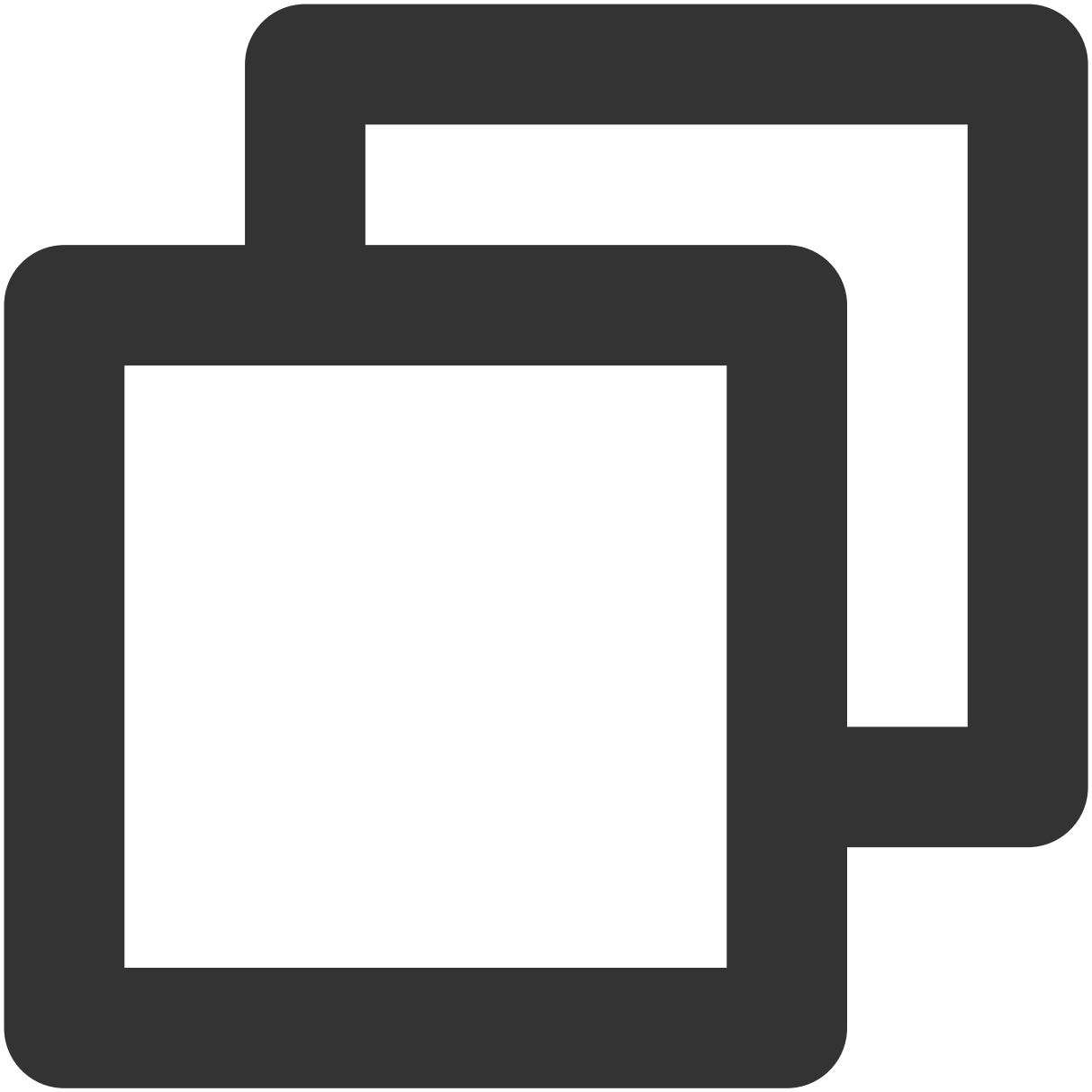
请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        }
})
```

```
opt := &cos.PutVideoAuditingJobOptions{
    InputObject: "demo.mp4",
    Conf: &cos.VideoAuditingJobConf{
        DetectType: "Porn,Ads",
        Snapshot: &cos.PutVideoAuditingJobSnapshot{
            Mode:         "Interval",
            TimeInterval: 50.5,
            Count:        100,
        },
    },
}
res, _, err := c.CI.PutVideoAuditingJob(context.Background(), opt)
```

参数说明



```
type PutVideoAuditingJobOptions struct {
    InputObject   string
    InputUrl      string
    InputDataId   string
    InputUserInfo *UserExtraInfo
    Conf          *VideoAuditingJobConf
    Type          string
}
type UserExtraInfo struct {
    TokenId  string
    Nickname string
```

```
    DeviceId string
    AppId    string
    Room     string
    IP       string
    Type     string
}
type VideoAuditingJobConf struct {
    DetectType      string
    Snapshot        *PutVideoAuditingJobSnapshot
    Callback        string
    CallbackVersion string
    CallbackType    int
    BizType         string
    DetectContent   int
}
type PutVideoAuditingJobSnapshot struct {
    Mode         string
    Count        int
    TimeInterval float32
}
```

参数名称	参数描述	类型
InputObject	当前 COS 存储桶中的视频文件名称，例如在目录 test 中的文件 video.mp4，则文件名称为 `test/video.mp4` 。Object 和 Url 只能选择其中一种。	String
InputUrl	视频文件的链接地址，例如 `http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/test.mp4` 。Object 和 Url 只能选择其中一种。	String
InputDataId	视频标识，该字段在结果中返回原始内容，长度限制为512字节。	String
InputUserInfo	用户业务字段。	Object
Type	审核的任务类型，直播流审核固定为 live_video。	String
Conf	审核规则配置。	Struct
BizType	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见设置审核策略。	String
DetectType	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	String
Snapshot	视频画面的审核通过视频截帧能力截取出一定量的截图，通过对截图逐一	Struct

	审核而实现的，该参数用于指定视频截帧的配置。	
Callback	审核结果以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如：`http://www.callback.com`。	String
CallbackVersion	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String
CallbackType	回调片段类型，有效值：1（回调全部截帧和音频片段）、2（仅回调违规截帧和音频片段）。默认为 1。	Integer
DetectContent	用于指定是否审核视频声音，当值为0时：表示只审核视频画面截图；值为1时：表示同时审核视频画面截图和视频声音。默认值为0。	Integer
Mode	截帧模式，默认值为Interval。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片。Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。Fps 模式：TimeInterval 表示每秒截取多少帧，未设置 TimeInterval 时，表示截取所有帧，Count 表示共截取多少帧。	String
Count	视频截帧数量，范围为(0, 10000]。	Integer
TimeInterval	视频截帧频率，范围为(0, 60]，单位为秒，支持 float 格式，执行精度精确到毫秒。	Float

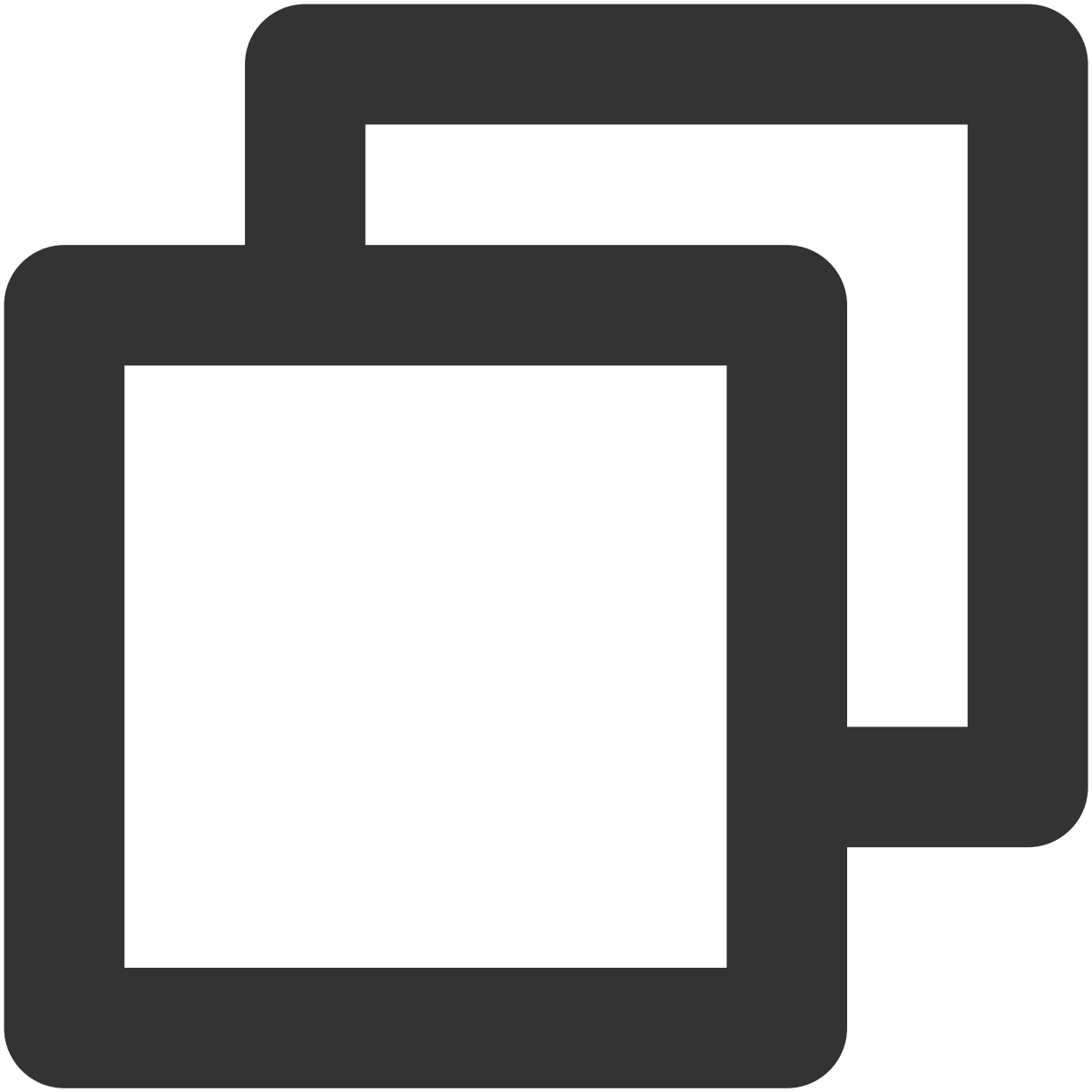
结果说明

调用 PutVideoAuditingJob 函数，会解析 api 返回的 xml 内容到 PutVideoAuditingJobResult 结构，具体返回参数可查看 提交视频审核任务 文档。

查询任务

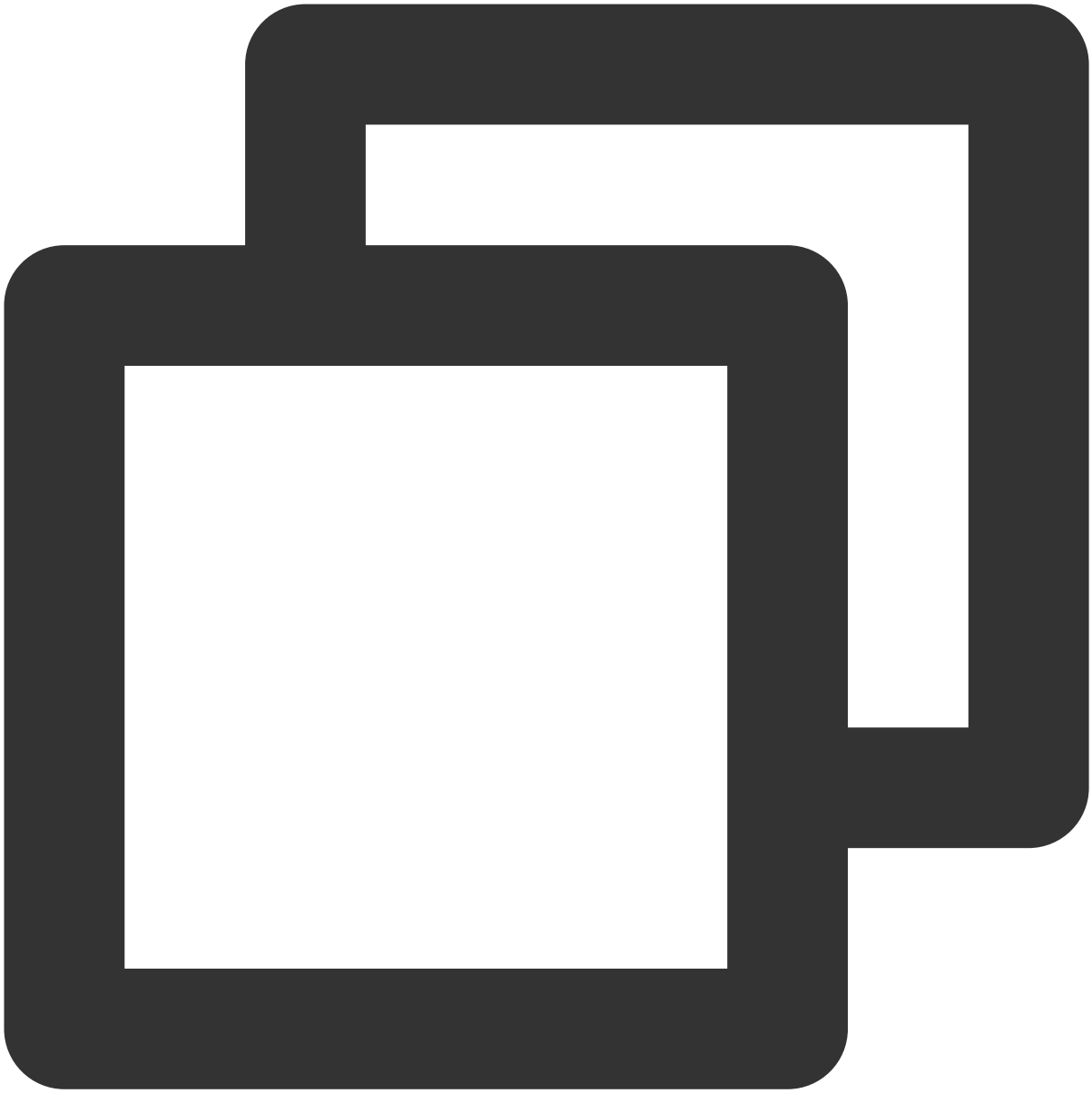
GetVideoAuditingJob 接口用来查询指定的视频审核任务。您可以根据视频审核任务的 JobId 来查询视频审核结果。

方法原型



```
func (s *CIService) GetVideoAuditingJob(ctx context.Context, jobid string) (*GetVid
```

请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        }
})
```

```
jobId := "avce25f391a72e11eb99f********"
res, _, err := c.CI.GetVideoAuditingJob(context.Background(), jobId)
```

参数说明

参数名称	参数描述	类型
jobId	任务 ID	String

结果说明

调用 GetVideoAuditingJob 函数，会解析 api 返回的 xml 内容到 GetVideoAuditingJobResult 结构，具体返回参数可查看 查询视频审核任务结果 文档。

音频审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

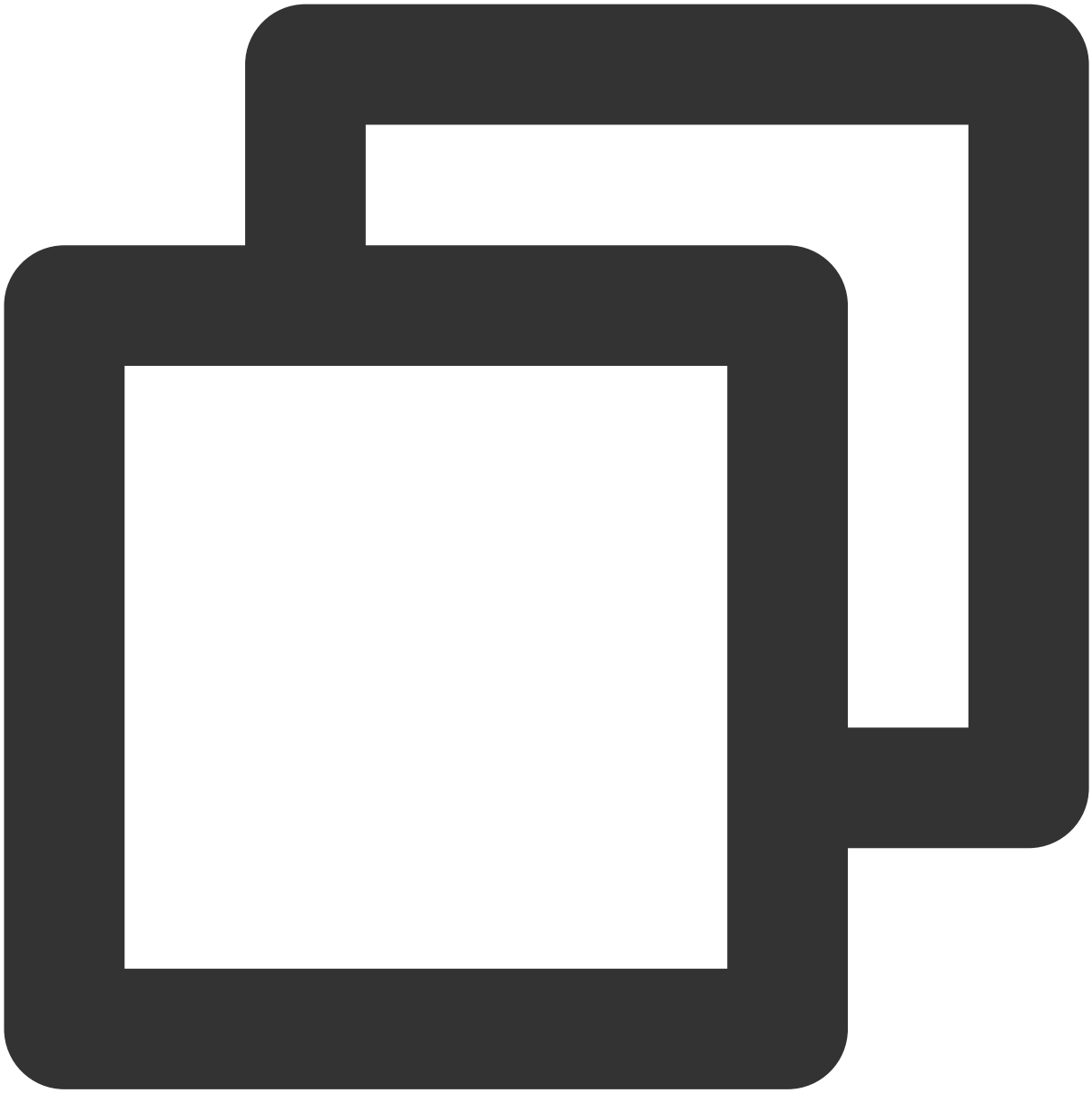
本文档提供关于音频审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
创建任务	用于提交一个音频审核任务
查询任务	用于查询指定的音频审核任务

创建任务

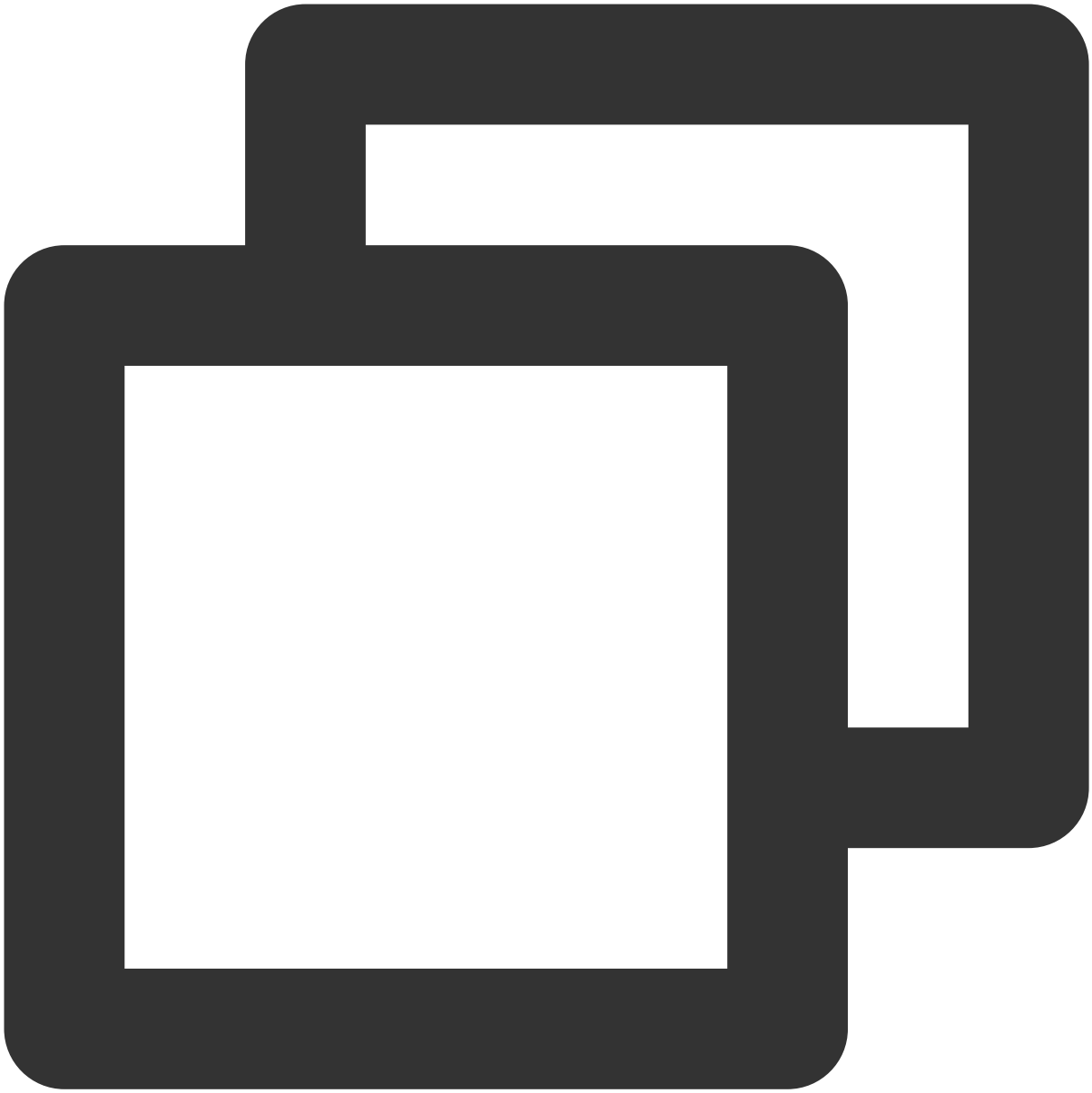
PutAudioAuditingJob 接口用来提交一个音频审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

方法原型



```
func (s *CIService) PutAudioAuditingJob(ctx context.Context, opt *PutAudioAuditingJ
```

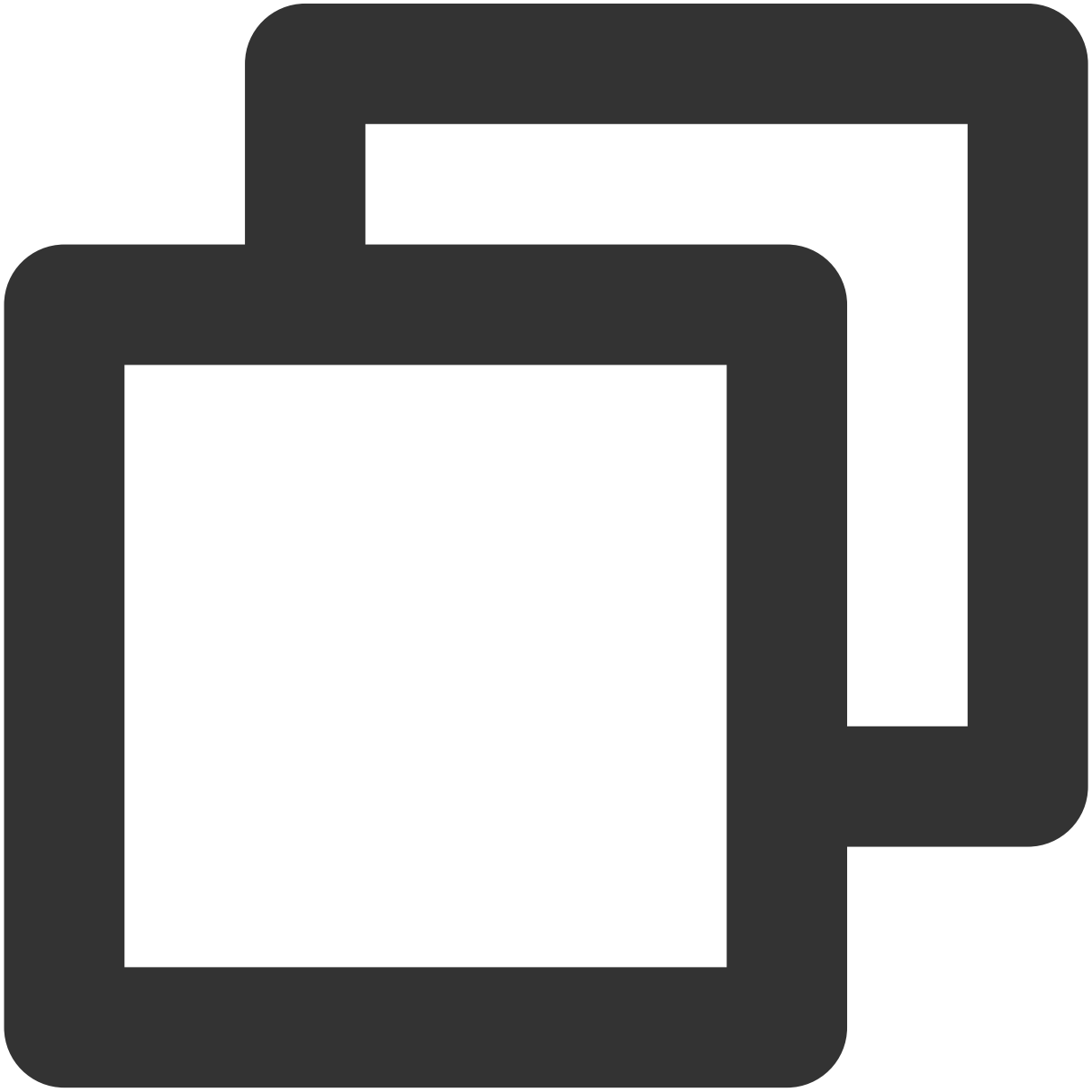
请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        }
})
```

```
opt := &cos.PutAudioAuditingJobOptions{
    InputObject: "test.mp3",
    Conf: &cos.AudioAuditingJobConf{
        DetectType: "Porn,Ads",
    },
}
res, _, err := c.CI.PutAudioAuditingJob(context.Background(), opt)
```

参数说明



```
type PutAudioAuditingJobOptions struct {
```

```
    InputObject   string
    InputUrl      string
    InputDataId   string
    InputUserInfo *UserExtraInfo
    Conf          *AudioAuditingJobConf
}
type UserExtraInfo struct {
    TokenId  string
    Nickname string
    DeviceId string
    AppId    string
    Room     string
    IP       string
    Type     string
}
type AudioAuditingJobConf struct {
    DetectType      string
    Callback        string
    CallbackVersion string
    BizType         string
}
```

参数名称	参数描述	类型
InputObject	存储在 COS 存储桶中的音频文件名称，例如在目录 test 中的文件 audio.mp3，则文件名称为 `test/audio.mp3` 。Object 和 Url 只能选择其中一种。	String
InputUrl	音频文件的链接地址，例如 `http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/audio.mp3` 。Object 和 Url 只能选择其中一种。	String
InputDataId	音频标识，该字段在结果中返回原始内容，长度限制为512字节。	String
InputUserInfo	用户业务字段。	Object
Conf	审核规则配置。	Struct
BizType	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置审核策略。	String
DetectType	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	String
Callback	审核结果以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String

CallbackVersion	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String

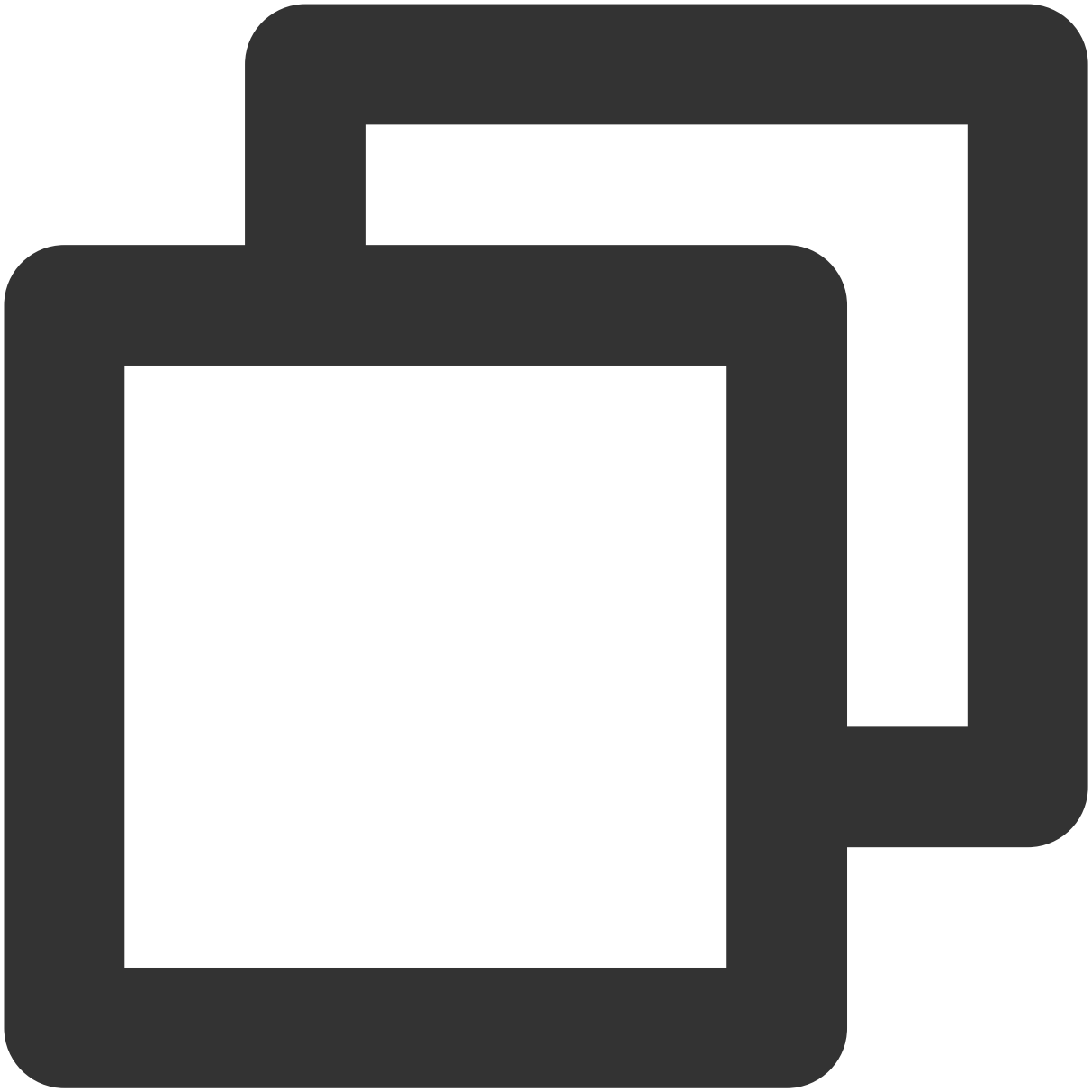
结果说明

调用 PutAudioAuditingJob 函数，会解析 api 返回的 xml 内容到 PutAudioAuditingJobResult 结构，具体返回参数可查看 提交音频审核任务 文档。

查询任务

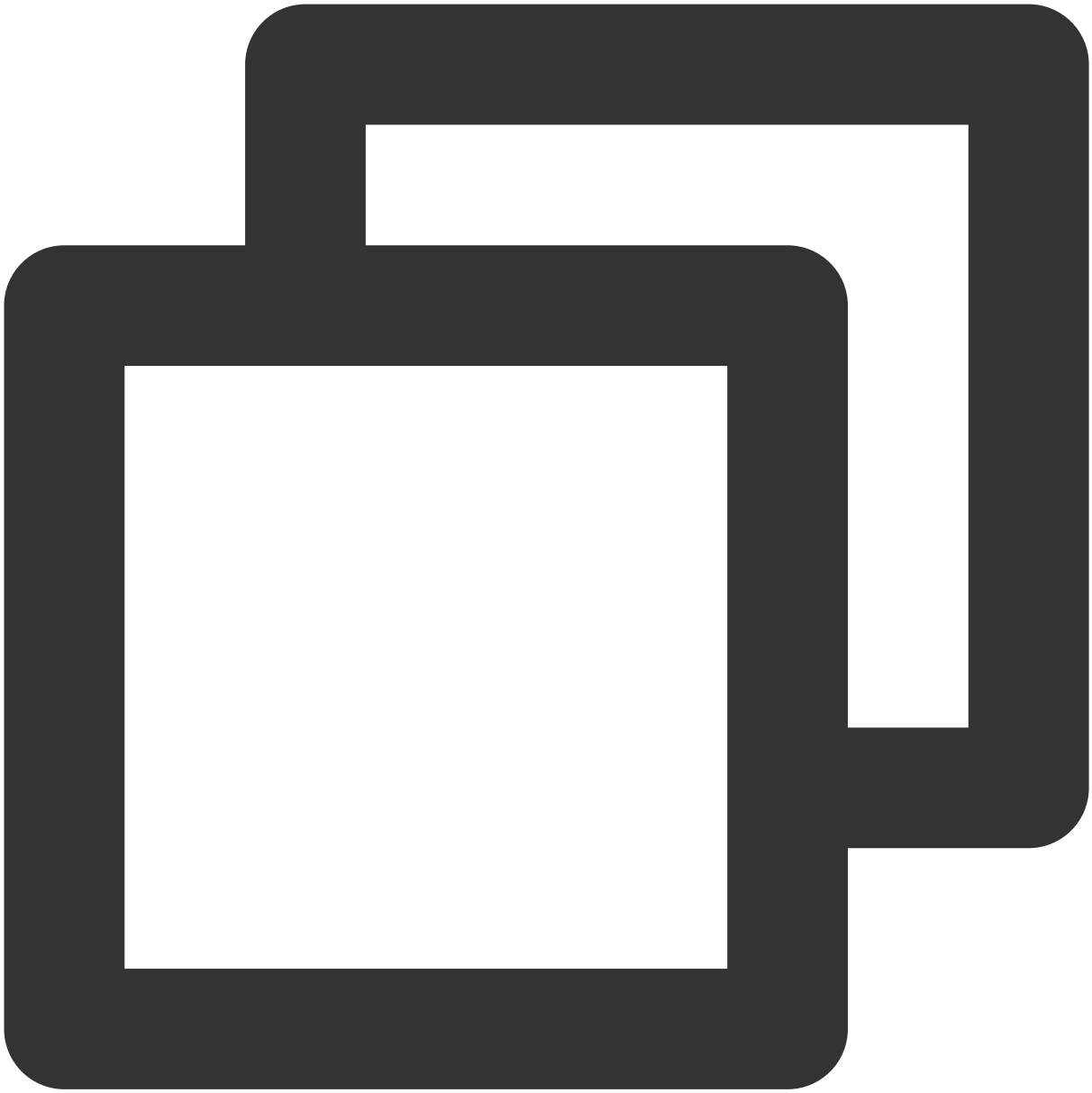
GetAudioAuditingJob 接口用来查询指定的音频审核任务。您可以根据音频审核任务的 JobId 来查询音频审核结果。

方法原型



```
func (s *CIService) GetAudioAuditingJob(ctx context.Context, jobid string) (*GetAud
```

请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        }
})
```

```
jobId := "sace25f391a72e11eb99f********"
res, _, err := c.CI.GetAudioAuditingJob(context.Background(), jobId)
```

参数说明

参数名称	参数描述	类型
jobId	任务 ID。	String

结果说明

调用 GetAudioAuditingJob 函数，会解析 api 返回的 xml 内容到 GetAudioAuditingJobResult 结构，具体返回参数可查看 查询音频审核任务结果 文档。

文本审核

最近更新时间：2024-03-04 16:55:55

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

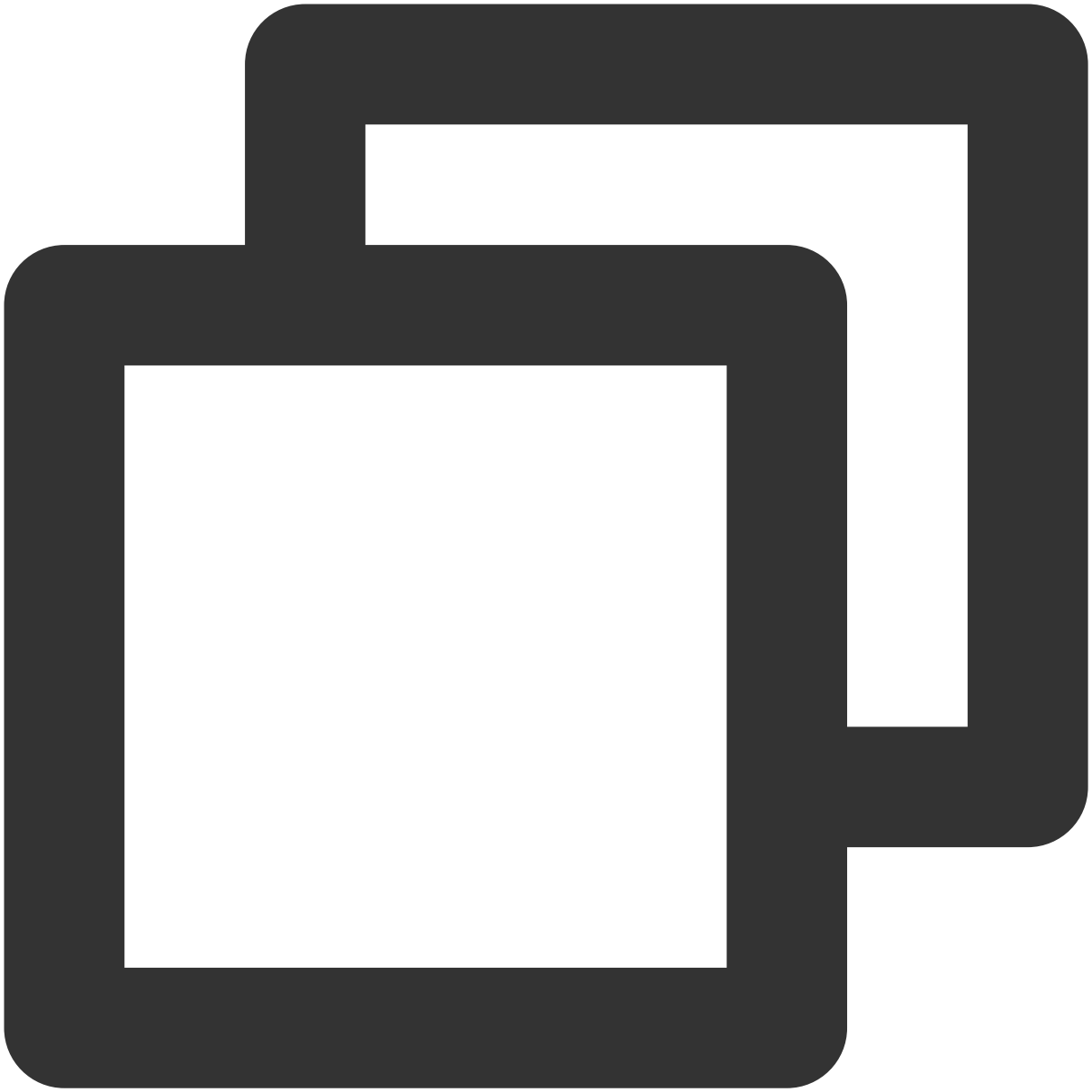
本文档提供关于文本审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
创建任务	用于提交一个文本审核任务
查询任务	用于查询指定的文本审核任务

创建任务

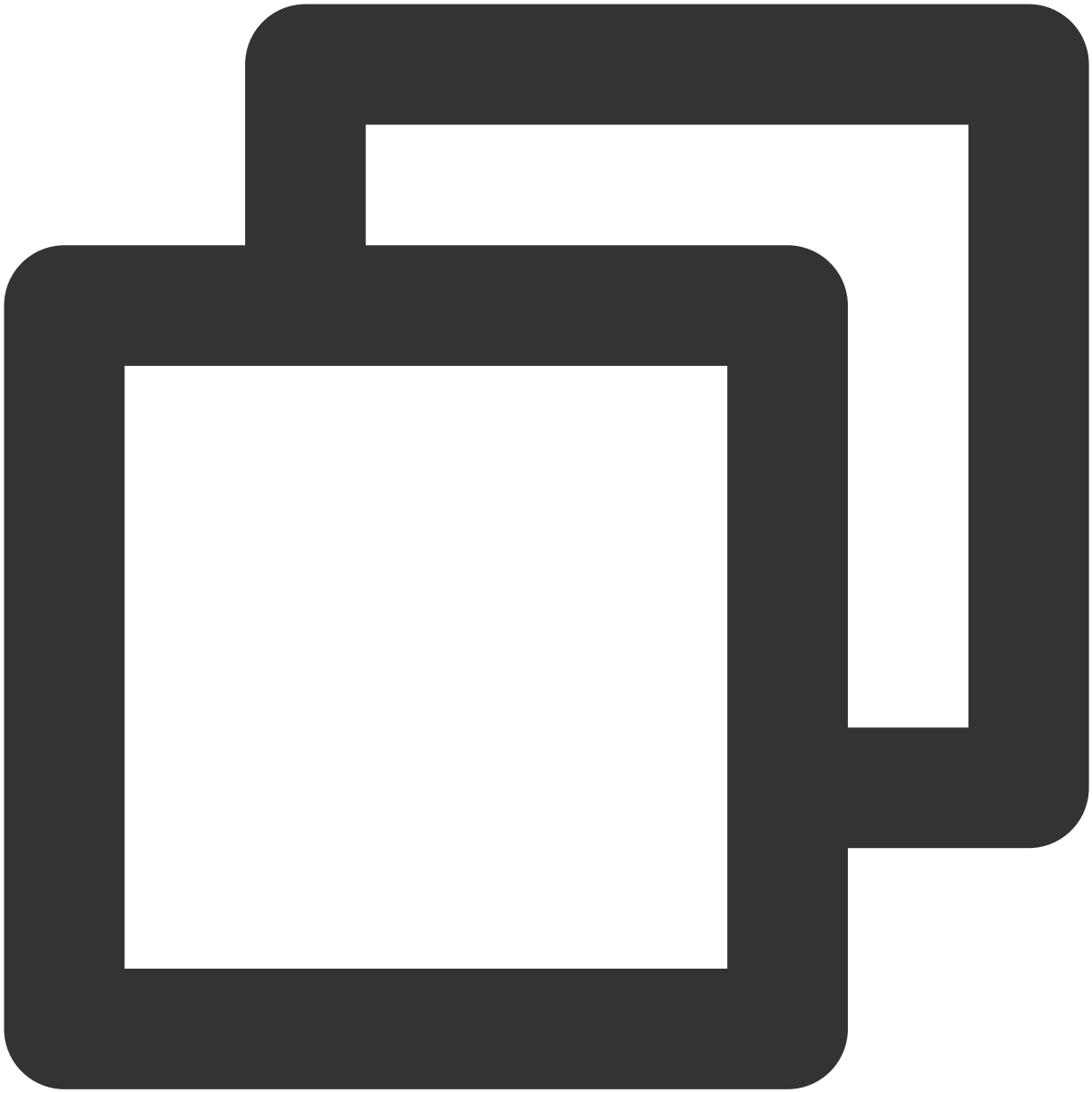
PutTextAuditingJob 接口用来提交一个文本审核任务。当审核文本内容时，响应体直接返回审核结果，当审核文本对象时，返回任务摘要，您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

方法原型



```
func (s *CIService) PutTextAuditingJob(ctx context.Context, opt *PutTextAuditingJob
```

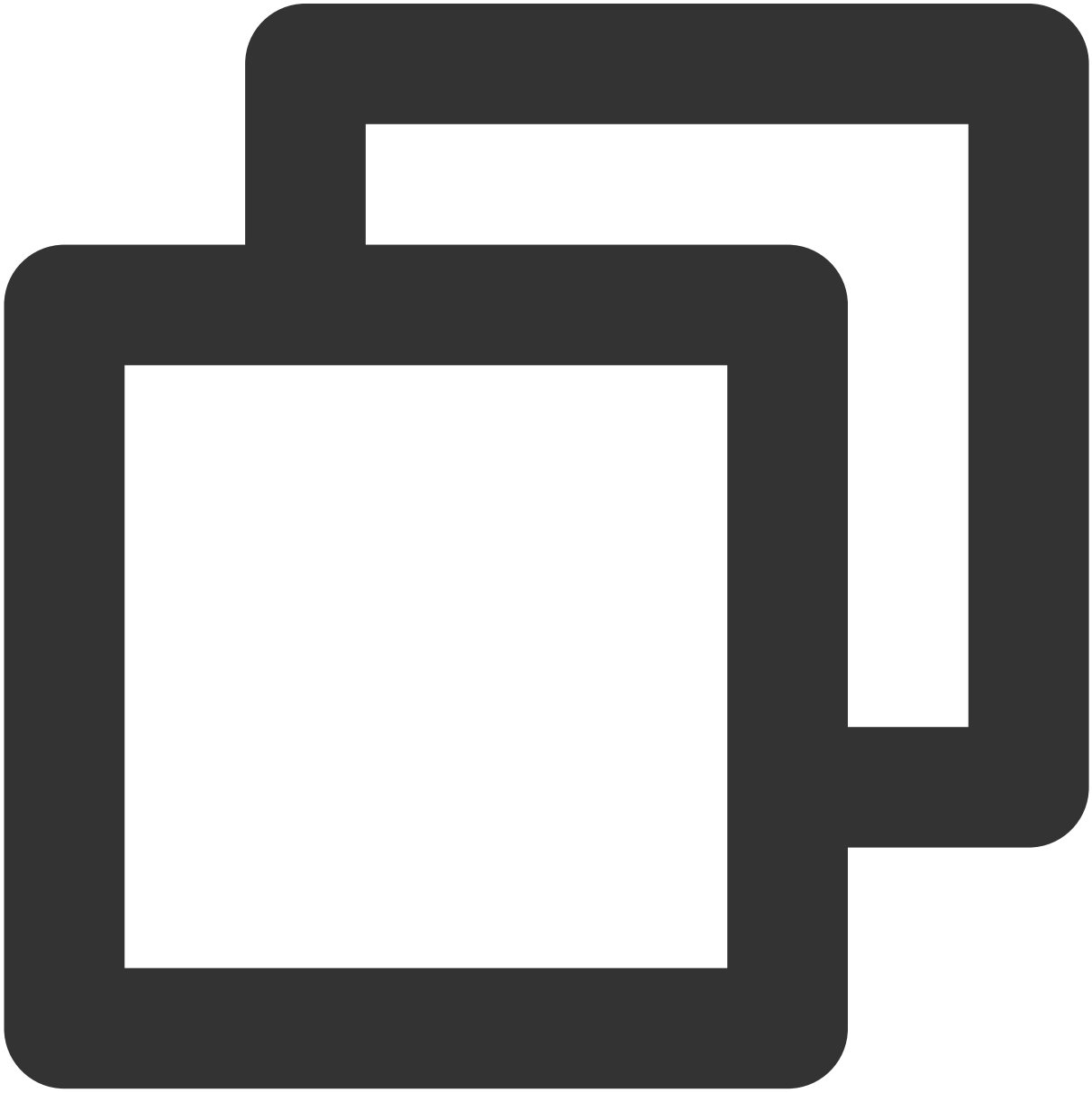
请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
        Transport: &cos.AuthorizationTransport{
            SecretID:  os.Getenv("SECRETID"),
            SecretKey: os.Getenv("SECRETKEY"),
        }
})
```

```
opt := &cos.PutTextAuditingJobOptions{
    InputObject: "test.txt",
    Conf: &cos.TextAuditingJobConf{
        DetectType: "Porn,Ads",
    },
}
res, _, err := c.CI.PutTextAuditingJob(context.Background(), opt)
```

参数说明



```
type PutTextAuditingJobOptions struct {
```

```
    InputObject   string
    InputUrl      string
    InputContent  string
    InputDataId   string
    InputUserInfo *UserExtraInfo
    Conf          *TextAuditingJobConf
}
type UserExtraInfo struct {
    TokenId  string
    Nickname string
    DeviceId string
    AppId    string
    Room     string
    IP       string
    Type     string
}
type TextAuditingJobConf struct {
    DetectType      string
    Callback        string
    CallbackVersion string
    BizType         string
}
```

参数名称	参数描述	类型
InputObject	当前 COS 存储桶中的文本文件名称，例如在目录 test 中的文件 test.txt，则文件名称为 `test/test.txt`，文本文件仅支持UTF8编码和 GBK 编码的内容，且文件大小不得超过1MB。Object， Url 和 Content 只能选择其中一种。	String
InputContent	当传入的内容为纯文本信息，需要先经过 base64 编码，文本编码前的原文长度不能超过10000个 utf8 编码字符。若超出长度限制，接口将会报错。Object, Url 和 Content 只能选择其中一种。	String
InputUrl	文本文件的链接地址，例如 `http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/test.txt` 。Object, Url 和 Content 只能选择其中一种。	String
InputDataId	文本标识，该字段在结果中返回原始内容，长度限制为512字节。	String
InputUserInfo	用户业务字段。	Object
Conf	审核规则配置。	Struct
BizType	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置审核策略。	String
DetectType	审核的场景类型，有效值：Porn（涉黄）、Ads（广告）、Illegal（违法）、	String

	Abuse（谩骂），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	
Callback	审核结果可以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。当Input使用Content时，该参数不生效，结果会直接返回。	String
CallbackVersion	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为Simple。	String

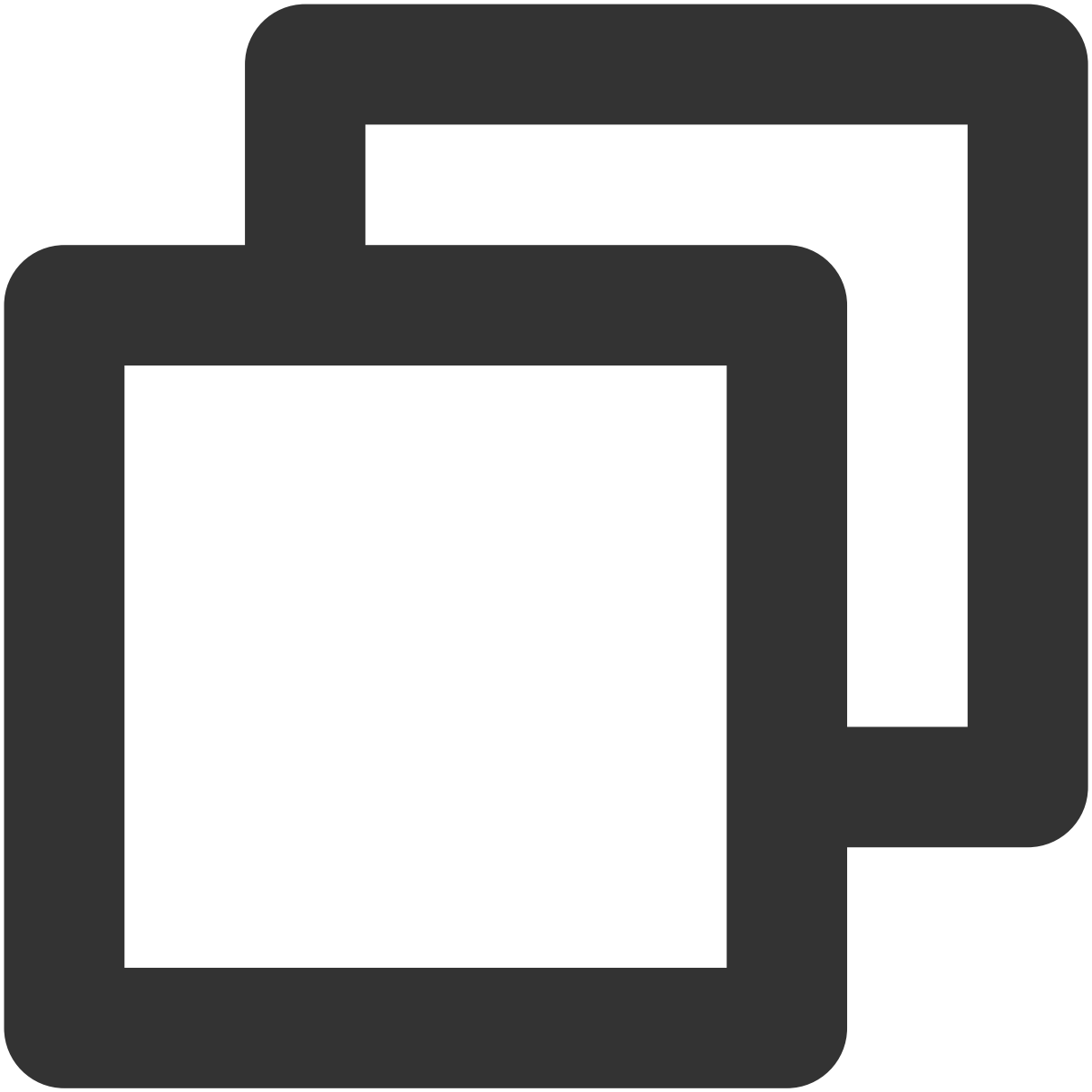
结果说明

调用 PutTextAuditingJob 函数，会解析 api 返回的 xml 内容到 PutTextAuditingJobResult 结构，具体返回参数可查看提交文本审核任务 文档。

查询任务

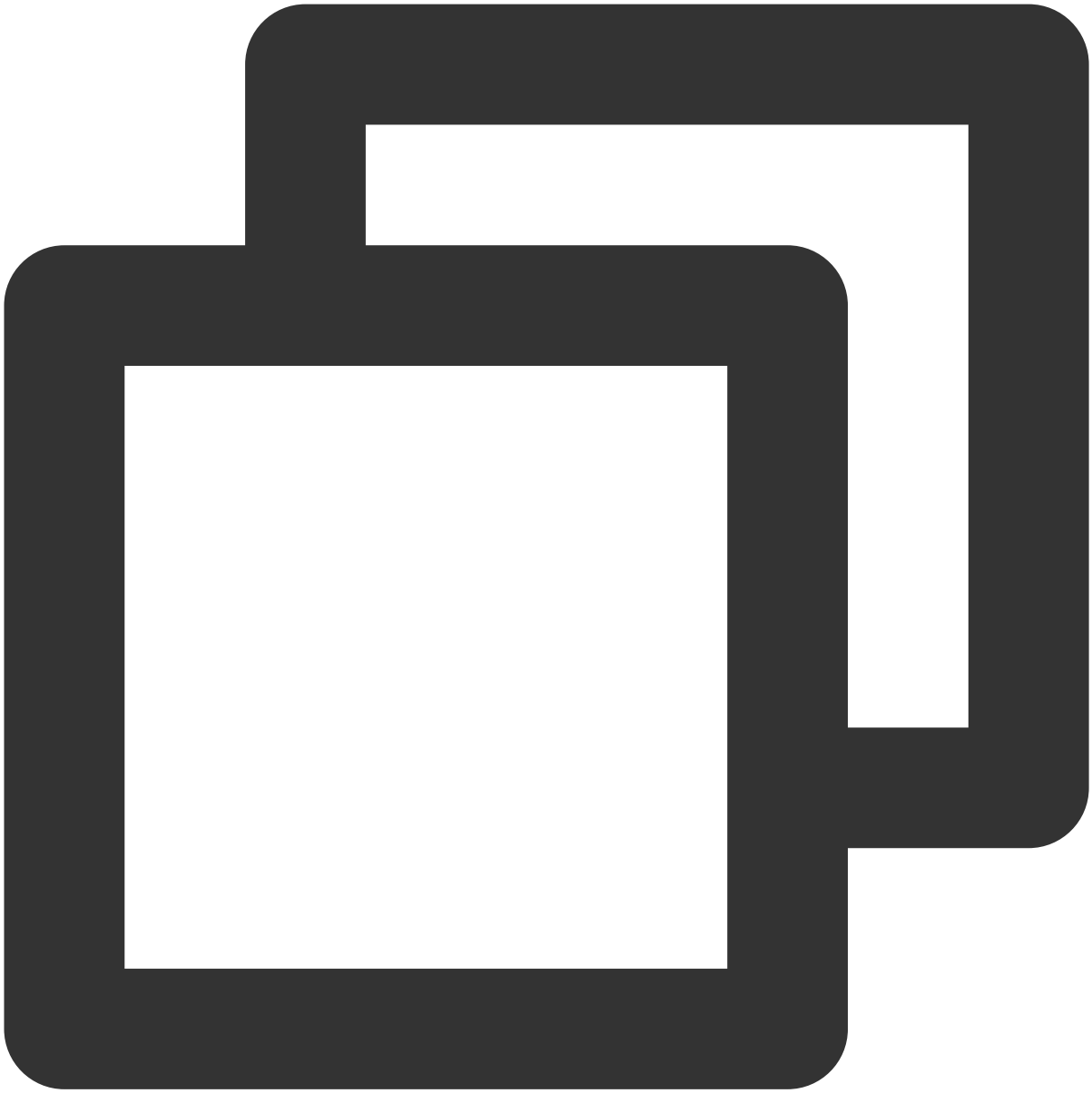
GetTextAuditingJob 接口用来查询指定的文本审核任务。您可以根据文本审核任务的 JobId 来查询文本审核结果。

方法原型



```
func (s *CIService) GetTextAuditingJob(ctx context.Context, jobid string) (*GetText
```

请求示例



```
// 将 examplebucket-1250000000 和 COS_REGION修改为真实的信息
// CI 任务需要提供CIURL
bu, _ := url.Parse("https://examplebucket-1250000000.cos.COS_REGION.myqcloud.com")
cu, _ := url.Parse("https://examplebucket-1250000000.ci.COS_REGION.myqcloud.com")
b := &cos.BaseURL{BucketURL: bu, CIURL: cu}
c := cos.NewClient(b, &http.Client{
    Transport: &cos.AuthorizationTransport{
        SecretID:  os.Getenv("SECRETID"),
        SecretKey: os.Getenv("SECRETKEY"),
    }
})
```

版权所有：腾讯云计算（北京）有限责任公司

```
jobId := "stce25f391a72e11eb99f********"
res, _, err := c.CI.GetTextAuditingJob(context.Background(), jobId)
```

参数说明

参数名称	参数描述	类型
jobId	任务 ID。	String

结果说明

调用 GetTextAuditingJob 函数，会解析 api 返回的 xml 内容到 GetTextAuditingJobResult 结构，具体返回参数可查看 查询文本审核任务结果 文档。

AI 内容识别
图片二维码

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于图片二维码相关的 API 概览以及 SDK 示例代码。

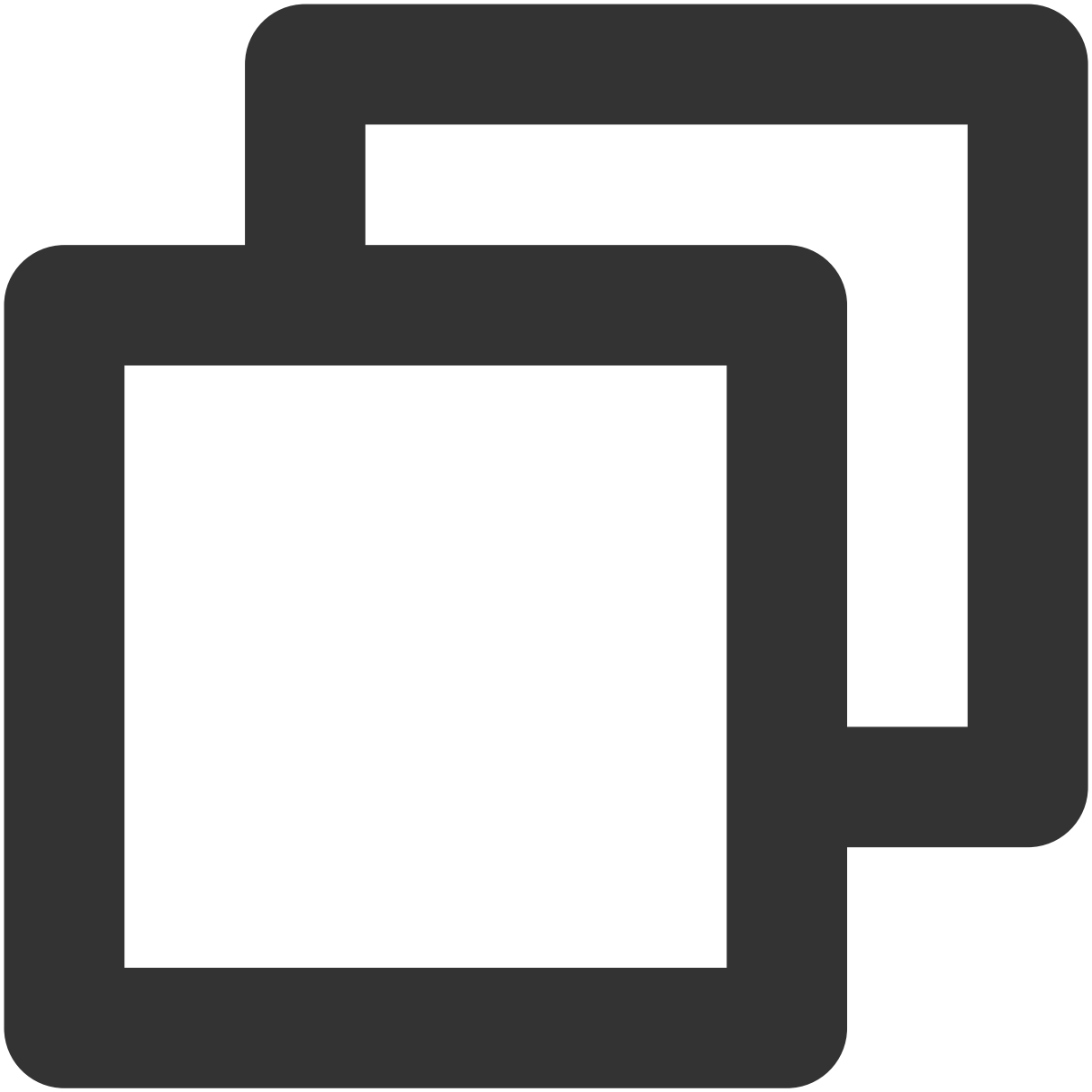
API	说明
二维码识别	二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克
二维码生成	二维码生成功能可根据用户指定的文本信息（URL 或文本），生成对应的二维码或条形码

二维码识别

功能说明

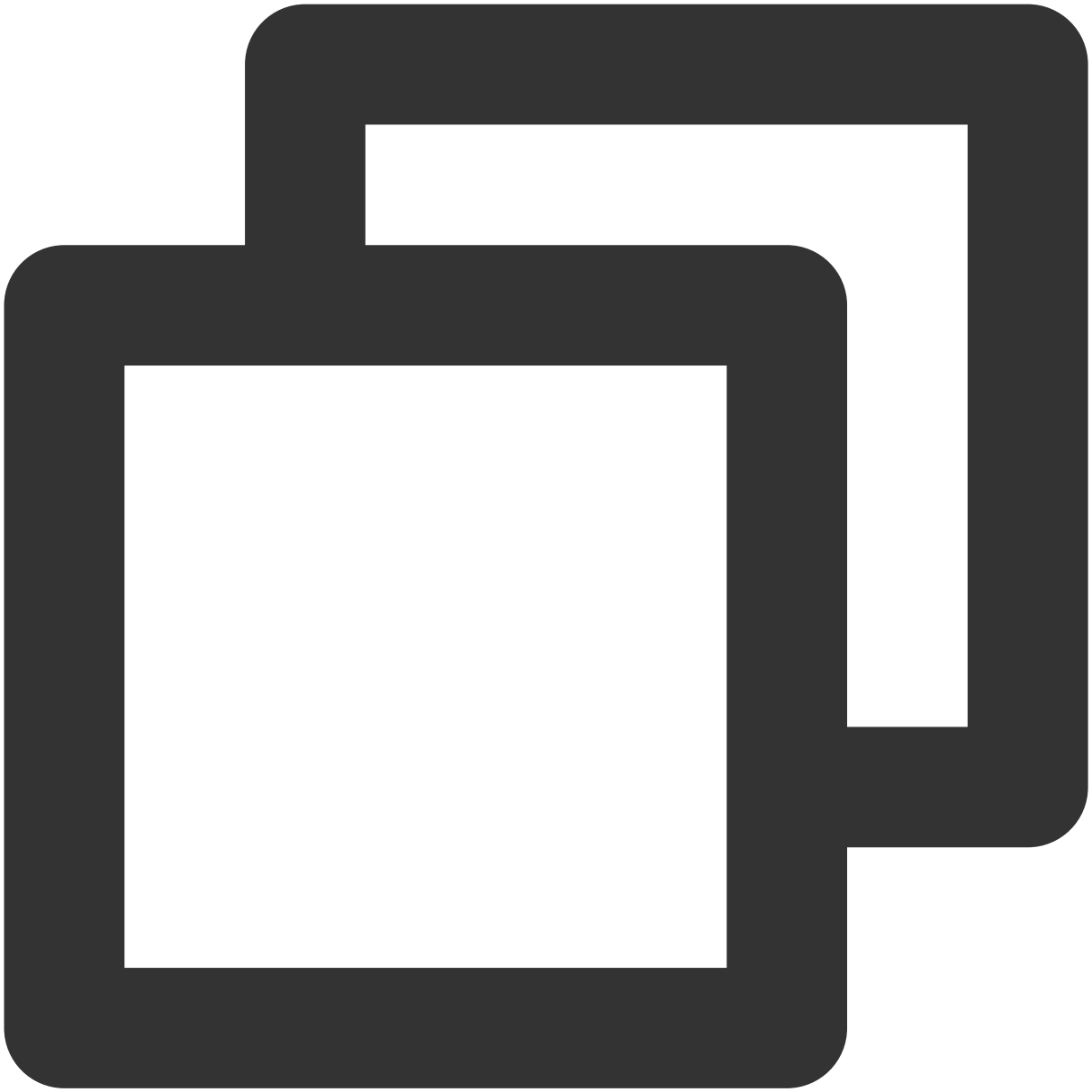
数据万象二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克。

方法原型



```
func (s *CIService) GetQRcode(ctx context.Context, name string, cover int, opt *Obj
```

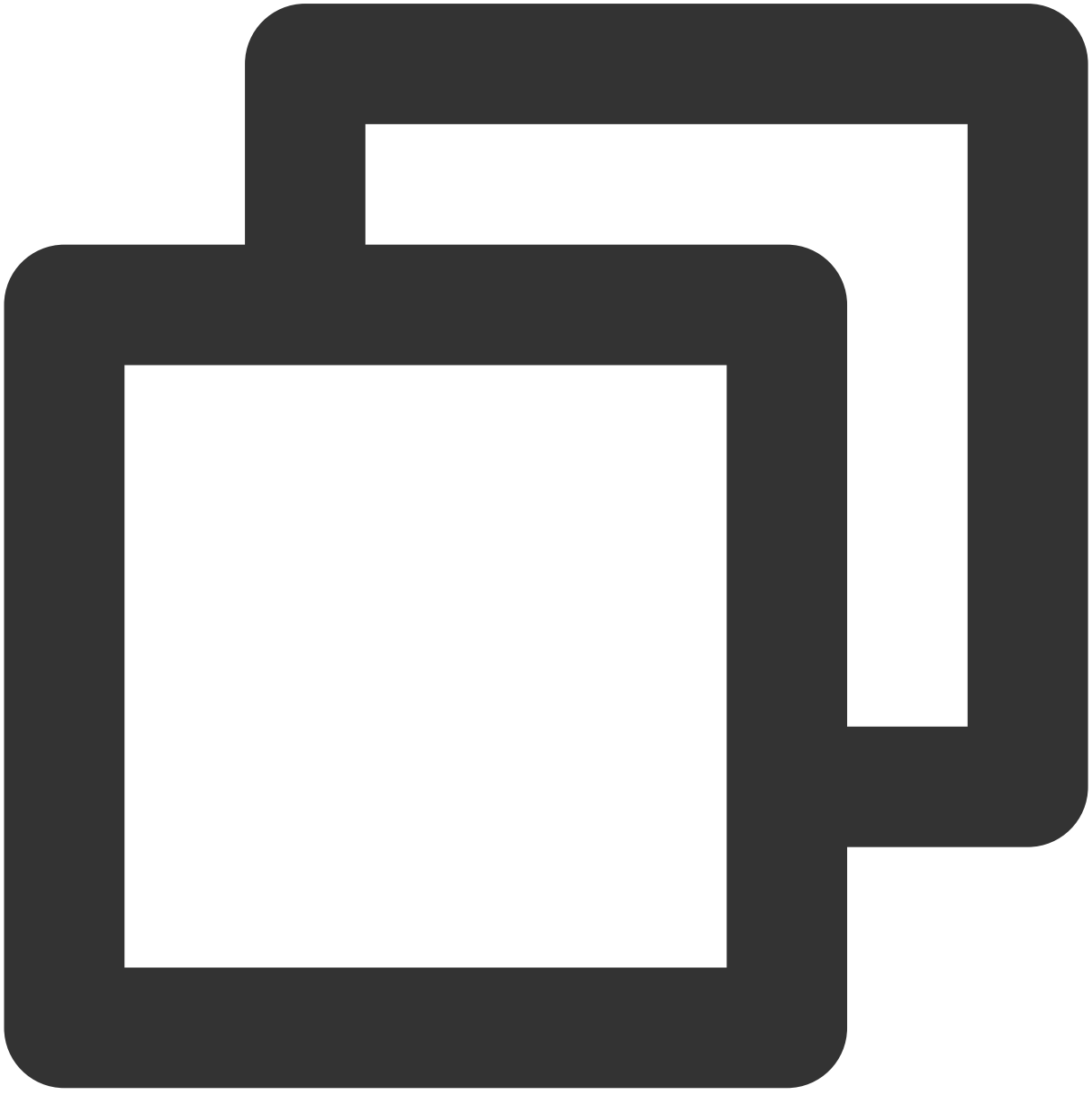
请求示例1：上传时识别



```
opt := &cos.ObjectPutOptions{
    nil,
    &cos.ObjectPutHeaderOptions{
        XOptionHeader: &http.Header{},
    },
}
pic := &cos.PicOperations{
    IsPicInfo: 1,
    Rules: []cos.PicOperationsRules{
        {
            FileId: "format.jpg",
```

```
            Rule:   "QRcode/cover/1",
        },
    },
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
name := "test.jpg"
filepath := "./QRcode.jpg"
res, _, err := c.CI.PutFromFile(context.Background(), name, filepath, opt)
```

请求示例2：下载时识别



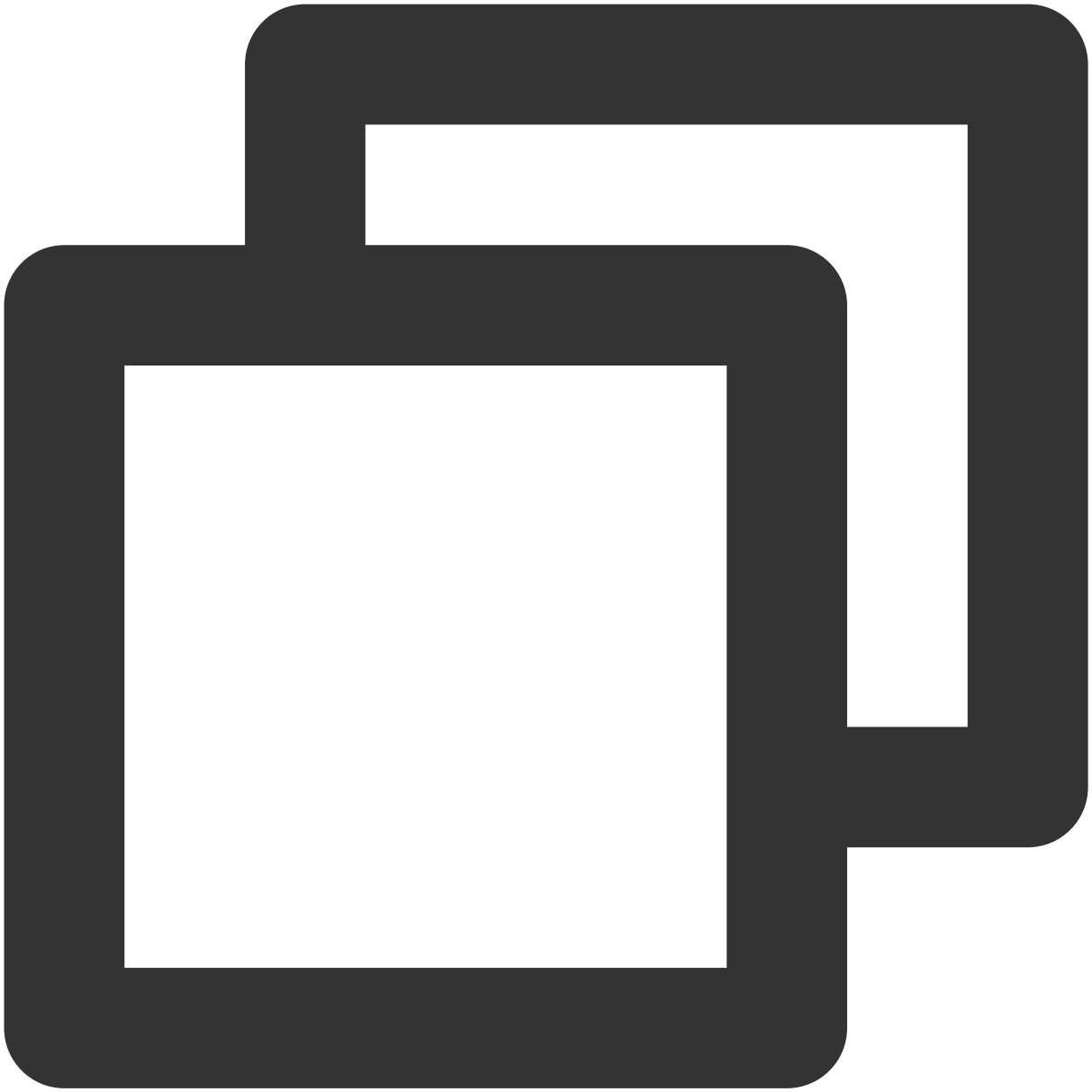
```
name := "test.jpg"
res, _, err := c.CI.GetQRcode(context.Background(), name, 1, nil)
```

参数说明

参数名称	参数描述	类型	必填
cover	二维码覆盖功能，将对识别出的二维码覆盖上马赛克。取值为0或1。0表示不开启二维码覆盖，1表示开启二维码覆盖，默认值0	int	是

结果说明

下载时识别的结果返回如下



```
type GetQRcodeResult struct {
    CodeStatus  int
    QRcodeInfo  *QRcodeInfo
    ResultImage string
}
type QRcodeInfo struct {
    CodeUrl      string
    CodeLocation *CodeLocation
}
type CodeLocation struct {
    Point []string
```

```
}
```

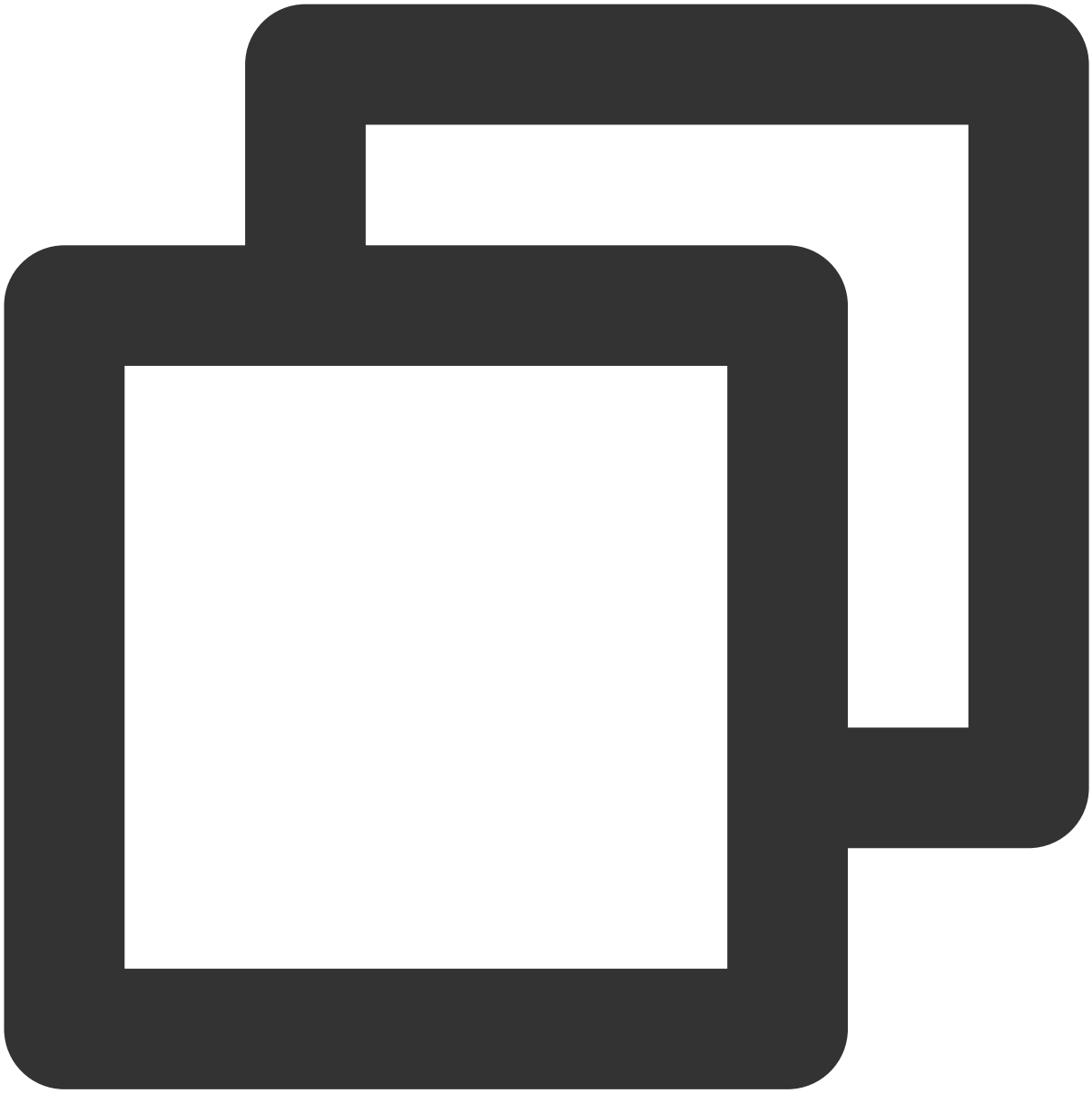
参数名称	参数描述	类型
CodeStatus	二维码识别结果。0表示未识别到二维码，1表示识别到二维码	int
QRcodeInfo	二维码识别结果，可能有多个	Struct
ResultImage	处理后的图片 base64数据，请求参数 cover 为1时返回	string
CodeUrl	二维码的内容。可能识别不到内容	string
CodeLocation	图中识别到的二维码位置坐标	Struct
Point	二维码坐标点（X坐标,Y坐标）	Array

二维码生成

功能说明

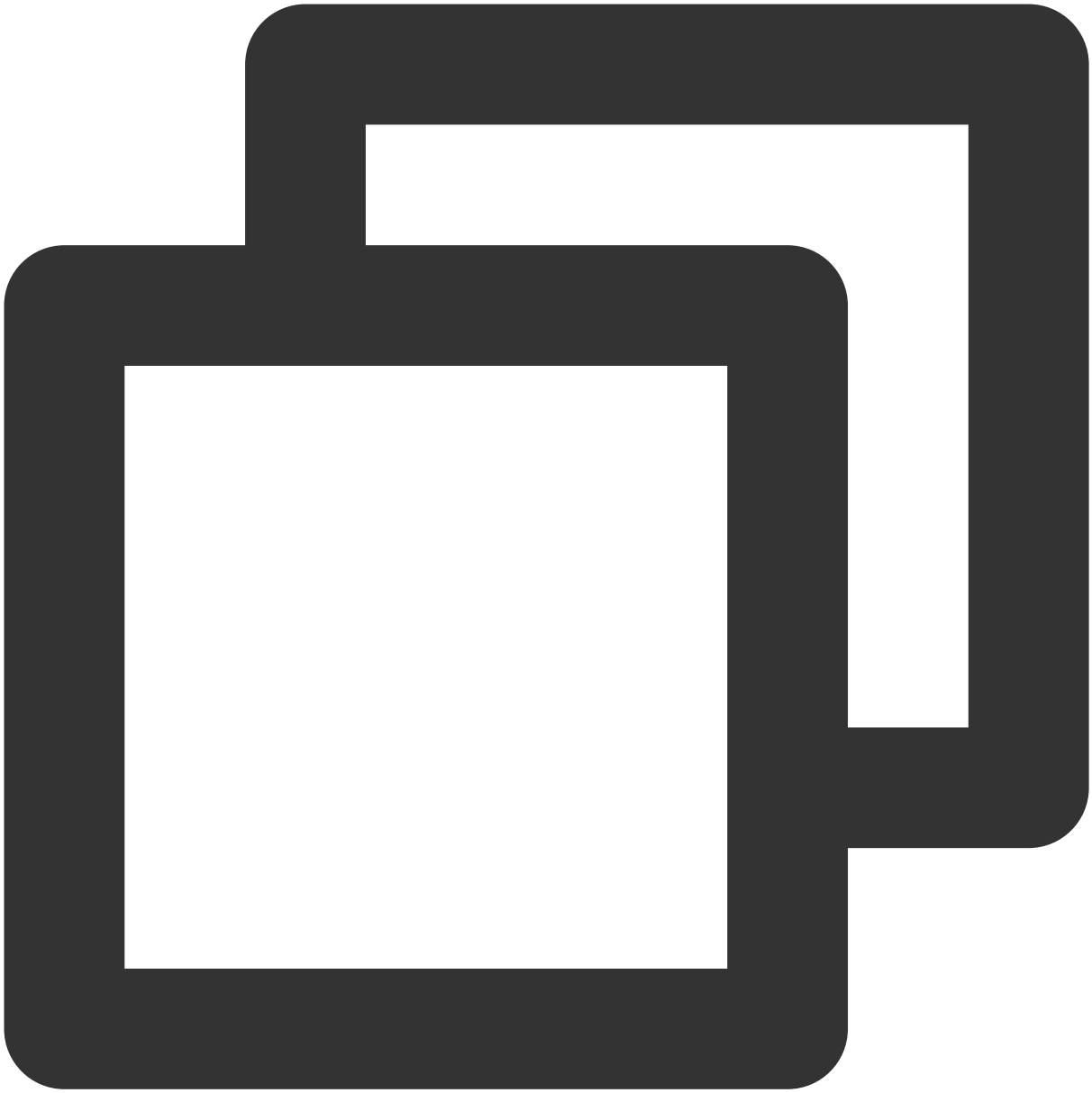
数据万象二维码生成功能可根据用户指定的文本信息（URL 或文本），生成对应的二维码或条形码。

方法原型



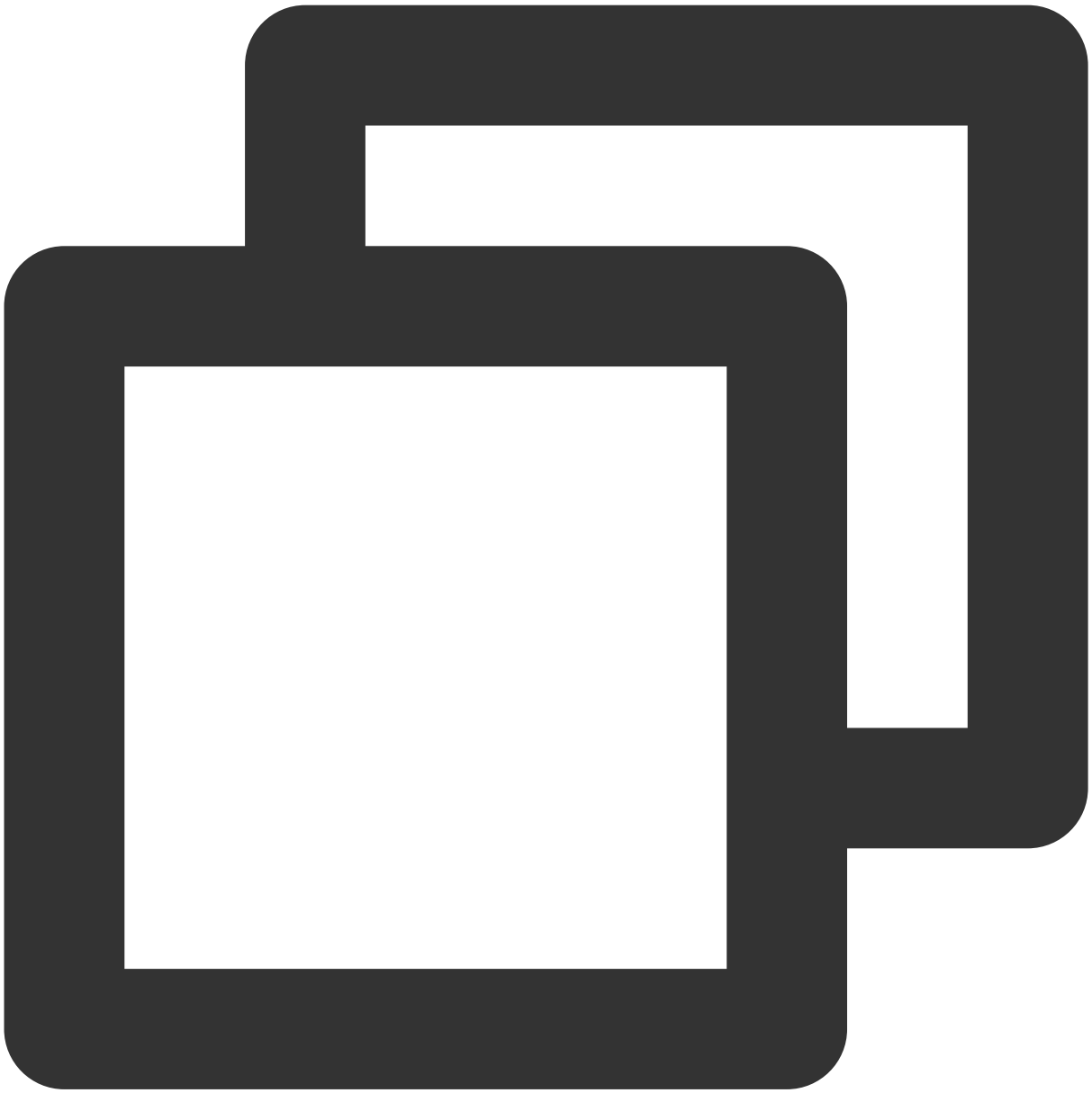
```
func (s *CIService) GenerateQRcode(ctx context.Context, opt *GenerateQRcodeOptions)

func (s *CIService) GenerateQRcodeToFile(ctx context.Context, filePath string, opt
```

请求示例



```
opt := &cos.GenerateQRcodeOptions{
    QRcodeContent: "<https://www.example.com>"
    Mode:          0,
    Width:         200,
}
// Case1 生成二维码
res, _, err := c.CI.GenerateQRcode(context.Background(), opt)

// Case2 生成二维码，并保存成文件
filepath := "example.jpg"
res, _, err := c.CI.GenerateQRcodeToFile(context.Background(), filepath, opt)
```

参数说明

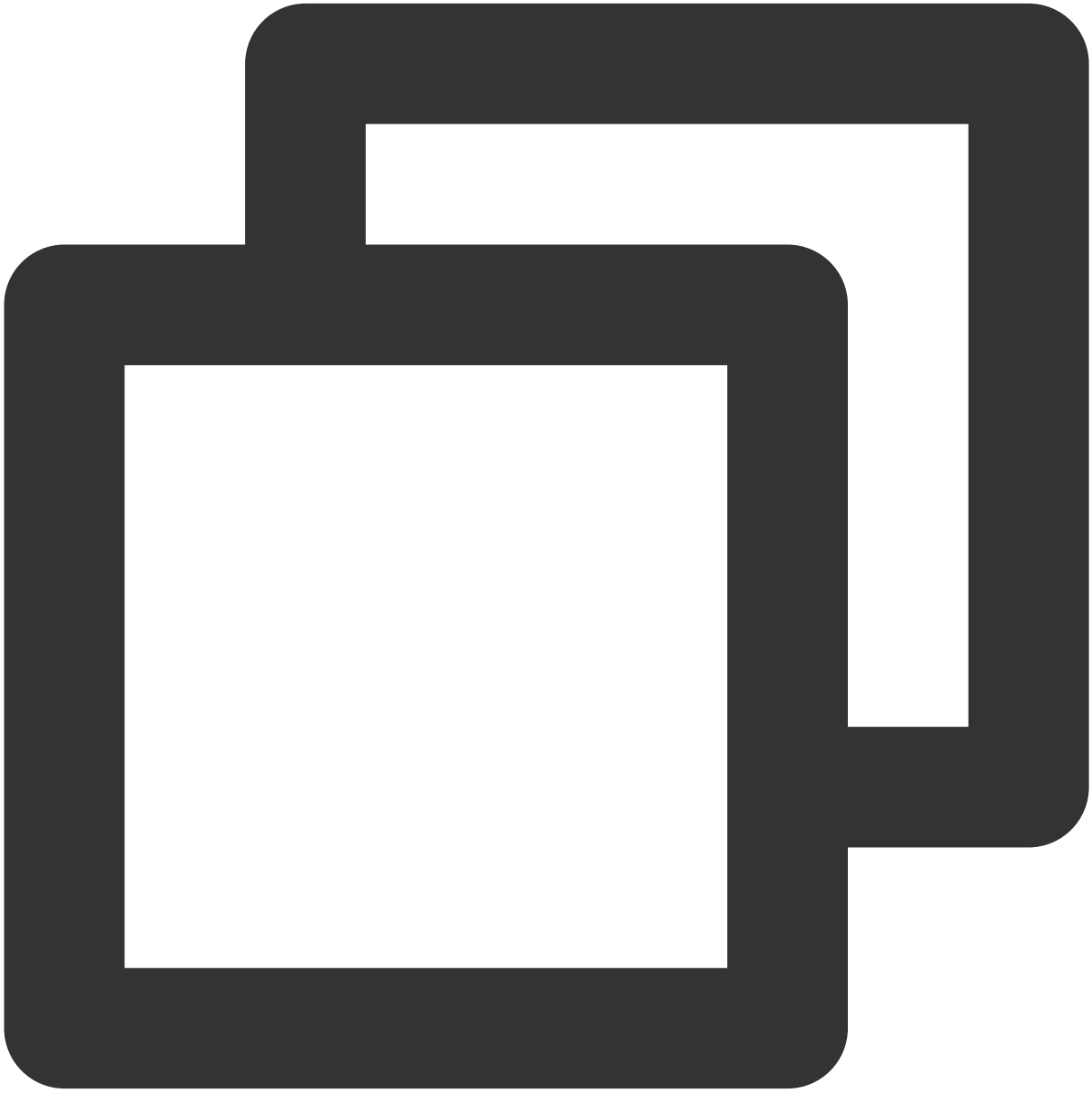


```
type GenerateQRcodeOptions struct {
    QRcodeContent string
    Mode          int
    Width         int
}
```

参数名称	参数描述	类型

QRcodeContent	可识别的二维码文本信息	string
Mode	生成的二维码类型，可选值：0或1。0为二维码，1为条形码，默认值为0	int
Width	指定生成的二维码或条形码的宽度，高度会进行等比压缩	int

结果说明



```
type GenerateQRcodeResult struct {
    ResultImage string
}
```

参数名称	参数描述	类型
ResultImage	二维码图片 base64 数据	string

文件处理
文件解压

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于数据万象文件解压的 API 概览以及 SDK 示例代码。

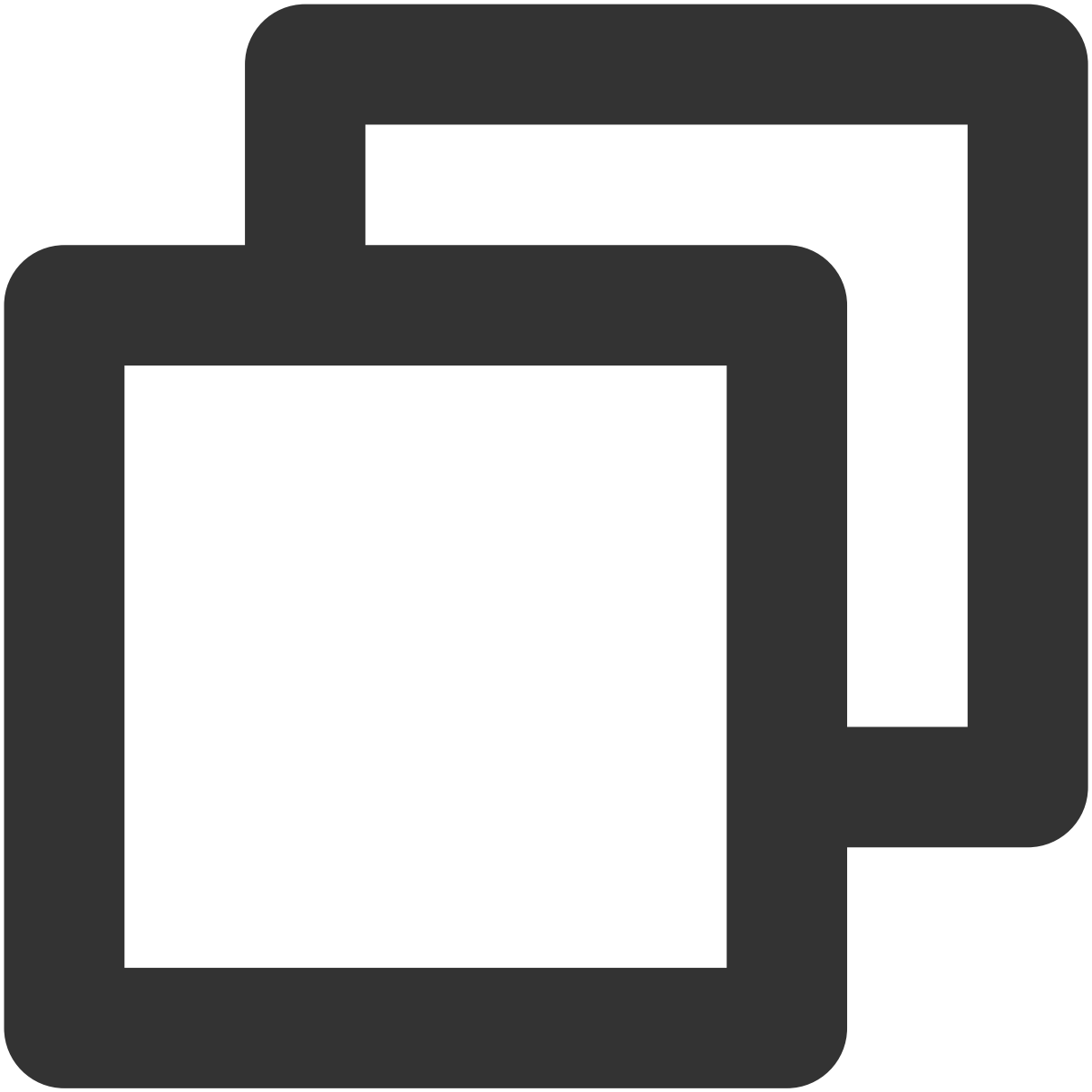
API	操作描述
提交文件解压任务	以提交任务的方式进行文件解压任务
查询文件解压结果	主动查询指定的文件解压任务结果

提交文件解压任务

功能说明

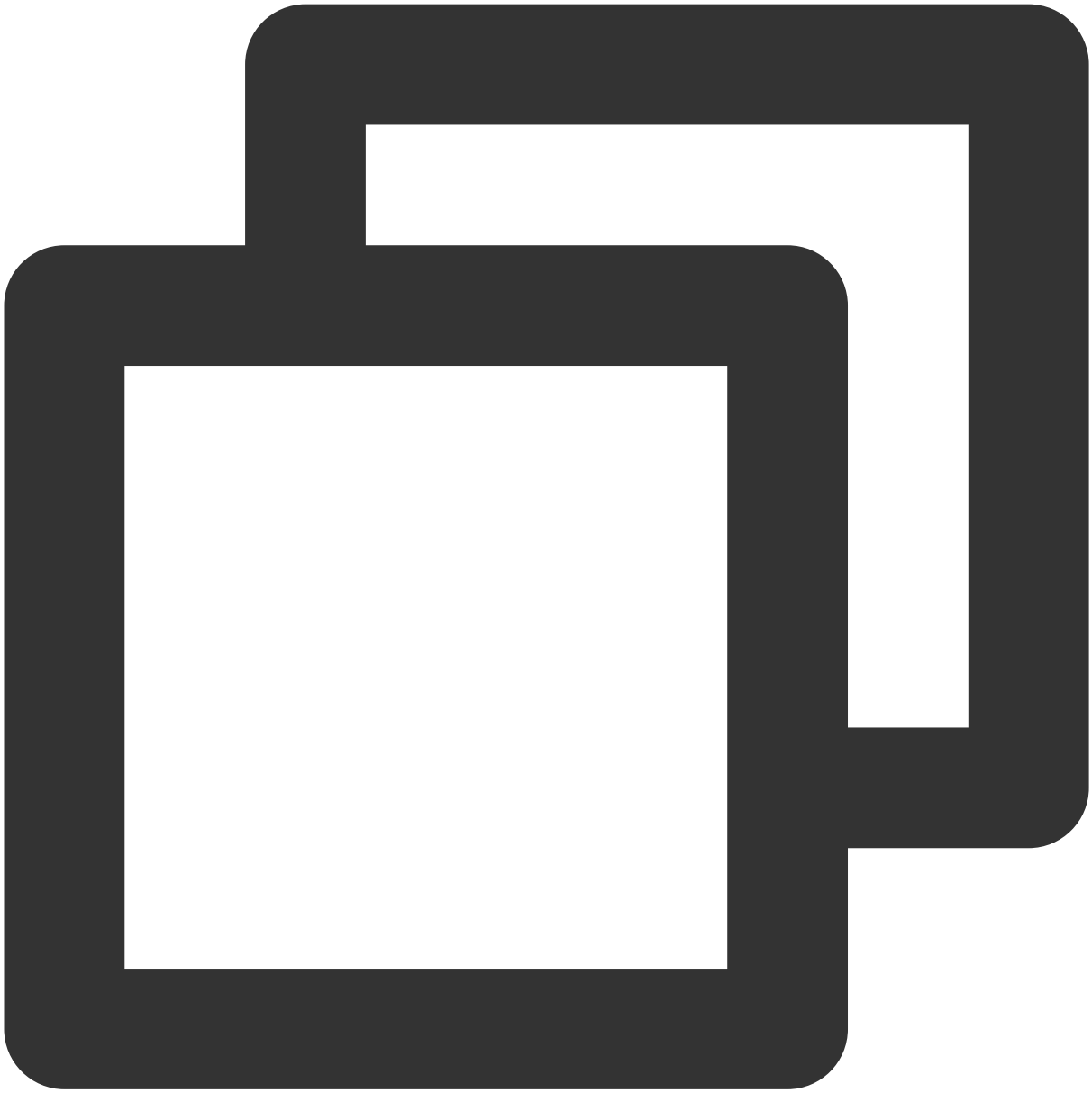
以提交任务的方式进行文件解压缩，异步返回压缩包内被解压出来的文件。

方法原型



```
func (s *CIService) CreateFileProcessJob(ctx context.Context, opt *FileProcessJobOp
```

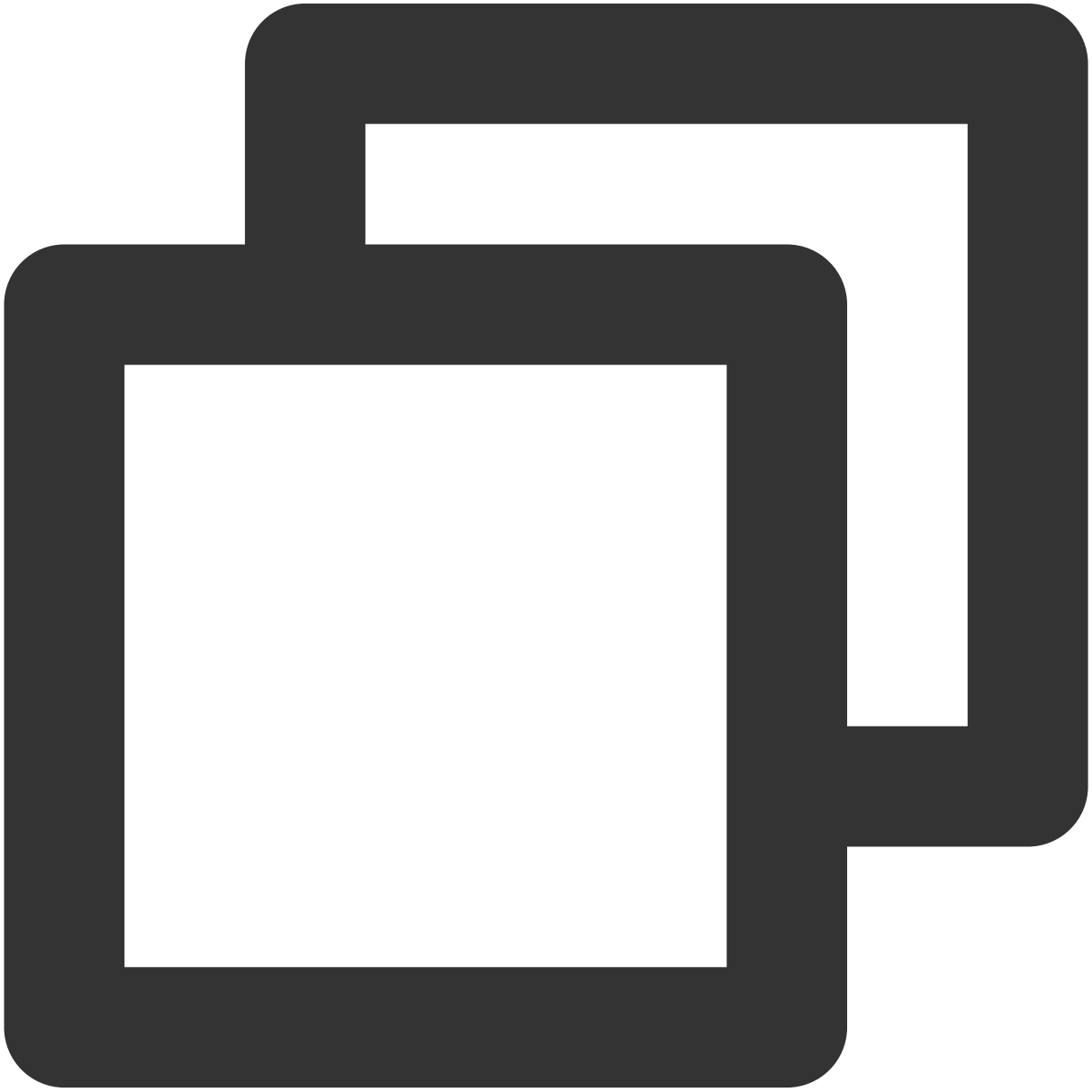
请求示例



```
createJobOpt := &cos.FileProcessJobOptions{
    Tag: "FileUncompress",
    Input: &cos.FileProcessInput{
        Object: "294028.zip",
    },
    Operation: &cos.FileProcessJobOperation{
        FileUncompressConfig: &cos.FileUncompressConfig{
            Prefix:         "uncomp",
            PrefixReplaced: "1",
        },
        Output: &cos.FileProcessOutput{
```

```
            Region: "ap-beijing",
            Bucket: "test-1250000000",
        }
    },
    QueueId: "p111a8dd208104ce3b11c78398f658ca8",
}
res, _, err := c.CI.CreateFileProcessJob(context.Background(), createJobOpt)
```

参数说明



```
type FileProcessJobOptions struct {
```

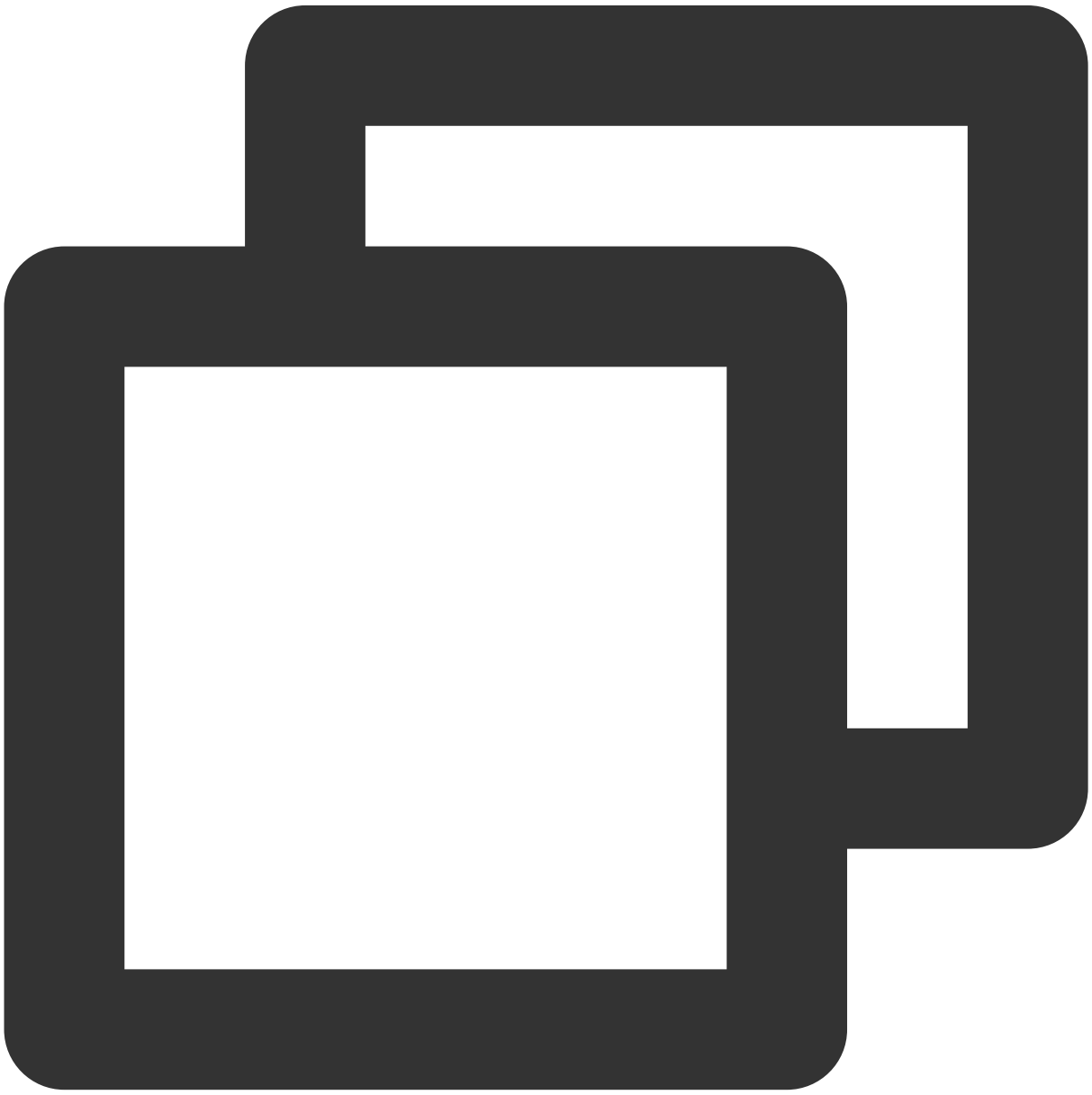
```
    XMLName          xml.Name
    Tag              string
    Input            *FileProcessInput
    Operation        *FileProcessJobOperation
    QueueId          string
    CallBackFormat   string
    CallBackType     string
    CallBack         string
    CallBackMqConfig *NotifyConfigCallBackMqConfig
}
type FileProcessInput struct {
    Region string
    Bucket string
    Object string
}
type FileProcessJobOperation struct {
    FileHashCodeConfig   *FileHashCodeConfig
    FileHashCodeResult   *FileHashCodeResult
    FileUncompressConfig *FileUncompressConfig
    FileUncompressResult *FileUncompressResult
    FileCompressConfig   *FileCompressConfig
    FileCompressResult   *FileCompressResult
    Output               *FileProcessOutput
    UserData             string
}
type FileHashCodeConfig struct {
    Type        string
    AddToHeader bool
}
type FileHashCodeResult struct {
    MD5          string
    SHA1         string
    SHA256       string
    FileSize     int
    LastModified string
    Etag         string
}
type FileUncompressConfig struct {
    Prefix         string
    PrefixReplaced string
}
type FileUncompressResult struct {
    Region    string
    Bucket    string
    FileCount string
}
type FileCompressConfig struct {
```

```
    Flatten string
    Format  string
    UrlList string
    Prefix  string
    Key     string
}
type FileCompressResult struct {
    Region string
    Bucket string
    Object string
}
type FileProcessOutput struct {
    Region string
    Bucket string
    Object string
}
type NotifyConfigCallBackMqConfig struct {
    MqMode   string
    MqRegion string
    MqName   string
}
```

参数名称	描述	类型	是否必选
Tag	创建任务的 Tag，目前仅支持：FileUncompress	String	是
Input	待操作的文件对象。详情见 Input	Container	是
Operation	操作规则	Container	是
Operation.FileUncompressConfig	指定文件解压的处理规则。详情见 FileUncompressConfig	Container	是
Operation.FileUncompressResult	文件解压的结果，任务未完成时不返回。详情见 FileUncompressResult	Container	否
Operation.Output	指定文件处理后的文件保存的地址信息。详情见 Output	Container	是
Operation.UserData	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
QueueId	任务所在的队列 ID	String	是
CallBackFormat	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否

CallBackType	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail FileProcessJobResult
}
type FileProcessJobResult struct {
    Code          string
    Message       string
    JobId         string
    Tag           string
    State         string
    CreationTime  string
    StartTime     string
    EndTime       string
    QueueId       string
    Input         *FileProcessInput
    Operation     *FileProcessJobOperation
}
```

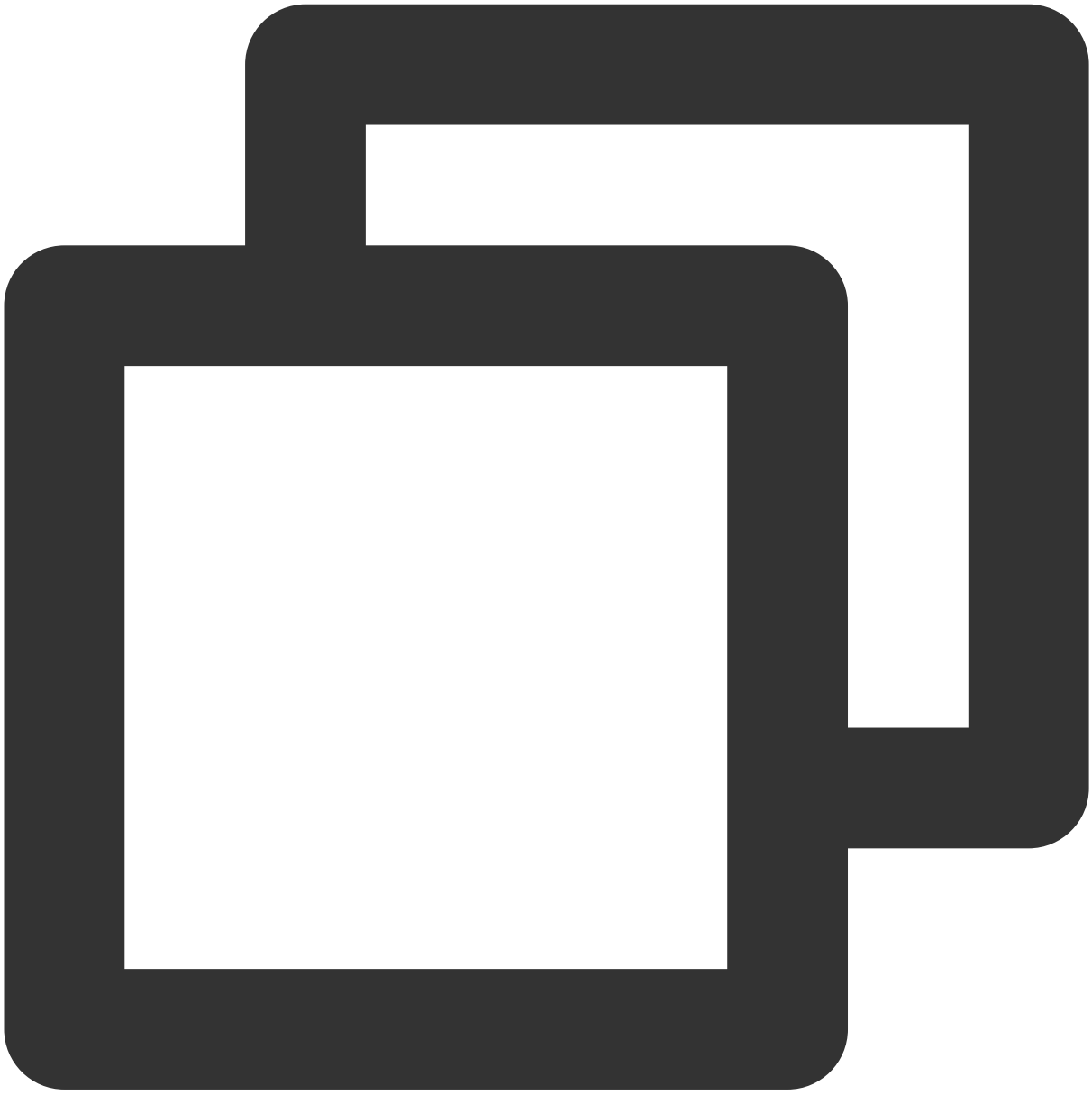
参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：FileUncompress	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String
CreationTime	任务的创建时间	String
StartTime	任务的开始时间	String
EndTime	任务的结束时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径，详情请参见 FileProcessJobOptions.Input	Container
Operation	该任务的规则，详情请参见 FileProcessJobOptions.Operation	Container

查询文件解压结果

功能说明

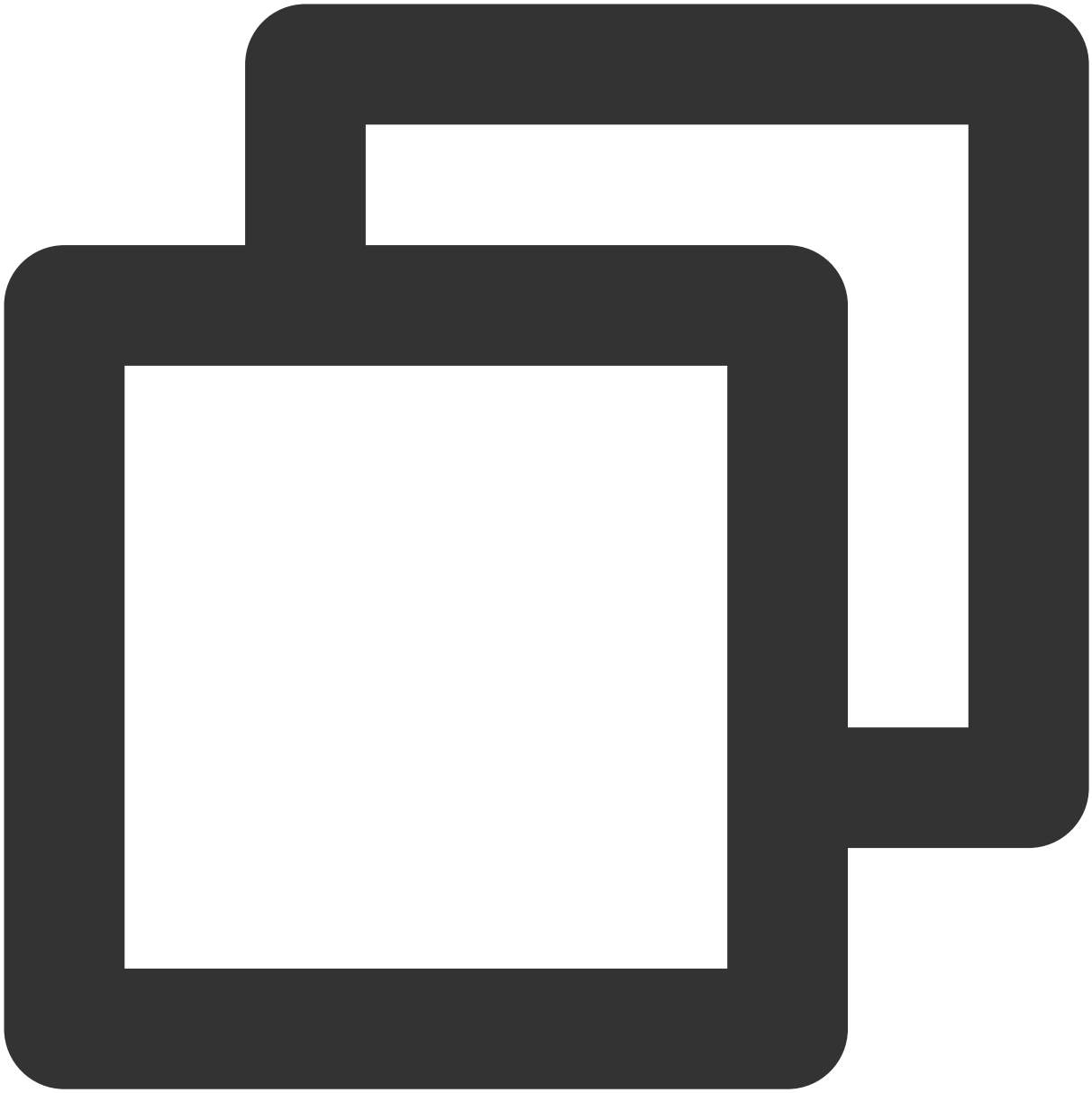
查询一个文件处理任务,根据任务 ID 查询任务详情。

方法原型



```
func (s *CIService) DescribeFileProcessJob(ctx context.Context, jobid string) (*Fil
```

请求示例

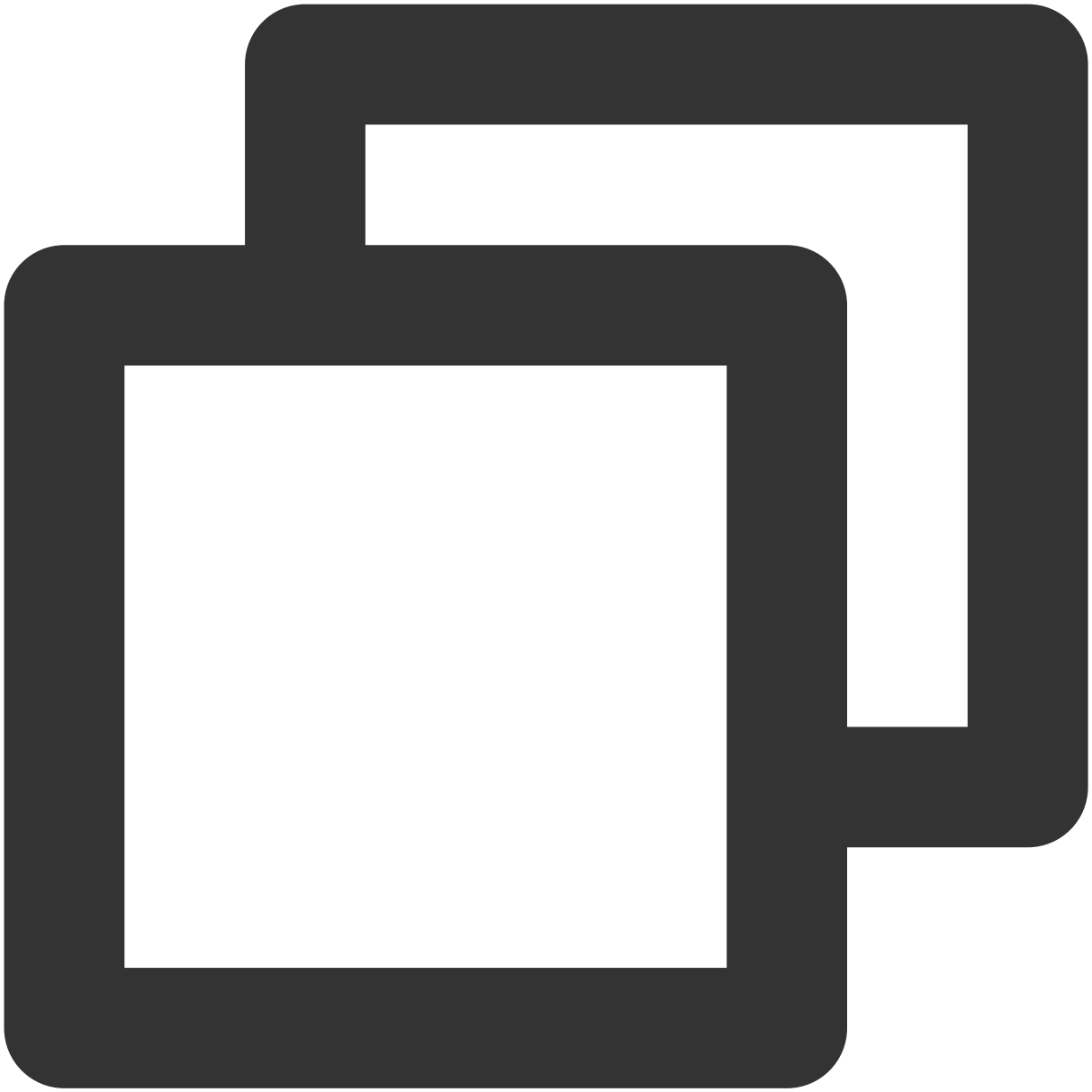


```
jobid := "<jobid>"
res, _, err := c.CI.DescribeFileProcessJob(context.Background(), jobid)
```

参数说明

参数名称	描述	类型	是否必选
jobid	文件处理任务 ID	String	是

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail *FileProcessJobsDetail
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息， 同 CreateFileProcessJob 接口的 Response.JobsDetail 节点	Container

文件压缩

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于数据万象文件压缩的 API 概览以及 SDK 示例代码。

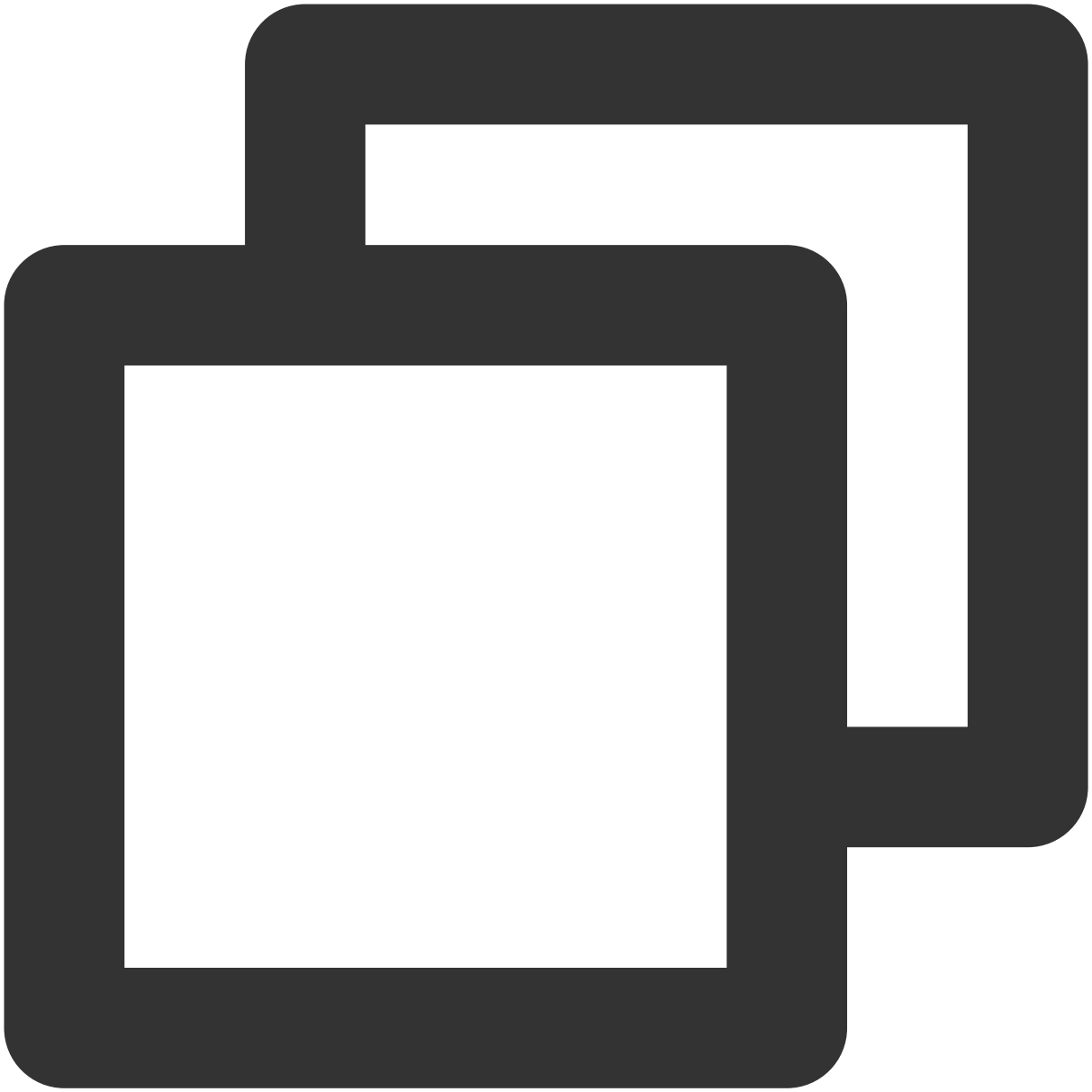
API	操作描述
提交多文件打包压缩任务	以提交任务的方式进行多文件打包压缩任务
查询多文件打包压缩结果	主动查询指定的多文件打包压缩任务结果

提交多文件打包压缩任务

功能说明

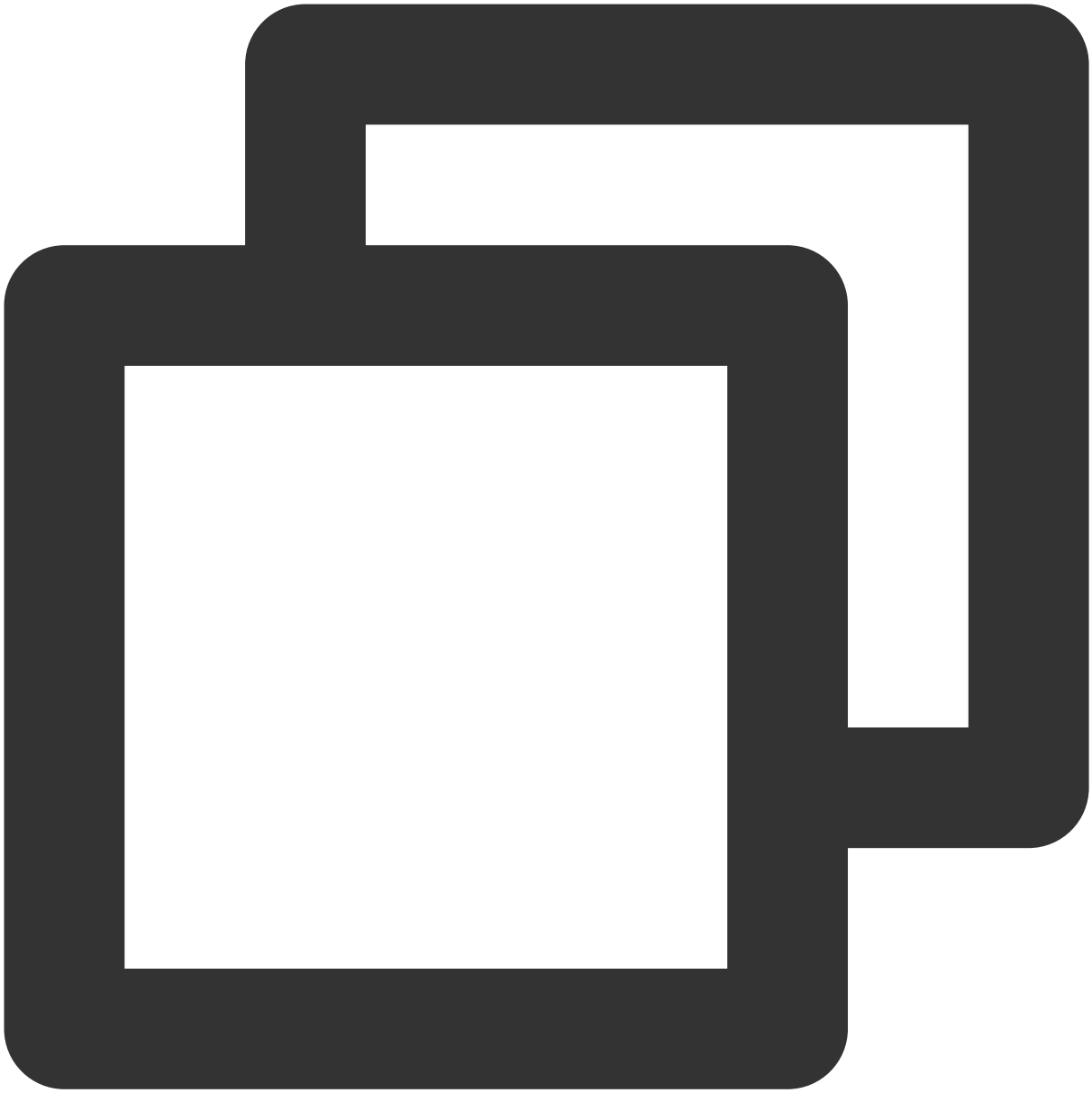
多文件打包压缩功能可以将您的多个文件，打包为 zip 等压缩包格式，以提交任务的方式进行多文件打包压缩，异步返回打包后的文件。

方法原型



```
func (s *CIService) CreateFileProcessJob(ctx context.Context, opt *FileProcessJobOp
```

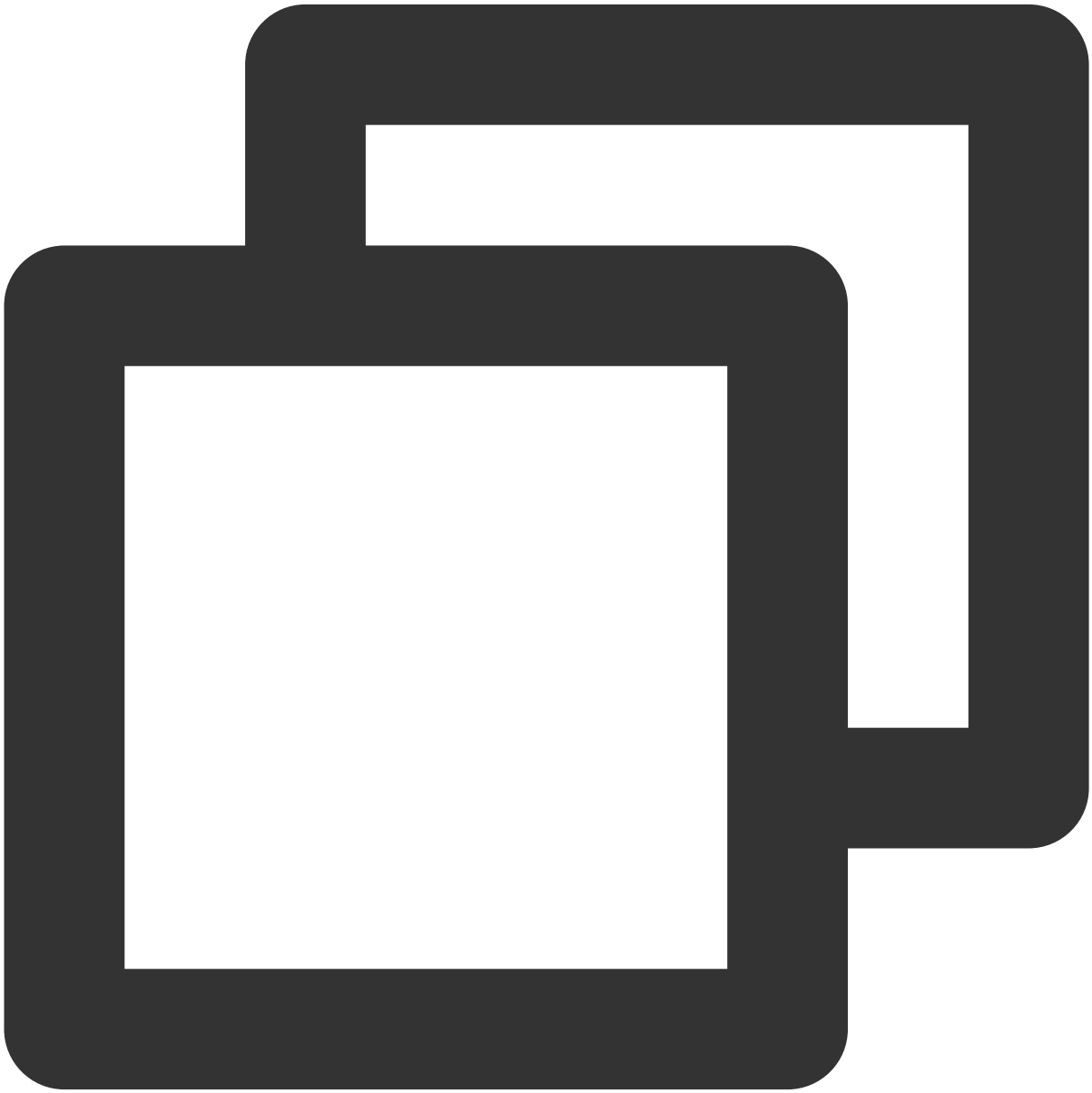
请求示例



```
createJobOpt := &cos.FileProcessJobOptions{
    Tag: "FileCompress",
    Operation: &cos.FileProcessJobOperation{
        FileCompressConfig: &cos.FileCompressConfig{
            Format:  "zip",
            Flatten: "0",
            Key:     []string{"1.mp3", "2.jpg"},
        },
        Output: &cos.FileProcessOutput{
            Region: "ap-shanghai",
            Bucket: "test-1250000000",
```

```
            Object: "vvvvxxxzz.zip",
        },
    },
    QueueId: "p111a8dd208104ce3b11c78398f658ca8",
}
res, _, err := c.CI.CreateFileProcessJob(context.Background(), createJobOpt)
```

参数说明



```
type FileProcessJobOptions struct {
    XMLName           xml.Name
```

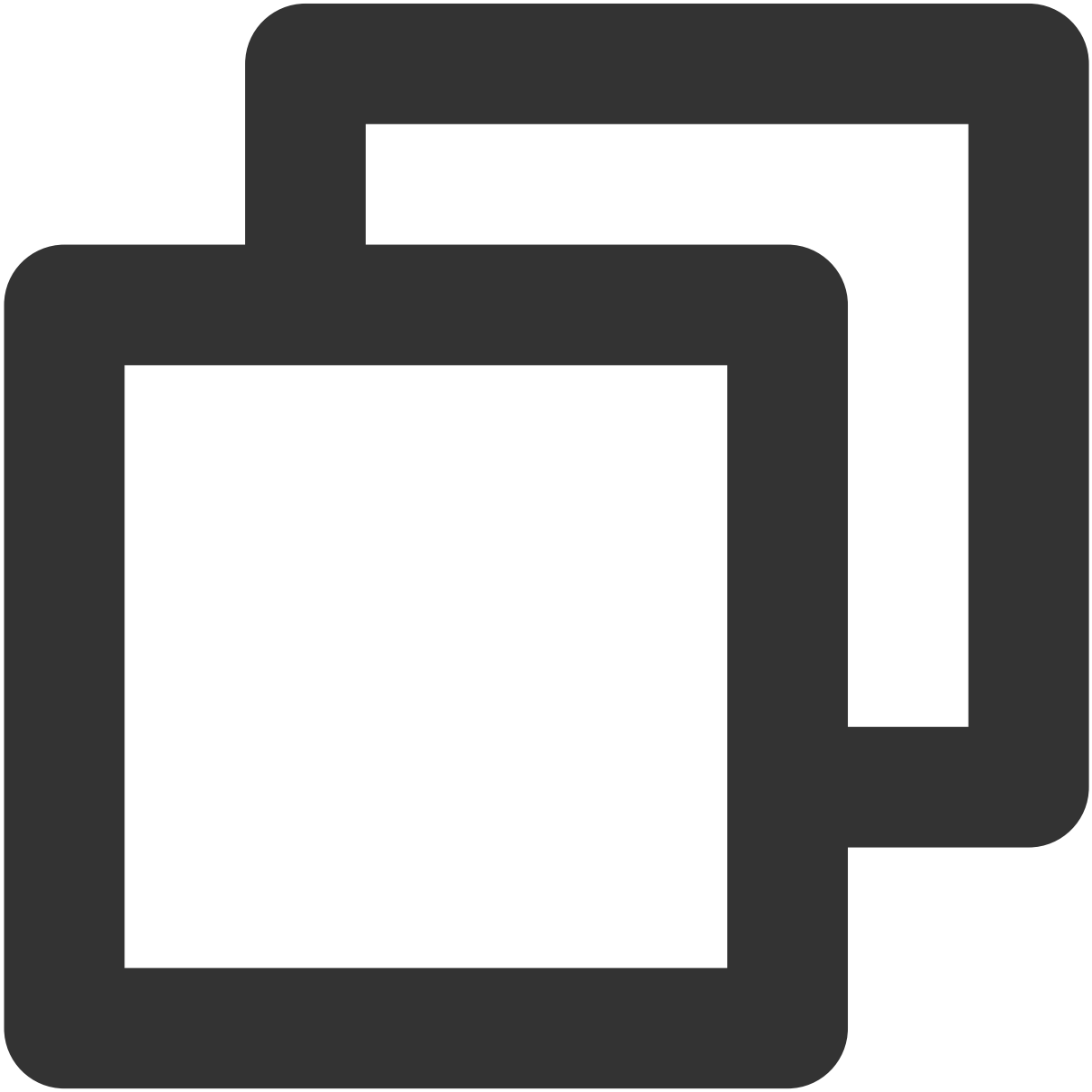
```
    Tag              string
    Input            *FileProcessInput
    Operation        *FileProcessJobOperation
    QueueId          string
    CallBackFormat   string
    CallBackType     string
    CallBack         string
    CallBackMqConfig *NotifyConfigCallBackMqConfig
}
type FileProcessInput struct {
    Region string
    Bucket string
    Object string
}
type FileProcessJobOperation struct {
    FileHashCodeConfig   *FileHashCodeConfig
    FileHashCodeResult   *FileHashCodeResult
    FileUncompressConfig *FileUncompressConfig
    FileUncompressResult *FileUncompressResult
    FileCompressConfig   *FileCompressConfig
    FileCompressResult   *FileCompressResult
    Output               *FileProcessOutput
    UserData             string
}
type FileHashCodeConfig struct {
    Type        string
    AddToHeader bool
}
type FileHashCodeResult struct {
    MD5          string
    SHA1         string
    SHA256       string
    FileSize     int
    LastModified string
    Etag         string
}
type FileUncompressConfig struct {
    Prefix         string
    PrefixReplaced string
}
type FileUncompressResult struct {
    Region    string
    Bucket    string
    FileCount string
}
type FileCompressConfig struct {
    Flatten string
```

```
    Format  string
    UrlList string
    Prefix  string
    Key     []string
}
type FileCompressResult struct {
    Region string
    Bucket string
    Object string
}
type FileProcessOutput struct {
    Region string
    Bucket string
    Object string
}
type NotifyConfigCallBackMqConfig struct {
    MqMode   string
    MqRegion string
    MqName   string
}
```

参数名称	描述	类型	是否必选
Tag	创建任务的 Tag，目前仅支持：FileCompress	String	是
Input	待操作的文件对象。详情见 Input	Container	是
Operation	操作规则	Container	是
Operation.FileCompressConfig	指定文件打包压缩的处理规则。详情见 FileCompressConfig	Container	是
Operation.FileCompressResult	多文件打包压缩的结果，任务未完成时不返回。详情见 FileCompressResult	Container	否
Operation.Output	指定文件处理后的文件保存的地址信息。详情见 Output	Container	是
Operation.UserData	透传用户信息，可打印的 ASCII 码，长度不超过1024	String	否
QueueId	任务所在的队列 ID	String	是
CallBackFormat	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队	String	否

	列的回调类型。		
CallBack	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail FileProcessJobResult
```

```
}
type FileProcessJobResult struct {
    Code          string
    Message       string
    JobId         string
    Tag           string
    State         string
    CreationTime  string
    StartTime     string
    EndTime       string
    QueueId       string
    Input         *FileProcessInput
    Operation     *FileProcessJobOperation
}
```

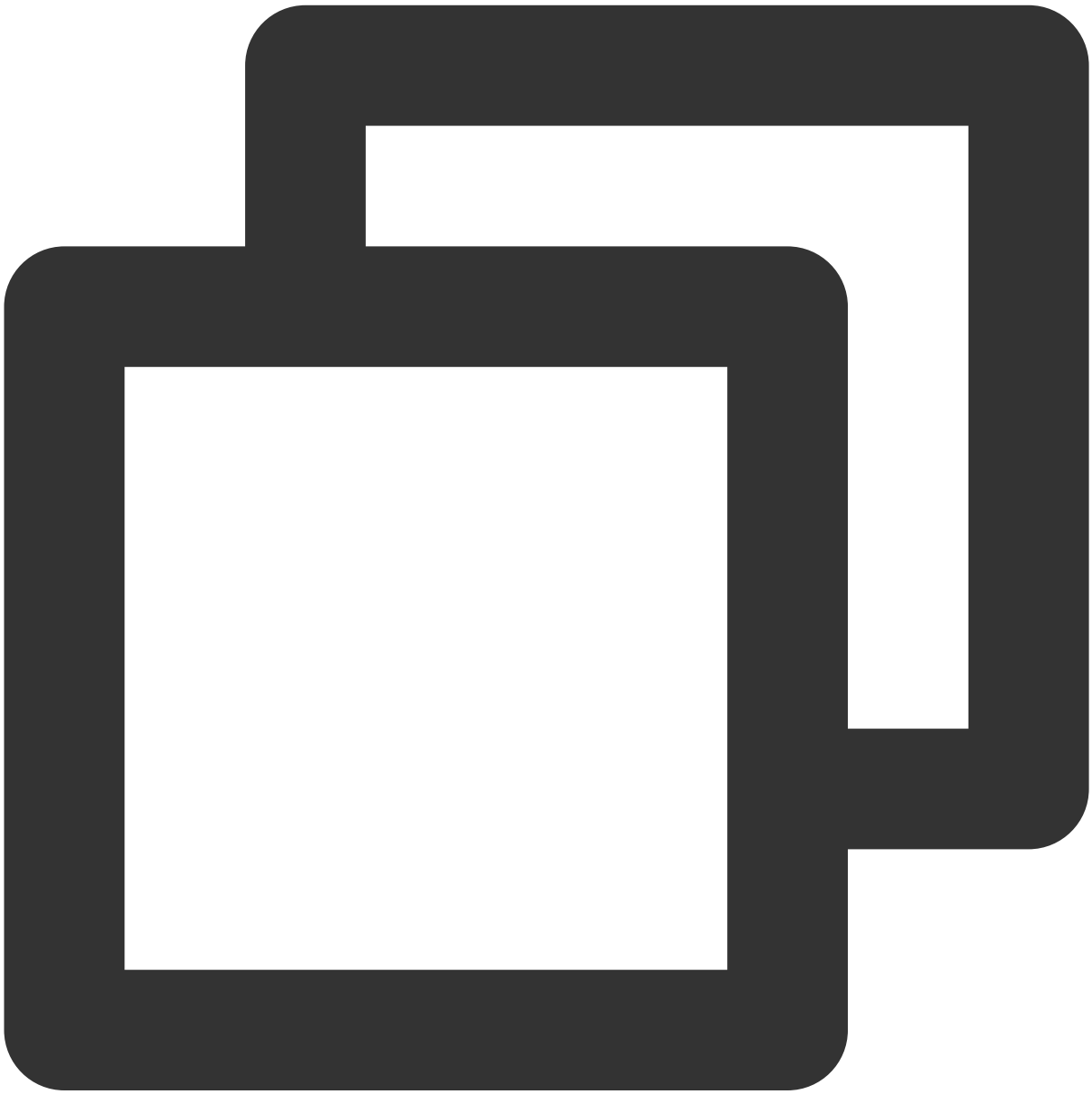
参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：FileCompress	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String
CreationTime	任务的创建时间	String
StartTime	任务的开始时间	String
EndTime	任务的结束时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径，详情请参见 FileProcessJobOptions.Input	Container
Operation	该任务的规则，详情请参见 FileProcessJobOptions.Operation	Container

查询多文件打包压缩结果

功能说明

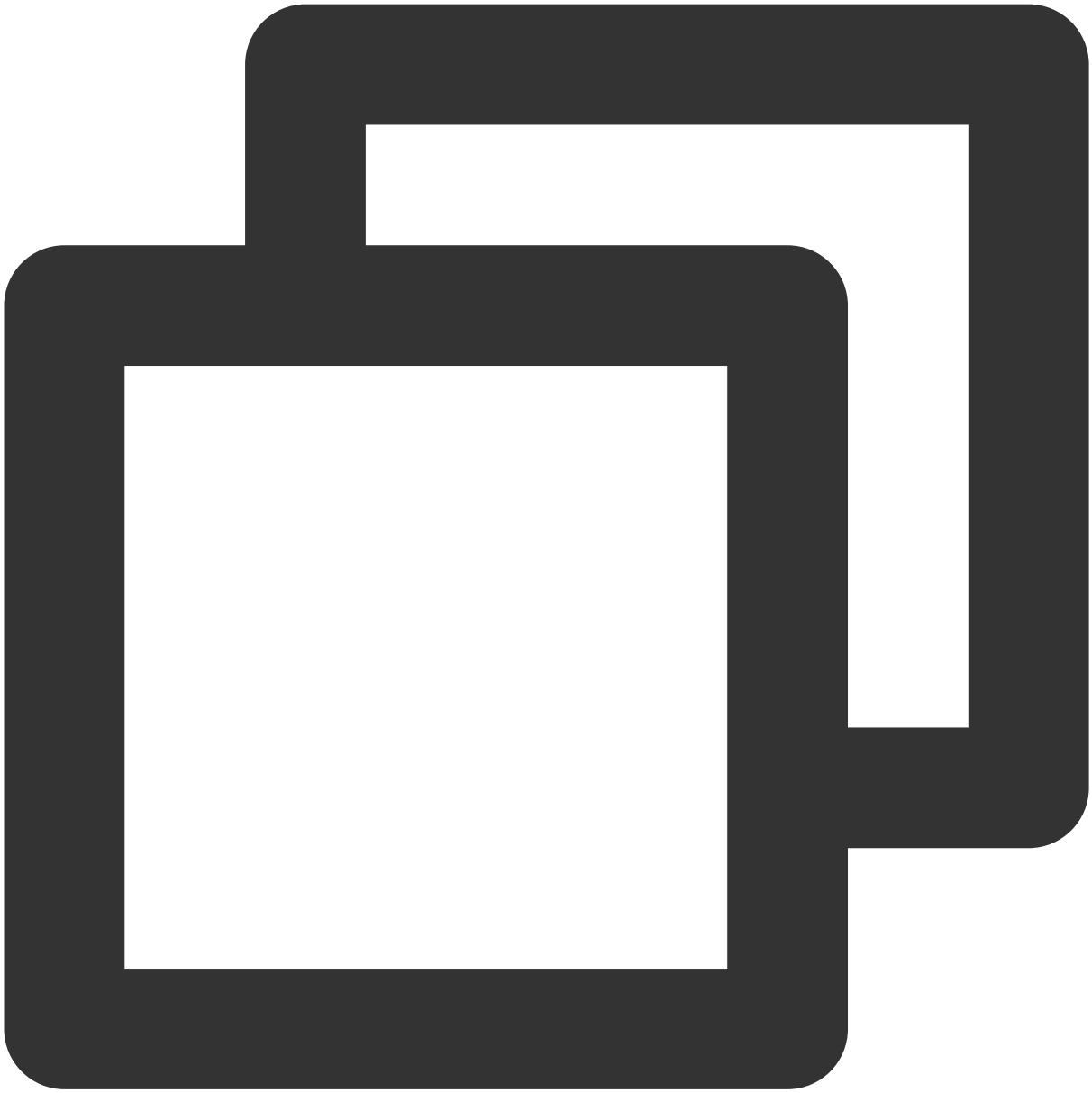
查询一个文件处理任务,根据任务 ID 查询任务详情。

方法原型



```
func (s *CIService) DescribeFileProcessJob(ctx context.Context, jobid string) (*Fil
```

请求示例

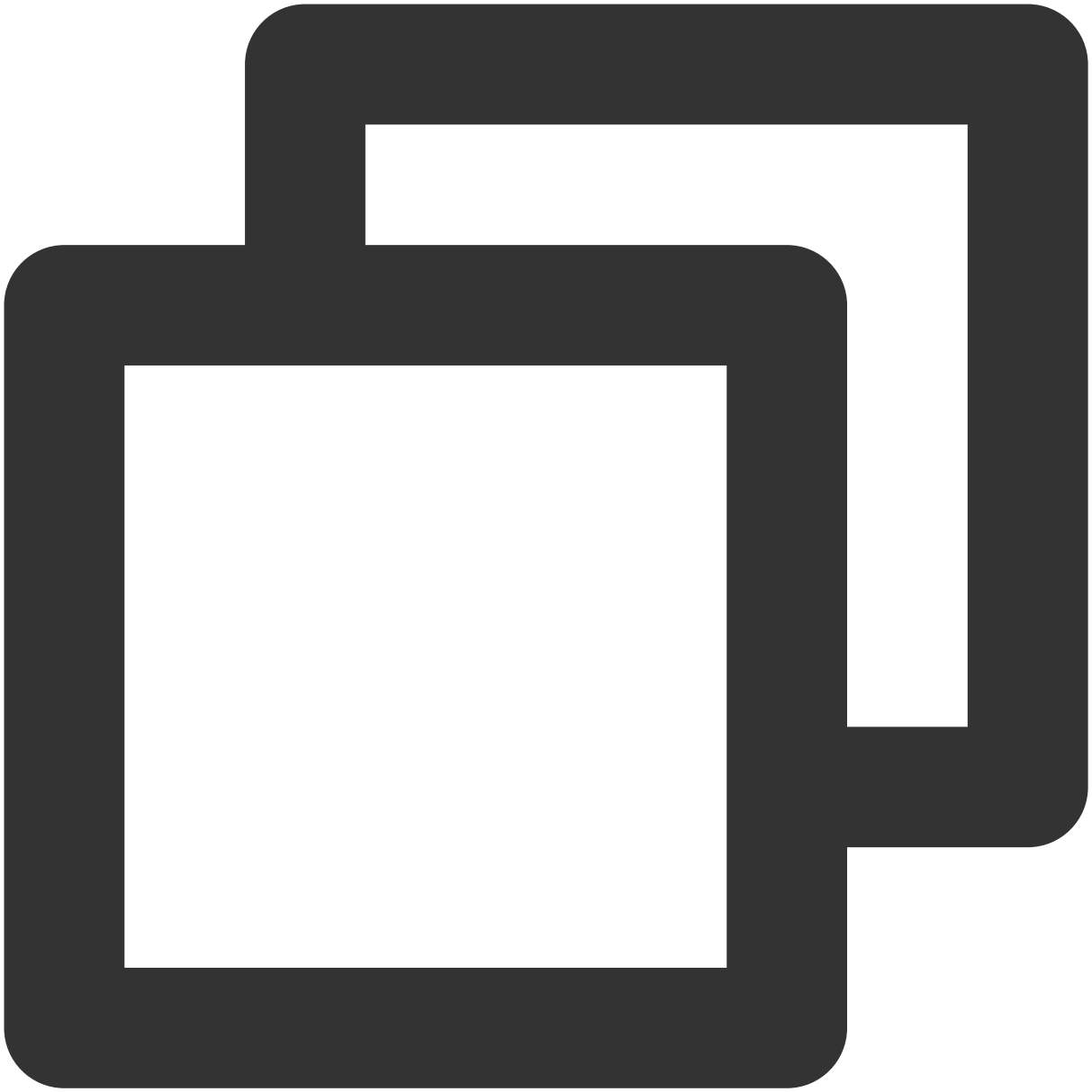


```
jobid := "<jobid>"
res, _, err := c.CI.DescribeFileProcessJob(context.Background(), jobid)
```

参数说明

参数名称	描述	类型	是否必选
jobid	文件处理任务 ID	String	是

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail *FileProcessJobsDetail
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息， 同 CreateFileProcessJob 接口的 Response.JobsDetail 节点	Container

文件哈希值计算

最近更新时间：2024-03-04 16:55:55

简介

本文档提供关于数据万象文件哈希值计算的 API 概览以及 SDK 示例代码。

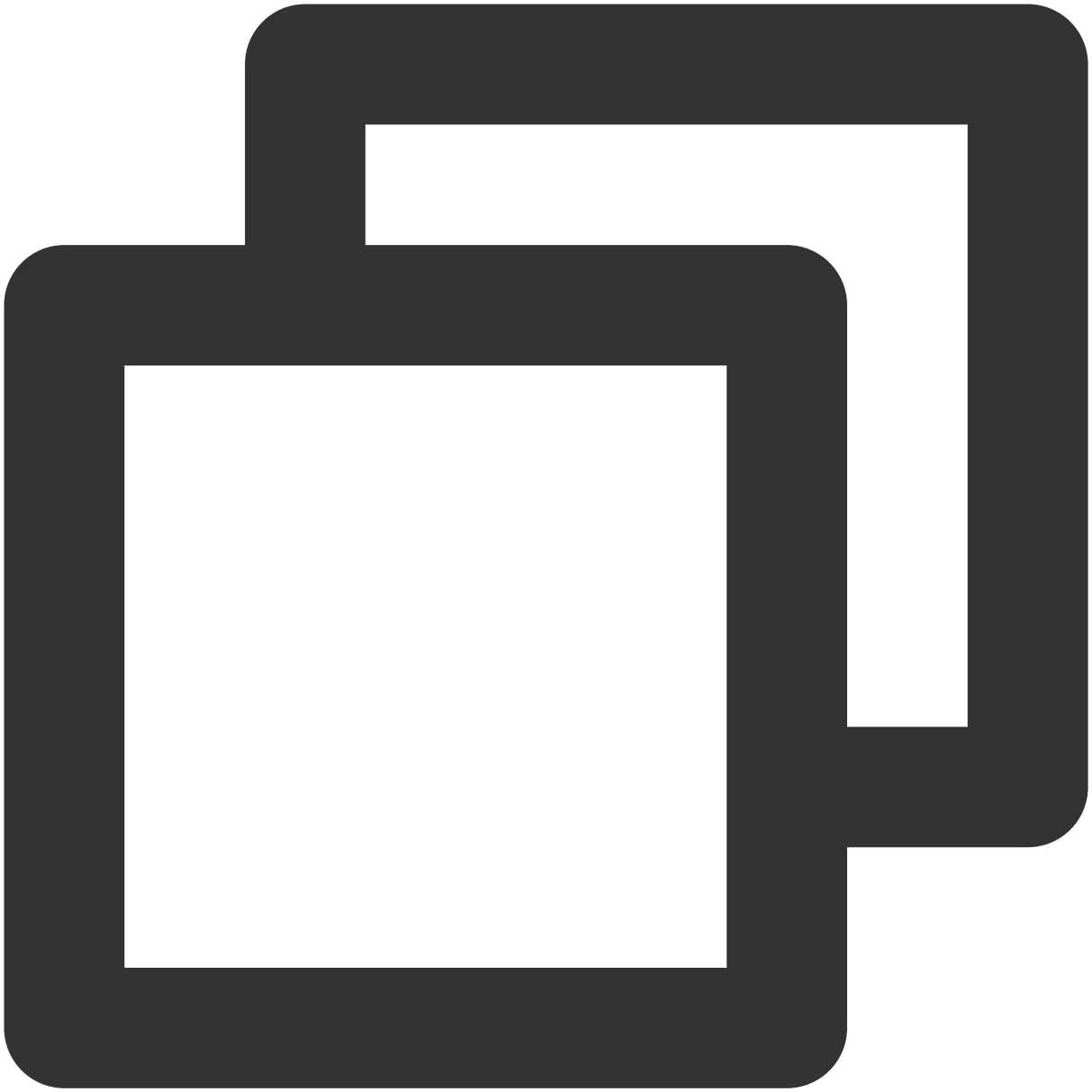
API	操作描述
哈希值计算同步请求	以同步请求的方式进行文件哈希值计算，实时返回计算得到的哈希值
提交哈希值计算任务	以提交任务的方式进行文件哈希值计算任务
查询哈希值计算结果	主动查询指定的文件哈希值计算任务结果

哈希值计算同步请求

功能说明

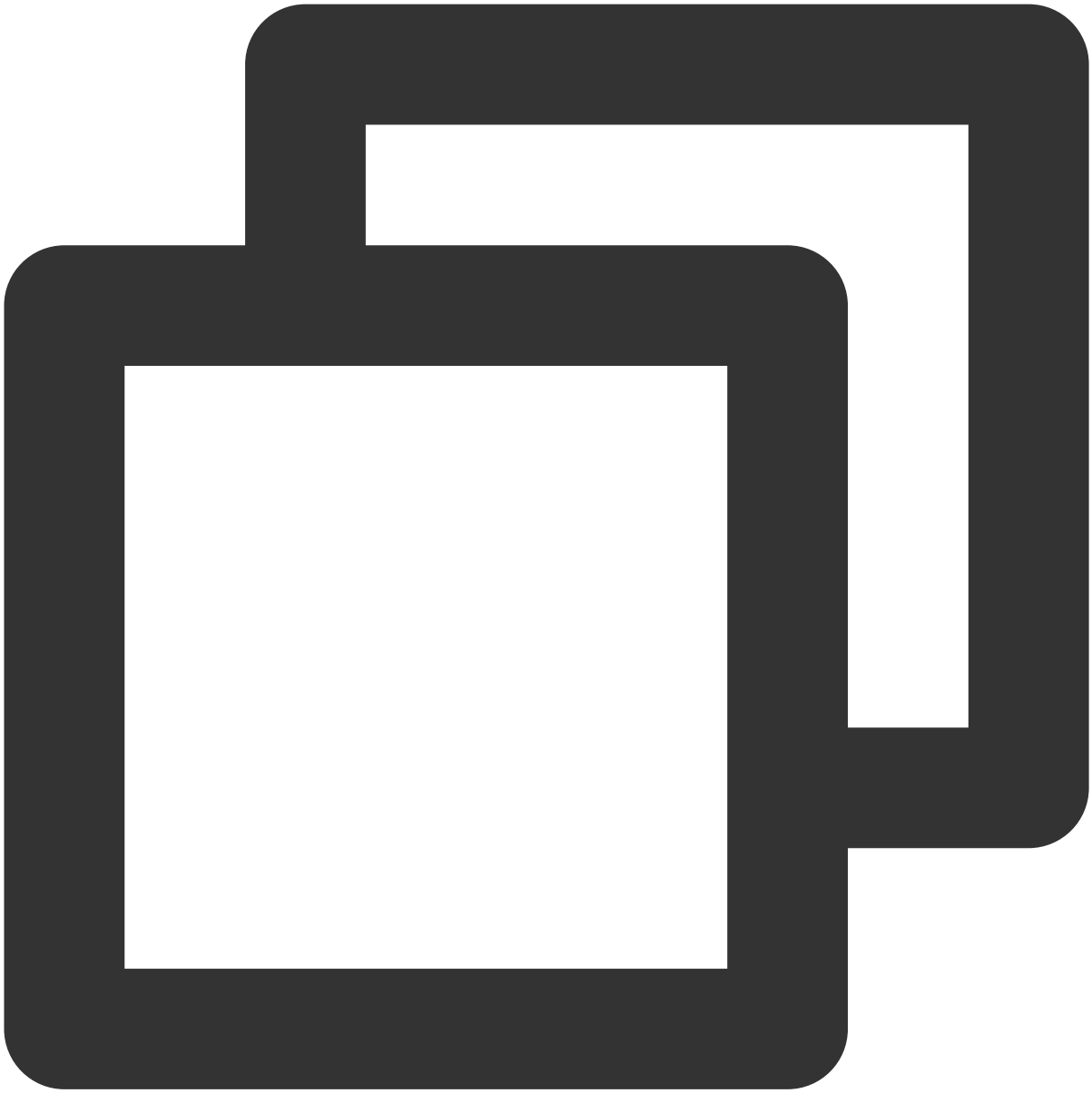
以同步请求的方式进行文件哈希值计算，返回得到的哈希值。

方法原型



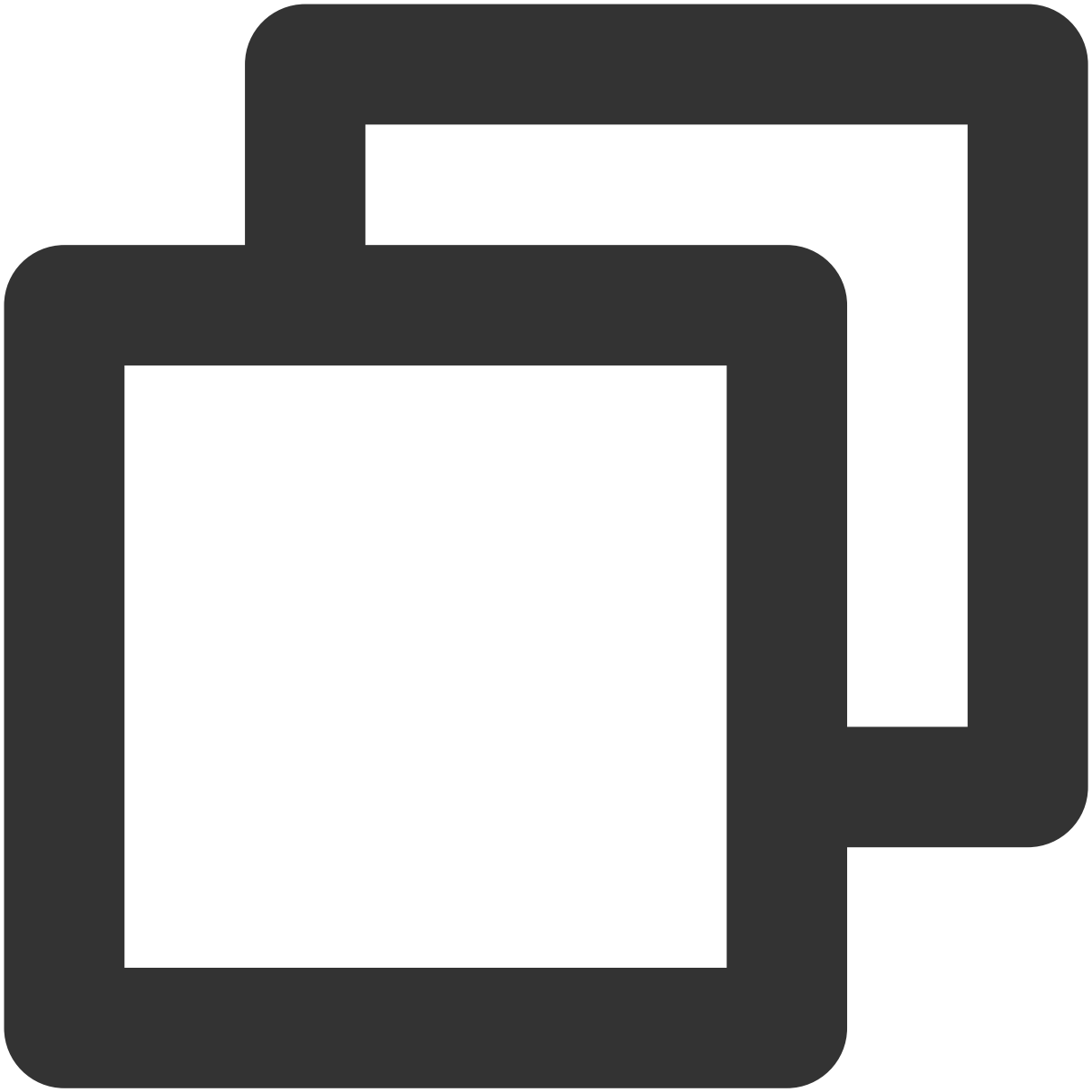
```
func (s *CIService) GetFileHash(ctx context.Context, name string, opt *GetFileHashO
```

请求示例



```
opt := &cos.GetFileHashOptions{
    CIProcess: "filehash",
    Type:      "md5",
}
res, _, err := c.CI.GetFileHash(context.Background(), "test.jpg", opt)
```

参数说明

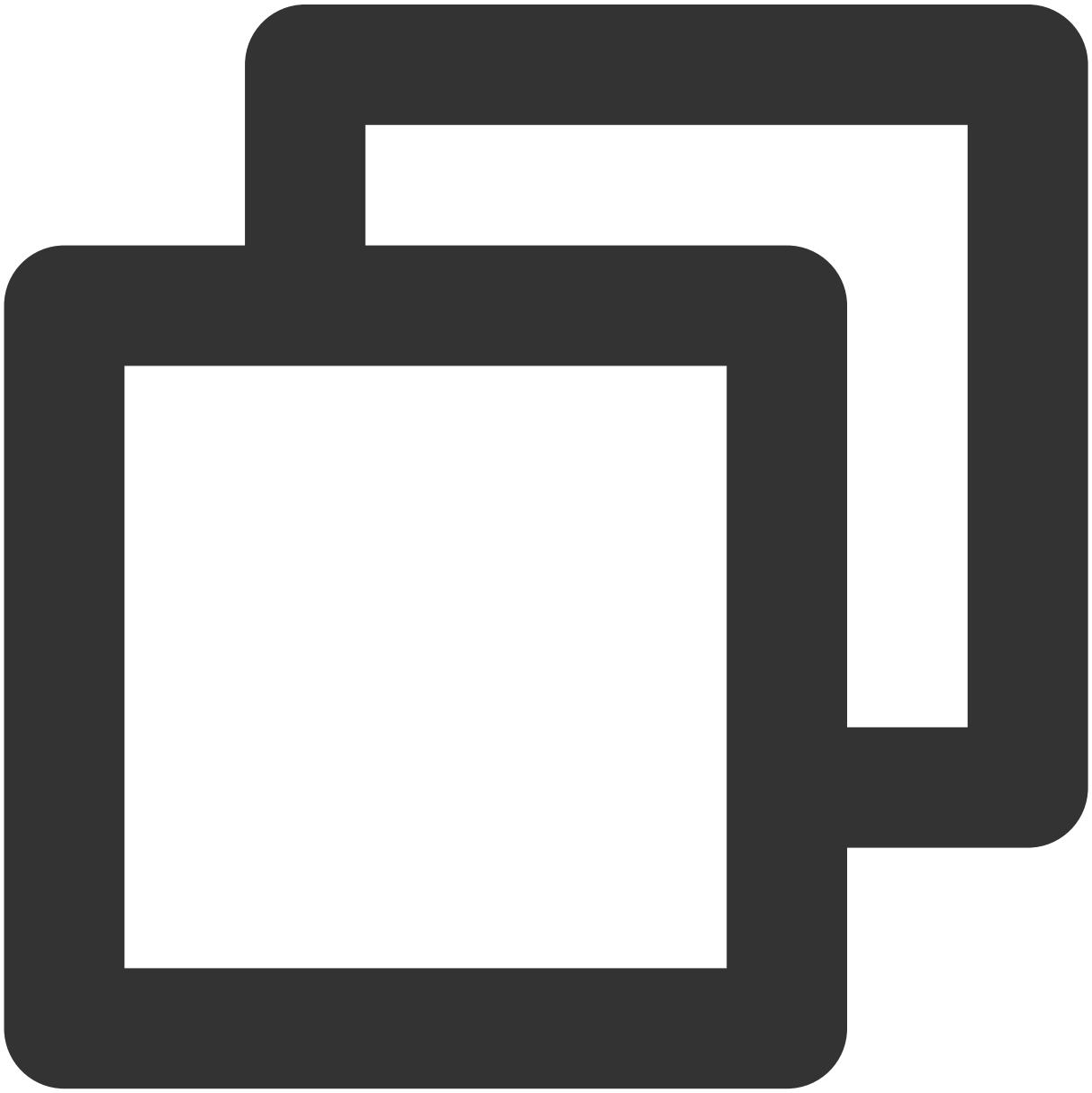


```
type GetFileHashOptions struct {
    CIProcess   string
    Type        string
    AddToHeader bool
}
```

参数名称	描述	类型	是否必选

name	执行操作的文件全路径	String	是
CIProcess	操作类型，哈希值计算固定为：filehash	String	是
Type	支持的哈希算法类型，有效值：md5、sha1、sha256	String	是
AddToHeader	是否将计算得到的哈希值，自动添加至文件的自定义header，格式为：x-cos-meta-md5/sha1/sha256; 有效值： true、false，不填则默认为false。	Bool	否

返回结果说明



```
type GetFileHashResult struct {
    FileHashCodeResult *FileHashCodeResult
    Input              *FileProcessInput
}
```

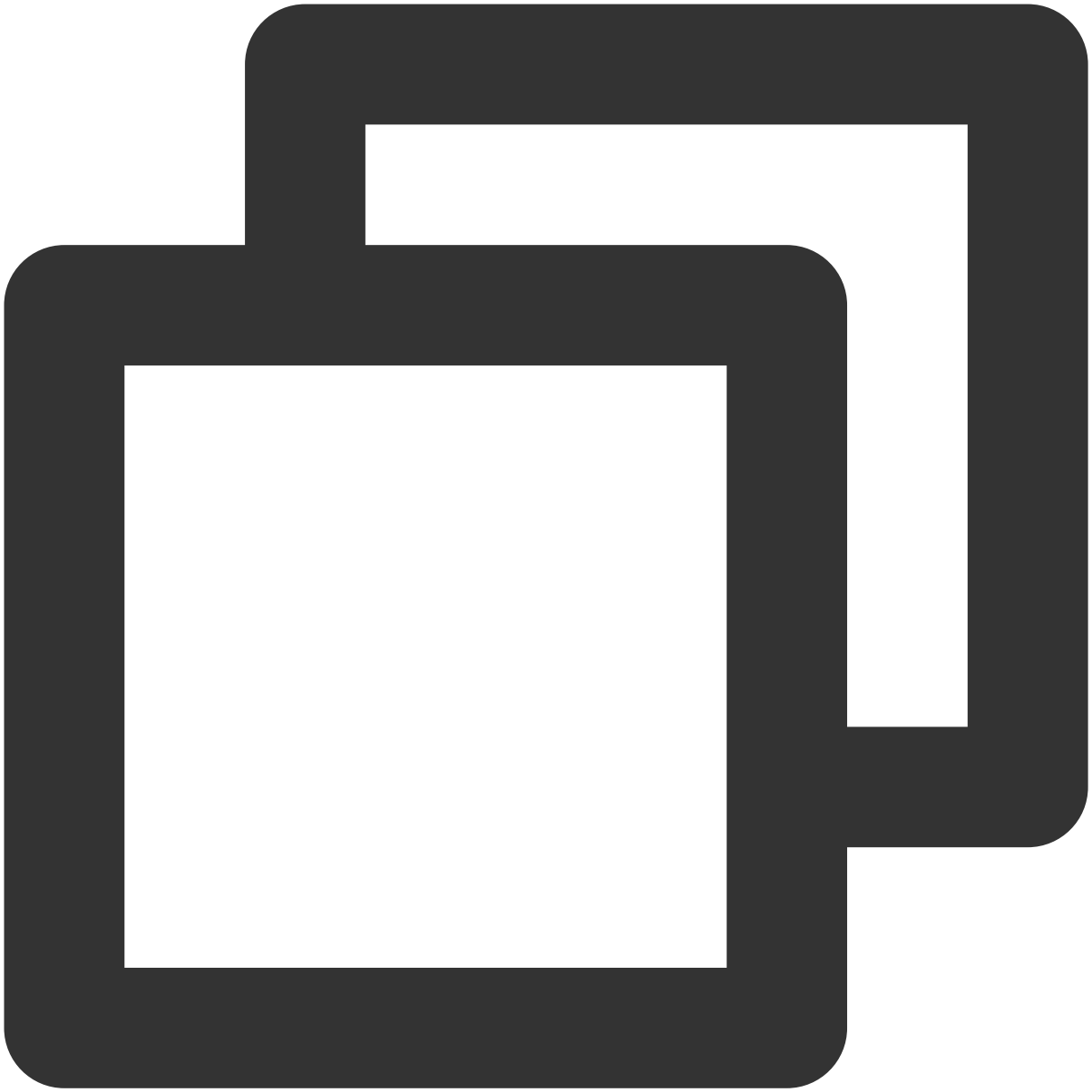
参数名称	描述	类型
FileHashCodeResult	文件哈希值的结果。详情请参见 FileProcessJobOptions.Input	Container
Input	输入文件的基本信息。详情请参见 FileProcessJobOptions.Operation.FileHashCodeResult	Container

提交哈希值计算任务

功能说明

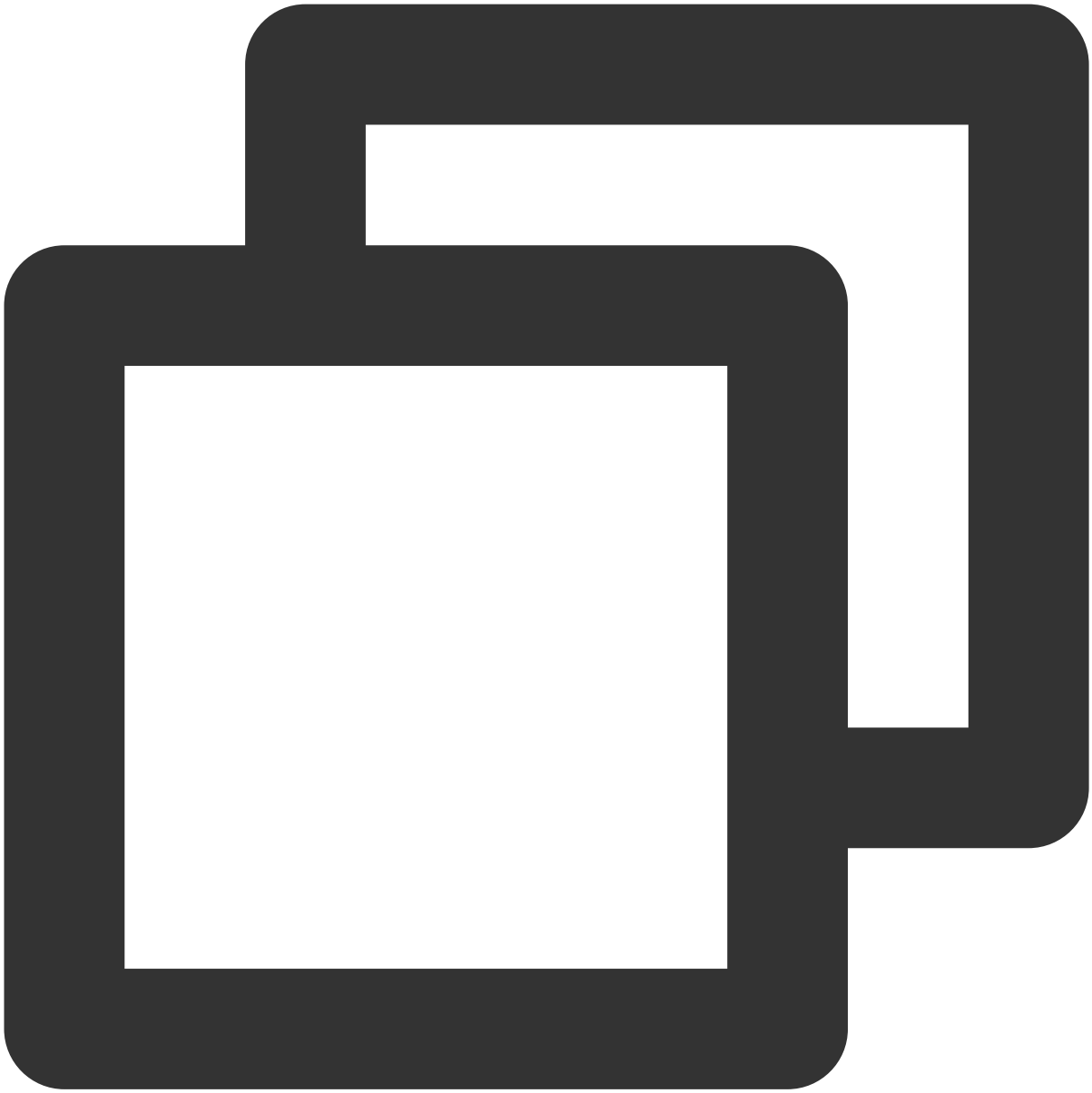
以提交任务的方式进行文件哈希值计算，异步返回计算得到的哈希值。

方法原型



```
func (s *CIService) CreateFileProcessJob(ctx context.Context, opt *FileProcessJobOp
```

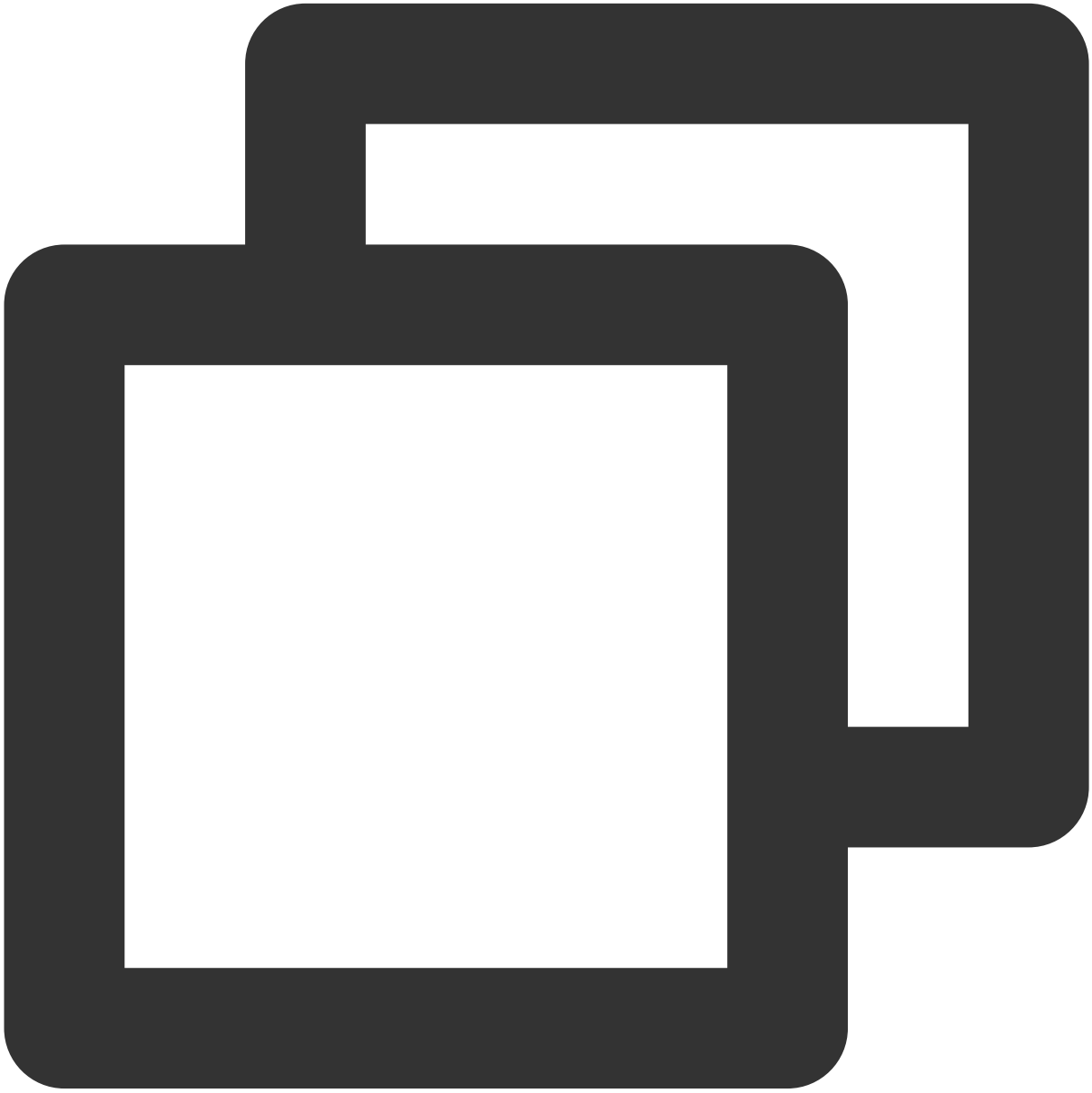
请求示例



```
createJobOpt := &cos.FileProcessJobOptions{
    Tag: "FileHashCode",
    Input: &cos.FileProcessInput{
        Object: "294028.zip",
    },
    Operation: &cos.FileProcessJobOperation{
        FileHashCodeConfig: &cos.FileHashCodeConfig{
            Type:        "sha1",
            AddToHeader: true,
        },
    },
```

```
    QueueId: "p111a8dd208104ce3b11c78398f658ca8",
}
res, _, err := c.CI.CreateFileProcessJob(context.Background(), createJobOpt)
```

参数说明



```
type FileProcessJobOptions struct {
    XMLName          xml.Name
    Tag              string
    Input            *FileProcessInput
    Operation        *FileProcessJobOperation
```

```
    QueueId          string
    CallBackFormat   string
    CallBackType     string
    CallBack         string
    CallBackMqConfig *NotifyConfigCallBackMqConfig
}
type FileProcessInput struct {
    Region string
    Bucket string
    Object string
}
type FileProcessJobOperation struct {
    FileHashCodeConfig   *FileHashCodeConfig
    FileHashCodeResult   *FileHashCodeResult
    FileUncompressConfig *FileUncompressConfig
    FileUncompressResult *FileUncompressResult
    FileCompressConfig   *FileCompressConfig
    FileCompressResult   *FileCompressResult
    Output               *FileProcessOutput
    UserData             string
}
type FileHashCodeConfig struct {
    Type        string
    AddToHeader bool
}
type FileHashCodeResult struct {
    MD5          string
    SHA1         string
    SHA256       string
    FileSize     int
    LastModified string
    Etag         string
}
type FileUncompressConfig struct {
    Prefix         string
    PrefixReplaced string
}
type FileUncompressResult struct {
    Region    string
    Bucket    string
    FileCount string
}
type FileCompressConfig struct {
    Flatten string
    Format  string
    UrlList string
    Prefix  string
```

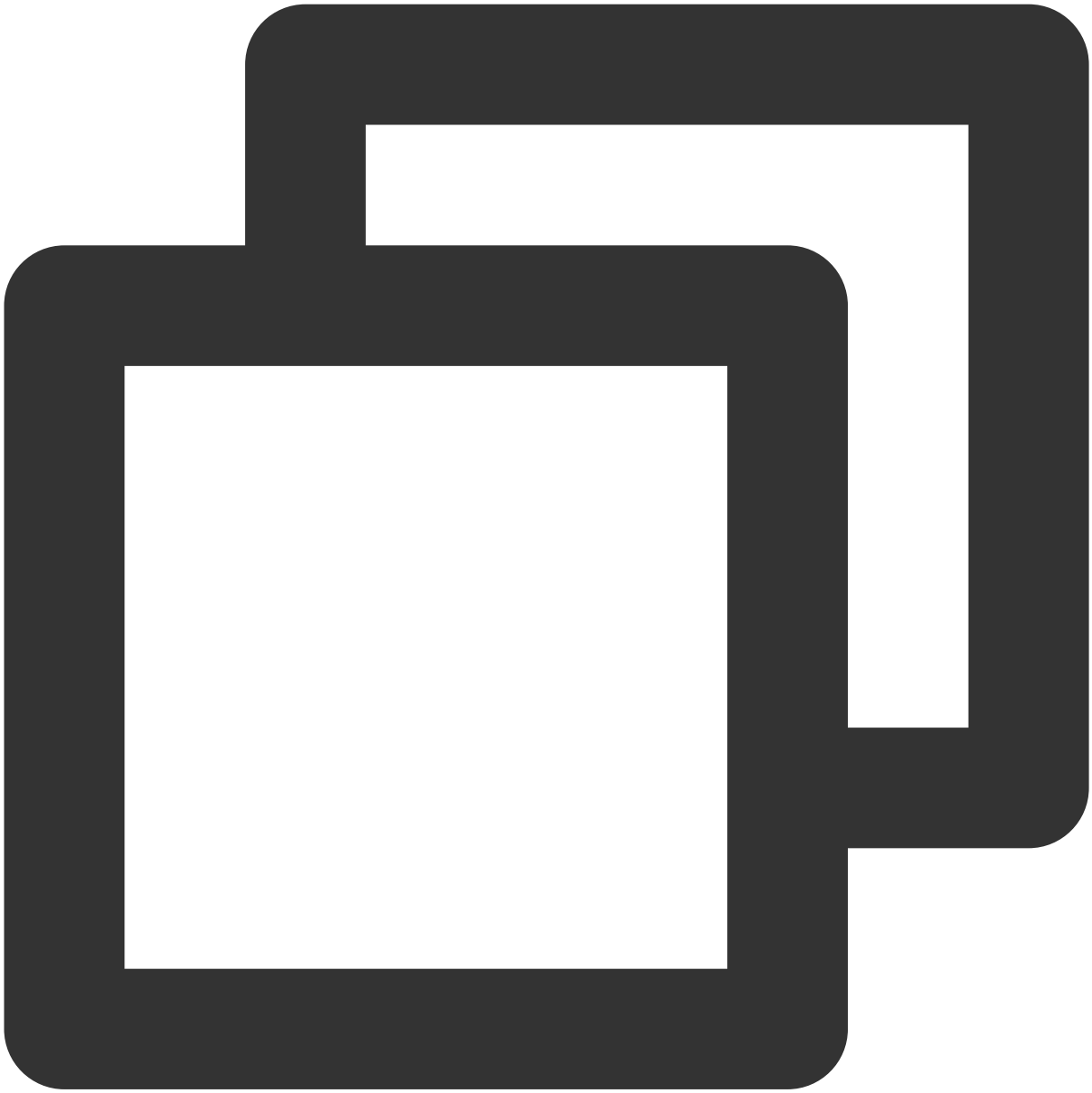
```
    Key     string
}
type FileCompressResult struct {
    Region string
    Bucket string
    Object string
}
type FileProcessOutput struct {
    Region string
    Bucket string
    Object string
}
type NotifyConfigCallBackMqConfig struct {
    MqMode   string
    MqRegion string
    MqName   string
}
```

参数名称	描述	类型	是否必选
Tag	创建任务的 Tag，目前仅支持：FileHashCode	String	是
Input	待操作的文件对象。详情见 Input	Container	是
Operation	操作规则	Container	是
Operation.FileHashCodeConfig	指定哈希值计算的处理规则。详情见 FileHashCodeConfig	Container	是
Operation.FileHashCodeResult	计算得到的文件 hash 值信息，任务未完成时不返回。详情见 FileHashCodeResult	Container	否
Operation.UserData	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
QueueId	任务所在的队列 ID	String	是
CallBackFormat	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时	Container	否

版权所有：腾讯云计算（北京）有限责任公司

	必填。详情见 CallBackMqConfig		

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail FileProcessJobResult
}
type FileProcessJobResult struct {
    Code          string
    Message       string
```

```
    JobId         string
    Tag           string
    State         string
    CreationTime  string
    StartTime     string
    EndTime       string
    QueueId       string
    Input         *FileProcessInput
    Operation     *FileProcessJobOperation
}
```

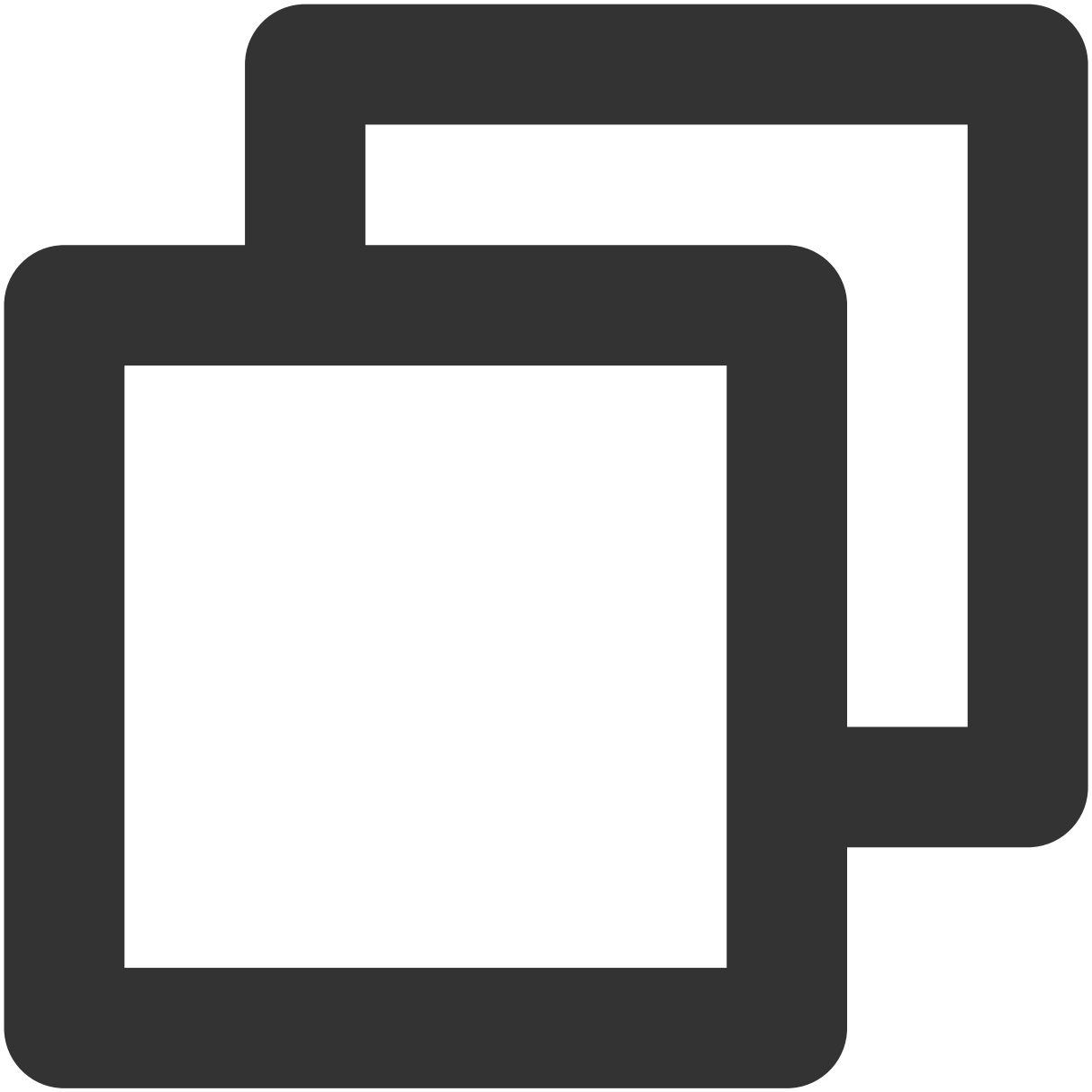
参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：FileHashCode	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String
CreationTime	任务的创建时间	String
StartTime	任务的开始时间	String
EndTime	任务的结束时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径，详情请参见 FileProcessJobOptions.Input	Container
Operation	该任务的规则，详情请参见 FileProcessJobOptions.Operation	Container

查询哈希值计算结果

功能说明

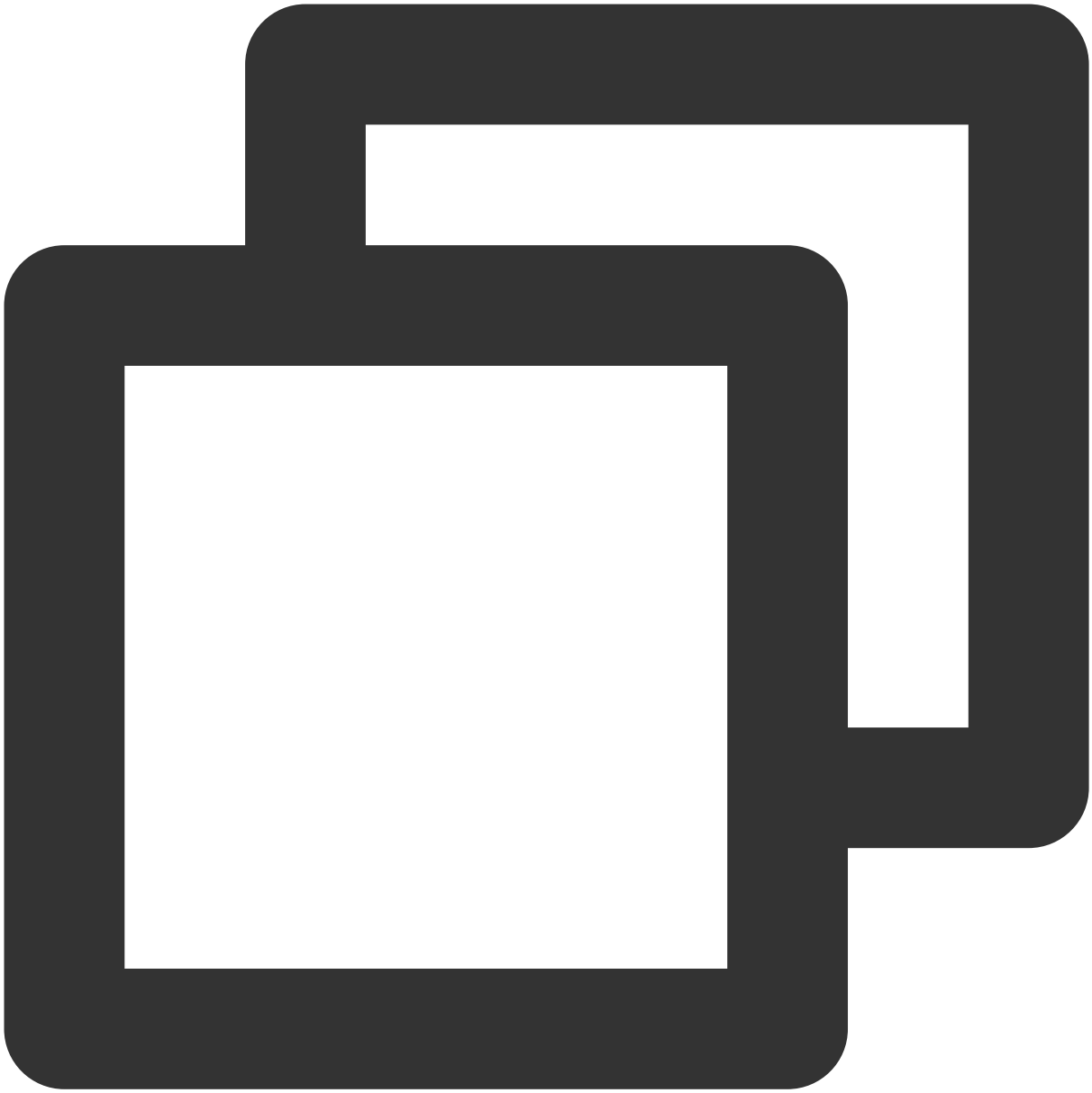
查询一个文件处理任务,根据任务 ID 查询任务详情。

方法原型



```
func (s *CIService) DescribeFileProcessJob(ctx context.Context, jobid string) (*Fil
```

请求示例

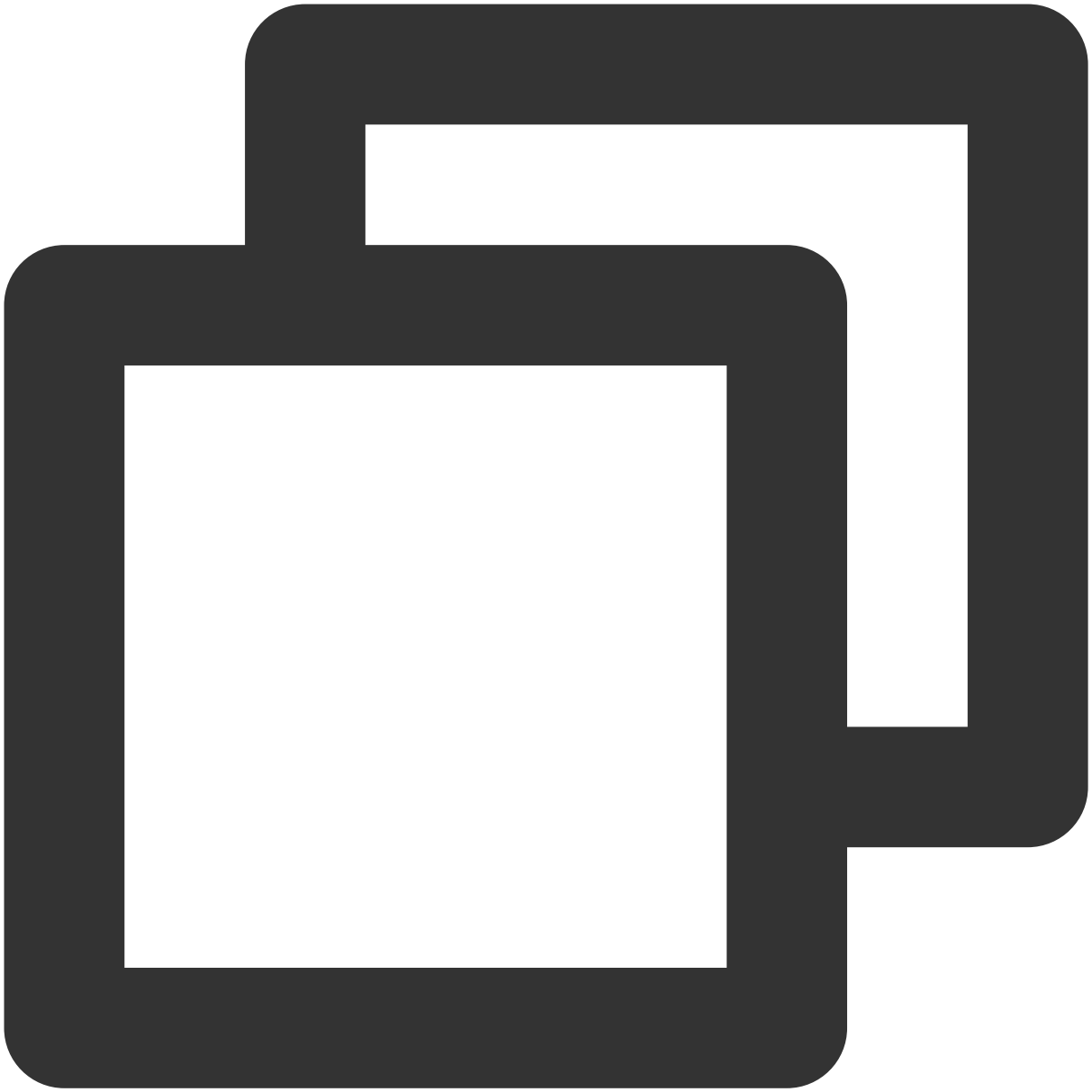


```
jobid := "<jobid>"
res, _, err := c.CI.DescribeFileProcessJob(context.Background(), jobid)
```

参数说明

参数名称	描述	类型
jobid	文件处理任务 ID	String

返回结果说明



```
type FileProcessJobResult struct {
    JobsDetail *FileProcessJobsDetail
}
```

参数名称	描述	类型	是否必选
JobsDetail	任务的详细信息， 同 CreateFileProcessJob 接口的 Response.JobsDetail 节点	Container	是

版权所有：腾讯云计算（北京）有限责任公司

COS iOS SDK
内容审核
图片审核

最近更新时间：2024-03-13 10:07:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核相关的 API 概览以及 SDK 示例代码。

API	操作描述
图片同步审核	对对象存储（Cloud Object Storage，COS）存量数据进行涉黄、违法违规以及广告引导类图片的扫描
图片批量审核	对多个图片进行批量审核
查询图片审核任务结果	本接口用于主动查询指定的图片审核任务结果

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

图片同步审核

功能说明

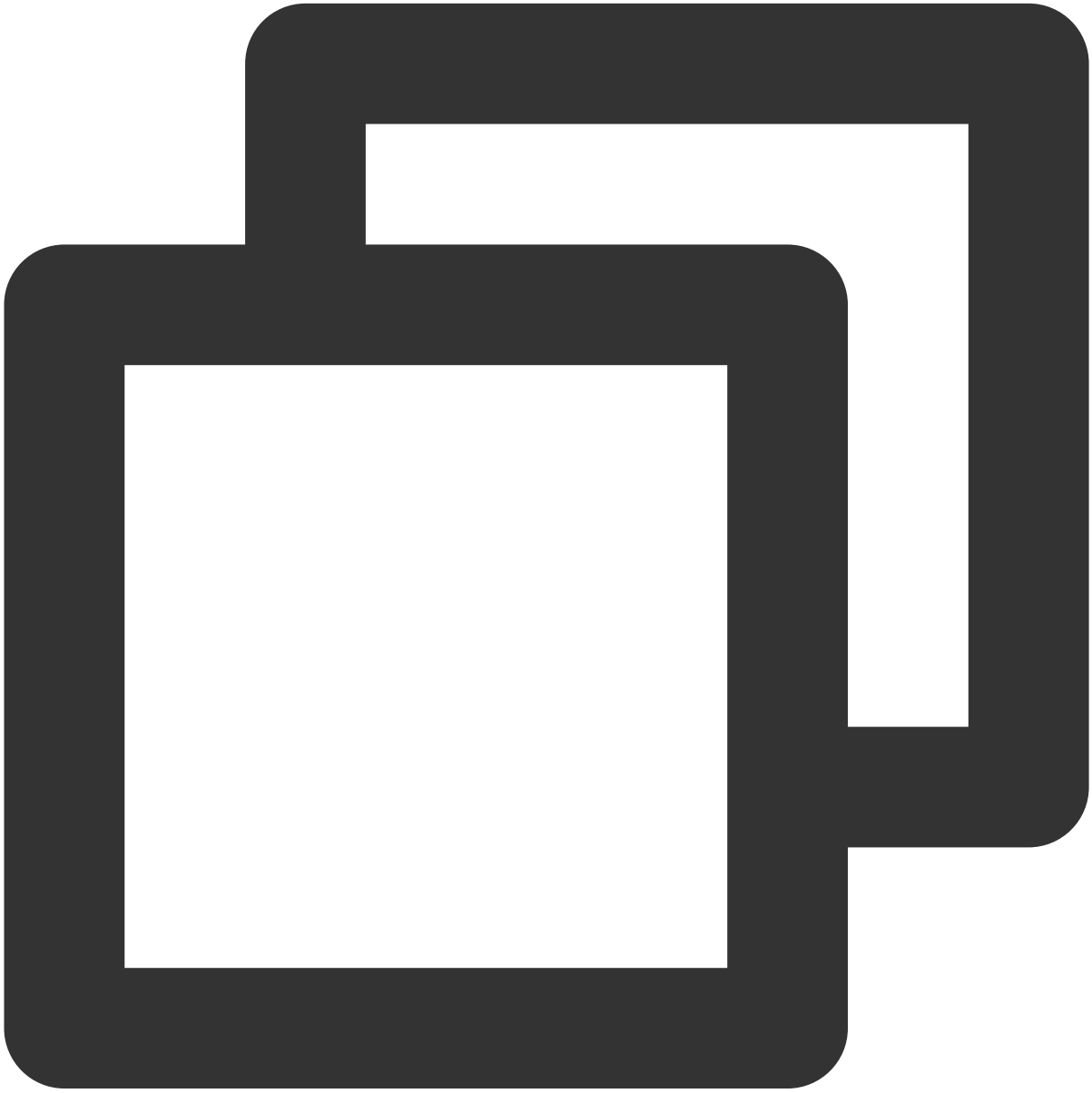
QCloudSyncImageRecognitionRequest 接口用于提交一个视频审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 Jobid 进行查询。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



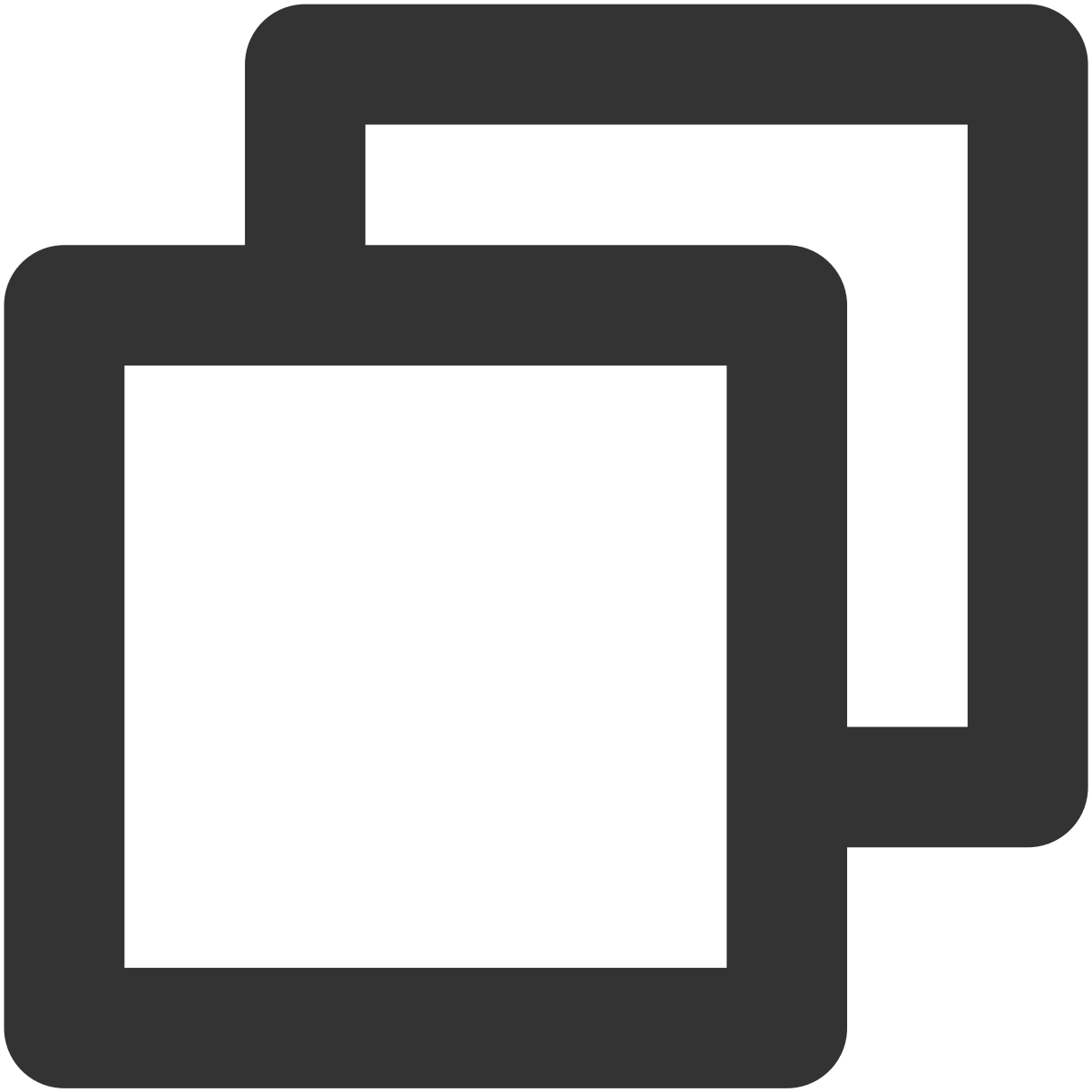
```
QCloudSyncImageRecognitionRequest * request = [[QCloudSyncImageRecognitionRequest a

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"bucket";

// 文件所在地域
```

```
request.regionName = @"regionName";

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = @"***.jpg";

// 审核类型，拥有 porn（涉黄识别）、ads（广告识别）
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionTerrorist | QCloudRec
[request setFinishBlock:^(QCloudImageRecognitionResult * _Nullable result, NSError
// outputObject 提交审核反馈信息，详细字段请查看api文档或者SDK源码
// QCloudImageRecognitionResult 类；
}];
[[QCloudCOSXMLService defaultCOSXML] SyncImageRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request : QCloudSyncImageRecognitionRequest = QCloudSyncImageRecognitionRequest

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "bucket";

// 文件所在地域
request.regionName = "regionName";

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = "***.jpg";
```

```
// 审核类型，拥有 porn（涉黄识别）、ads（广告识别）
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

request.finishBlock = { (result, error) in
        // outputObject 提交审核反馈信息，详细字段请查看api文档或者SDK源码
    // QCloudImageRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().syncImageRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

图片批量审核

功能说明

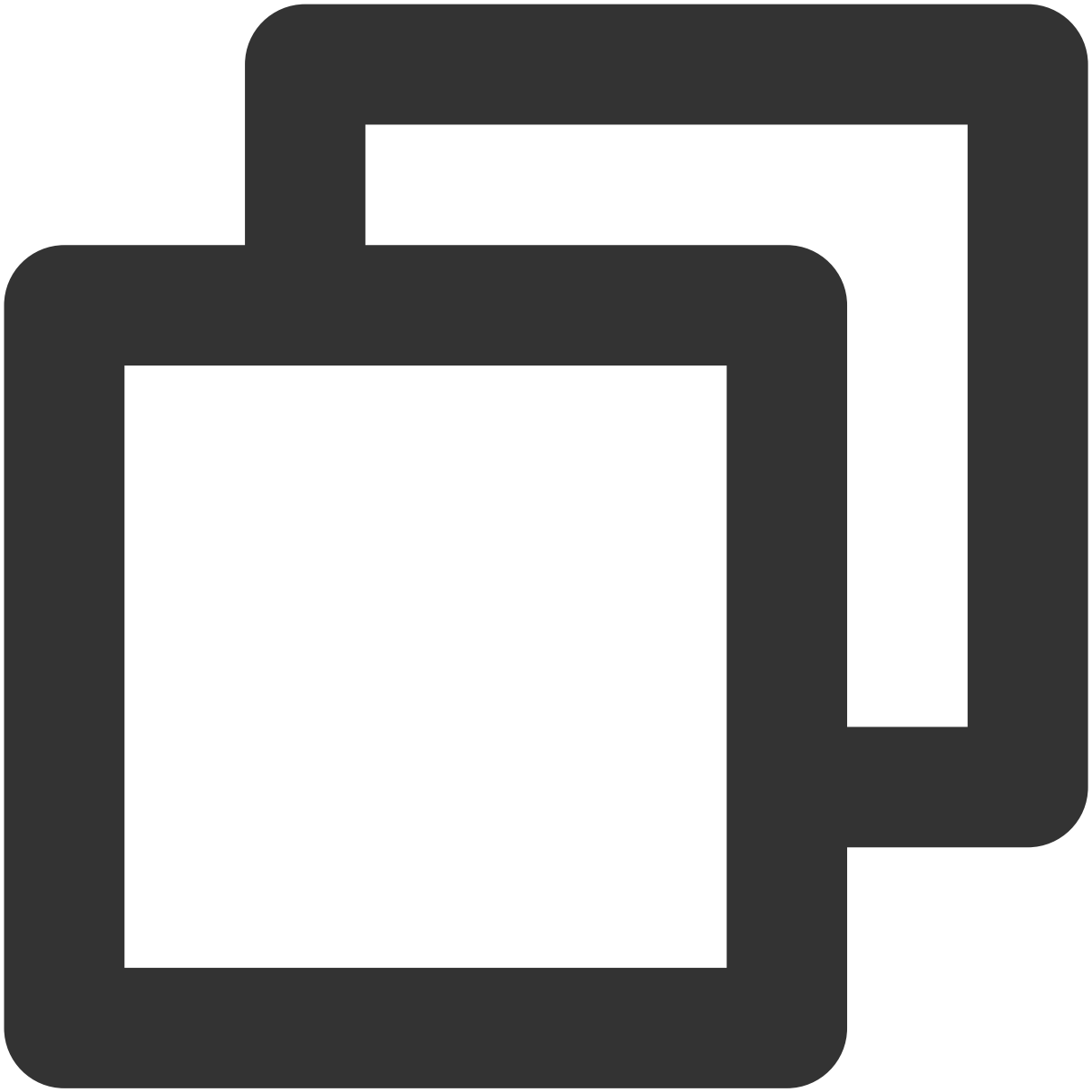
图片批量审核接口为同步请求方式，您可以通过本接口对多个图片文件进行内容审核。该接口属于 POST 请求。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



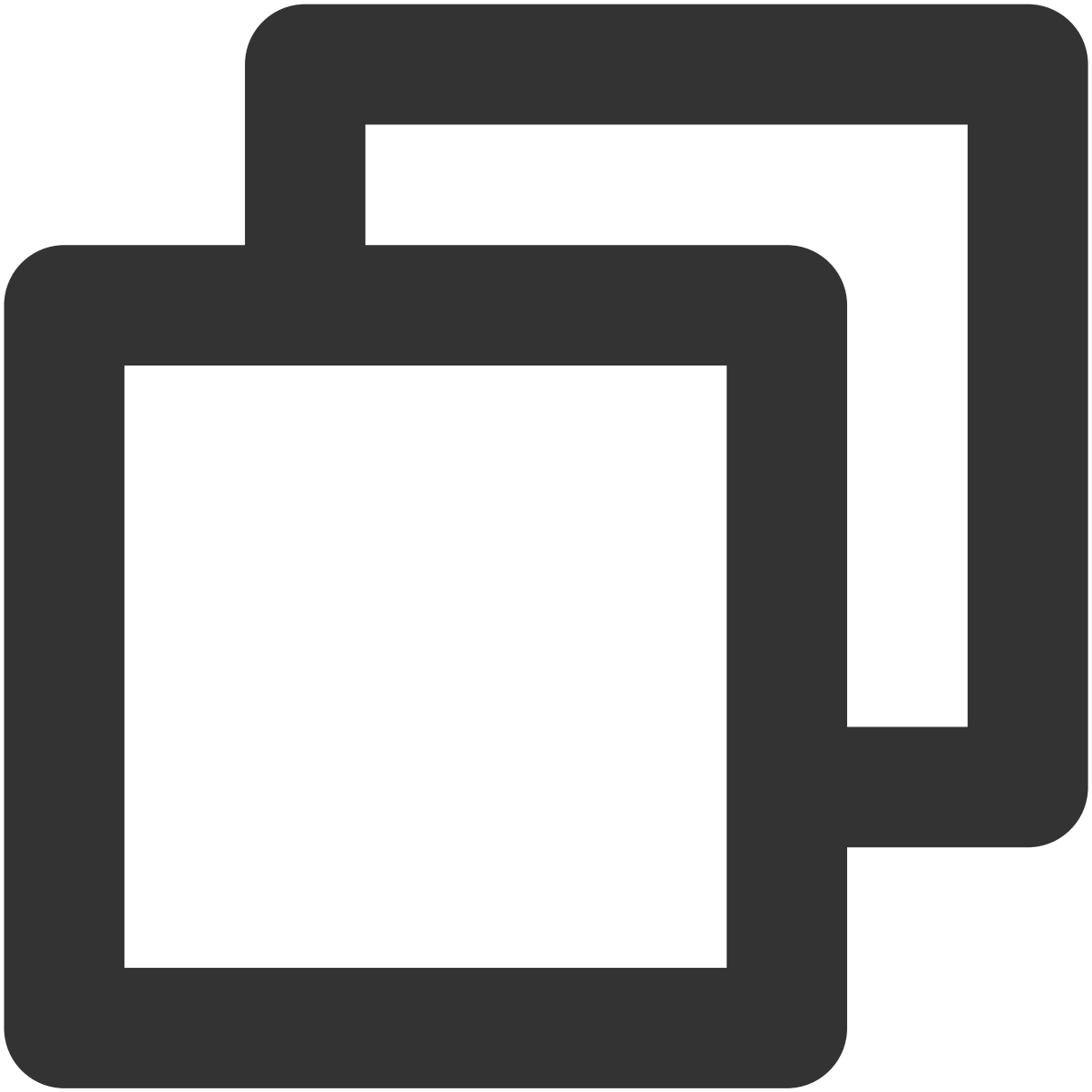
```
QCloudBatchimageRecognitionRequest * request = [[QCloudBatchimageRecognitionRequest

request.bucket = @"bucket";
// 文件所在地域
request.regionName = @"regionName";

NSMutableArray * input = [NSMutableArray new];

// 待审核的图片对象
QCloudBatchRecognitionImageInfo * input1 = [QCloudBatchRecognitionImageInfo new];
```

```
input1.Object = @"***.jpg";
[input addObject:input1];

QCloudBatchRecognitionImageInfo * input2 = [QCloudBatchRecognitionImageInfo new];
input2.Object = @"***.jpg";
[input addObject:input2];

// 待审核的图片对象数组
request.input = input;
request.detectType = QCloudRecognitionPorn | QCloudRecognitionTerrorist | QCloudRec
[request setFinishBlock:^(QCloudBatchImageRecognitionResult * _Nullable result, NSE
// outputObject 审核结果，详细字段请查看api文档或者SDK源码
// QCloudBatchImageRecognitionResult 类；
}];
[[QCloudCOSXMLService defaultCOSXML] BatchImageRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudBatchimageRecognitionRequest();
request.bucket = "bucket";

// 文件所在地域
request.regionName = "regionName";

// 待审核的图片对象
let input1 = QCloudBatchRecognitionImageInfo();
input1.object = "***.jpg";

let input2 = QCloudBatchRecognitionImageInfo();
```

```
input2.object = "***.jpg";

// 待审核的图片对象数组
request.input = [input1,input2];
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw
request.setFinish { outputObject, error in
// outputObject 审核结果，详细字段请查看api文档或者SDK源码
// QCloudBatchImageRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().batchImageRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询图片审核任务结果

功能说明

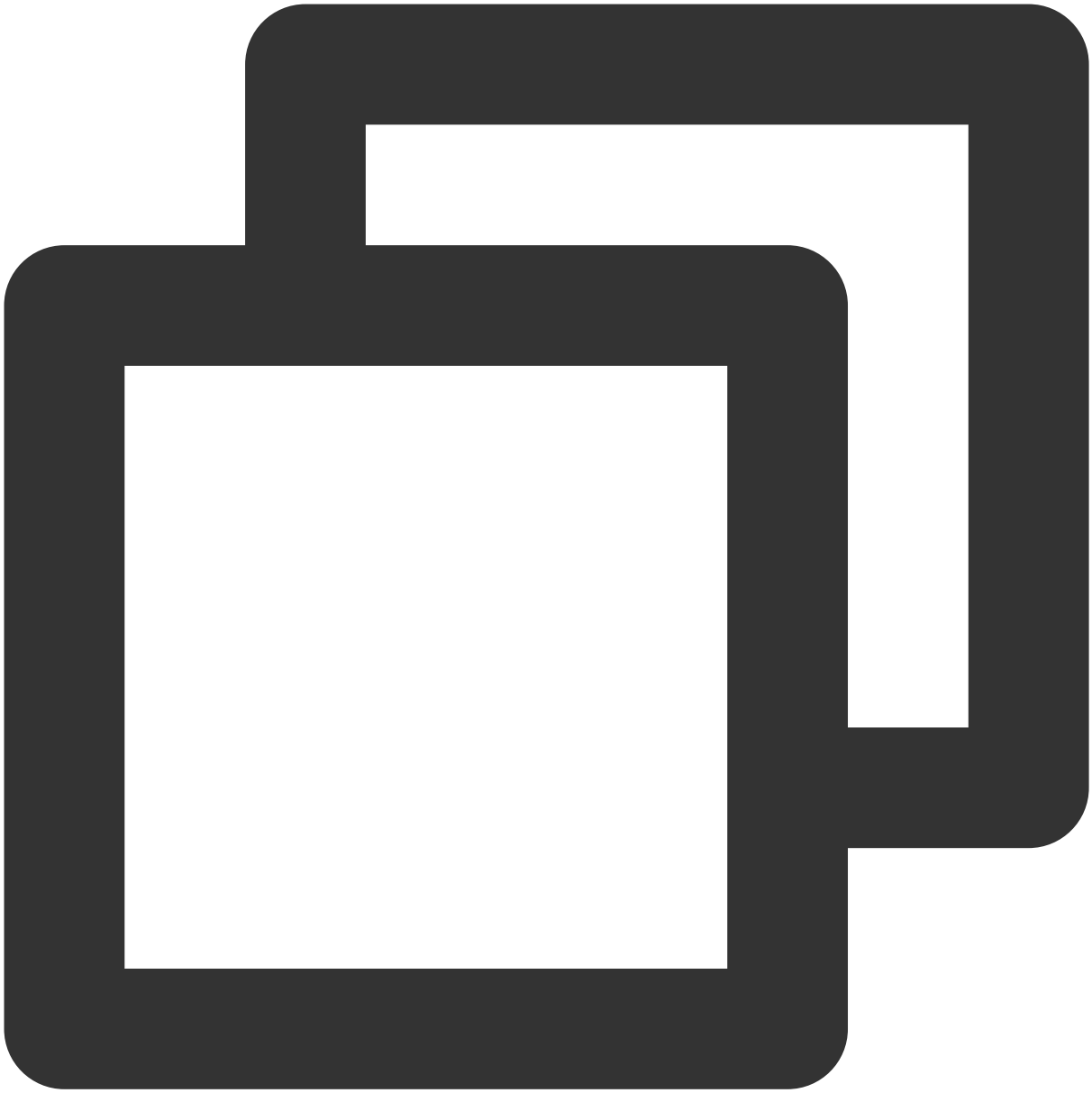
QCloudGetImageRecognitionRequest 本接口用于主动查询指定的图片审核任务结果。。您可以根据图片同步审核或批量审核任务的 JobId 来查询图片审核结果。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



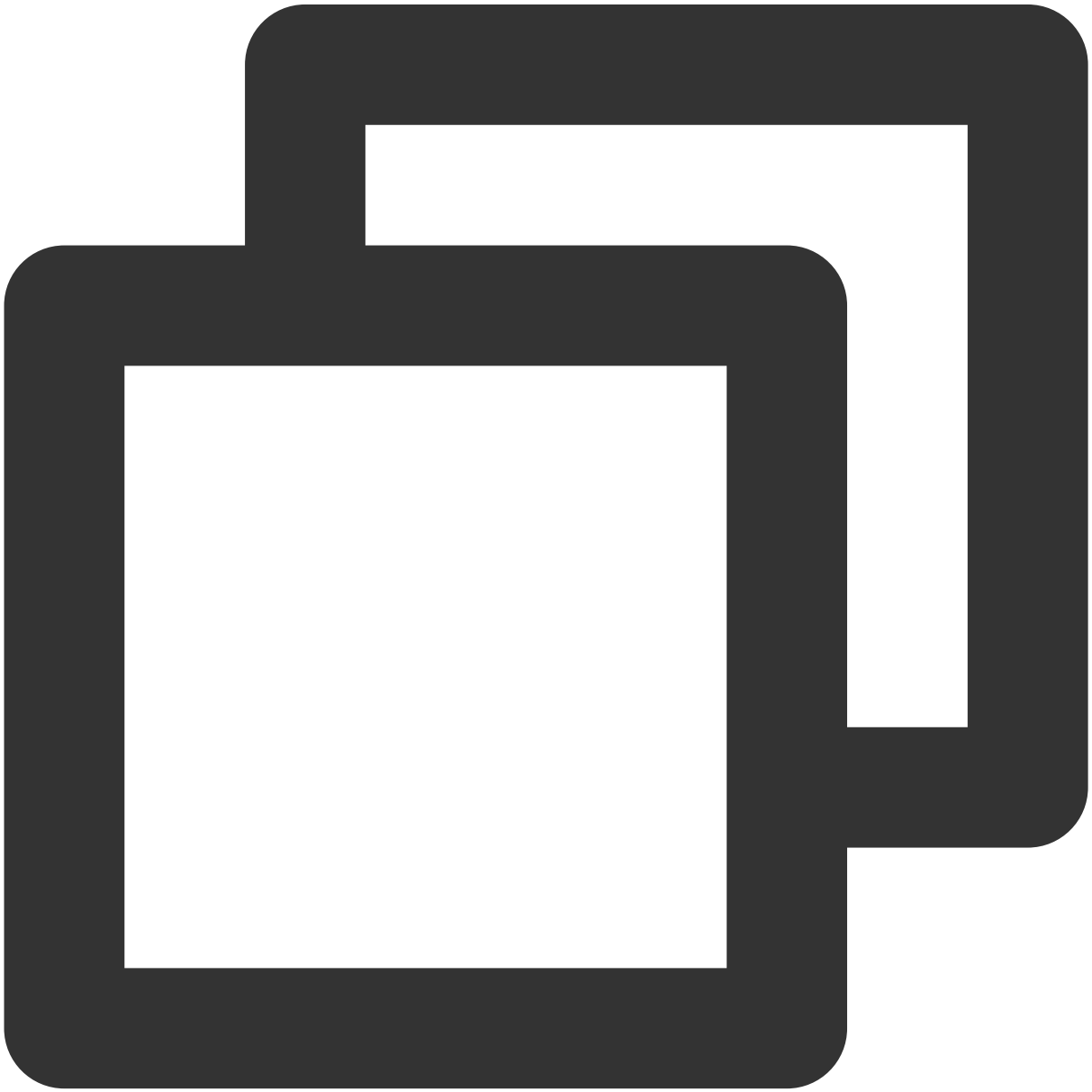
```
QCloudGetImageRecognitionRequest * request = [[QCloudGetImageRecognitionRequest all

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

// 同步审核或批量审核返回结果的jobid
request.jobId = @"jobid";
```

```
request.finishBlock = ^(QCloudImageRecognitionResult * outputObject, NSError *error
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudImageRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetImageRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudGetImageRecognitionRequest();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

request.regionName = "regionName";

// 同步审核或批量审核返回结果的jobid
request.jobId = "jobid";

request.setFinish { outputObject, error in
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudWebRecognitionResult 类；
};
QCloudCOSXMLService.defaultCOSXML().getImageRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

视频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
提交视频审核任务	用于提交一个视频审核任务
查询视频审核任务结果	用于查询指定的视频审核任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

创建任务

功能说明

QCloudPostVideoRecognitionRequest 接口用于提交一个视频审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 Jobid 进行查询。

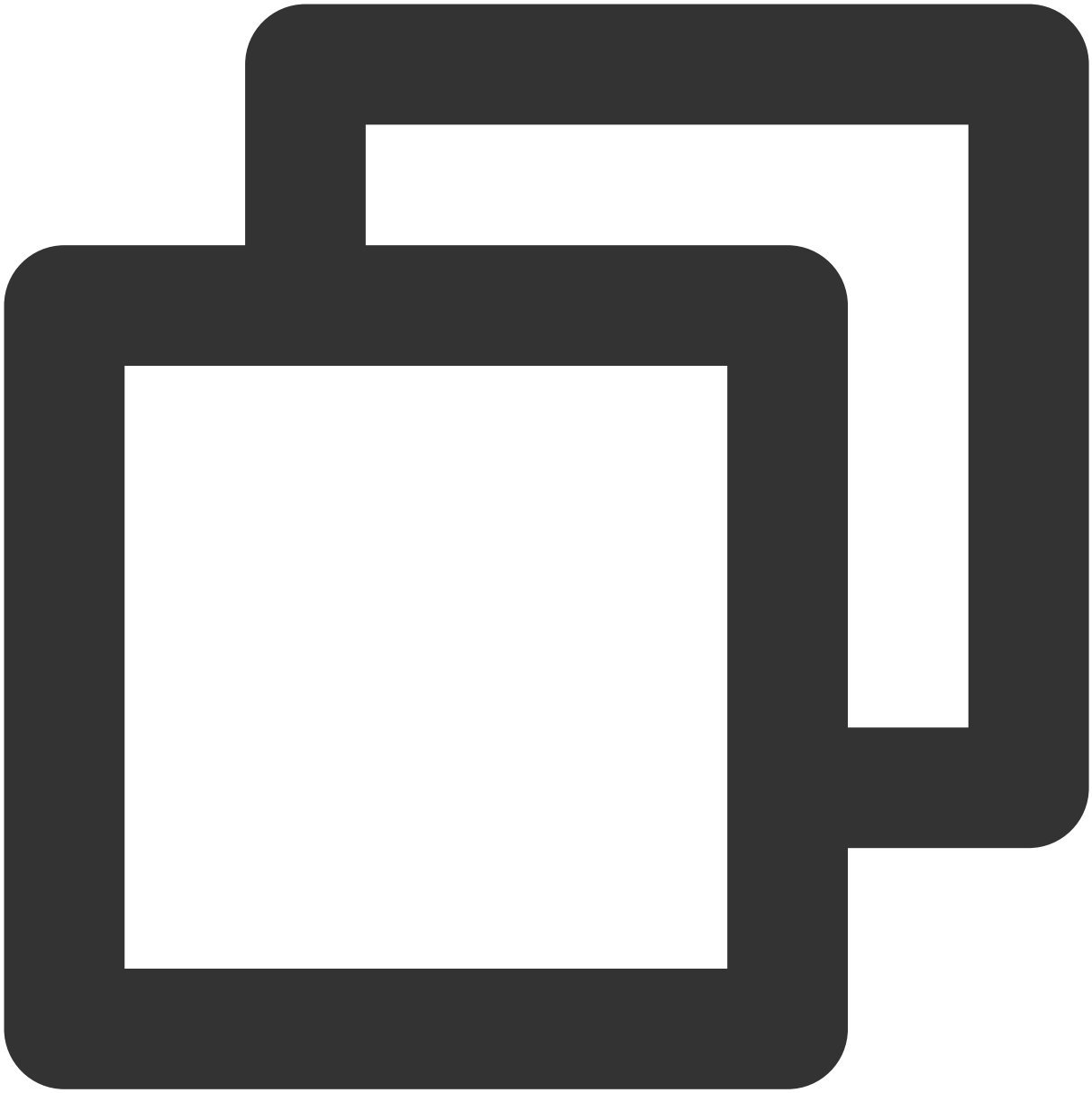
注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C

Swift

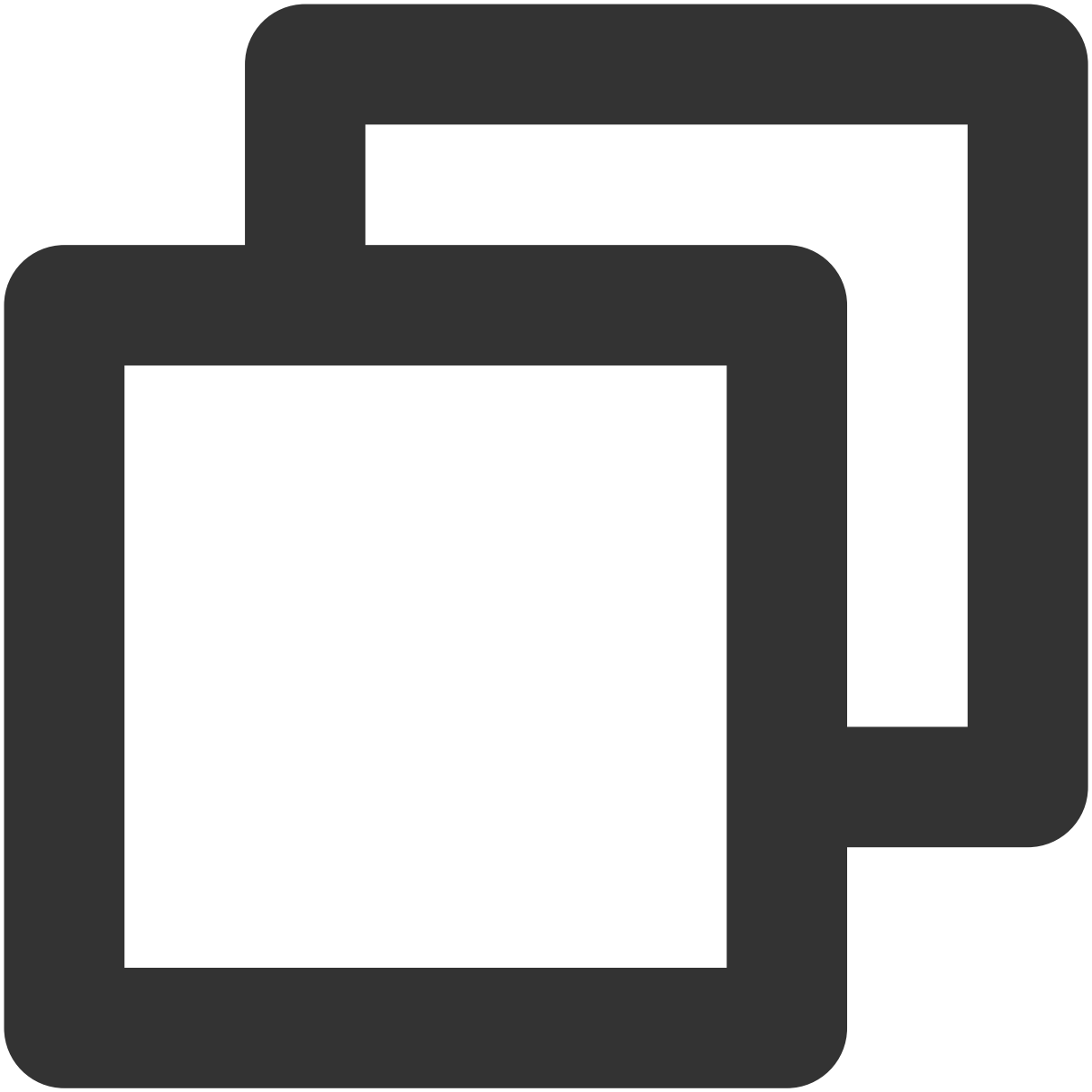


```
QCloudPostVideoRecognitionRequest * request = [[QCloudPostVideoRecognitionRequest a

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = @"exampleobject";

// 文件所在地域
request.regionName = @"regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";
```

```
// 审核类型，拥有 porn（涉黄识别）、ads（广告识别）
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionAds;

// 截帧模式。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。
// Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示
// Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。
// Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧。
request.mode = QCloudVideoRecognitionModeFps;

// 视频截帧频率，范围为(0.000, 60.000]，单位为秒，支持 float 格式，执行精度精确到毫秒
request.timeInterval = 1;

// 视频截帧数量，范围为(0, 10000]。
request.count = 10;

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.bizType = @"BizType";

// 用于指定是否审核视频声音，当值为 0 时：表示只审核视频画面截图；值为1时：表示同时审核视频画面截图
request.detectContent = YES;

request.finishBlock = ^(QCloudPostVideoRecognitionResult * outputObject, NSError *e
// outputObject 提交审核反馈信息 包含用于查询的 job id，详细字段请查看 api 文档或者 SDK 源码
// QCloudPostVideoRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] PostVideoRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。



```
let request : QCloudPostVideoRecognitionRequest = QCloudPostVideoRecognitionRequest

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = "exampleobject";

// 文件所在地域
request.regionName = "regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";
```

```
// 审核类型，拥有 porn（涉黄识别）、ads（广告识别）
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

// 截帧模式。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。
// Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示
// Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。
// Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧。
request.mode = QCloudVideoRecognitionMode.fps;

// 视频截帧频率，范围为(0.000, 60.000]，单位为秒，支持 float 格式，执行精度精确到毫秒
request.timeInterval = 1;

// 视频截帧数量，范围为(0, 10000]。
request.count = 10;

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.bizType = "BizType";

// 用于指定是否审核视频声音，当值为 0 时：表示只审核视频画面截图；值为1时：表示同时审核视频画面截图
request.detectContent = true;

request.finishBlock = { (result, error) in
        // outputObject 提交审核反馈信息 包含用于查询的 job id，详细字段请查看 api 文档或者
    // QCloudPostVideoRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().postVideoRecognition(request);
```

说明：

完整示例，请前往 GitHub 查看。

查询任务

功能说明

QCloudGetVideoRecognitionRequest 接口用来查询指定的视频审核任务。您可以根据视频审核任务的 job Id 来查询视频审核结果。

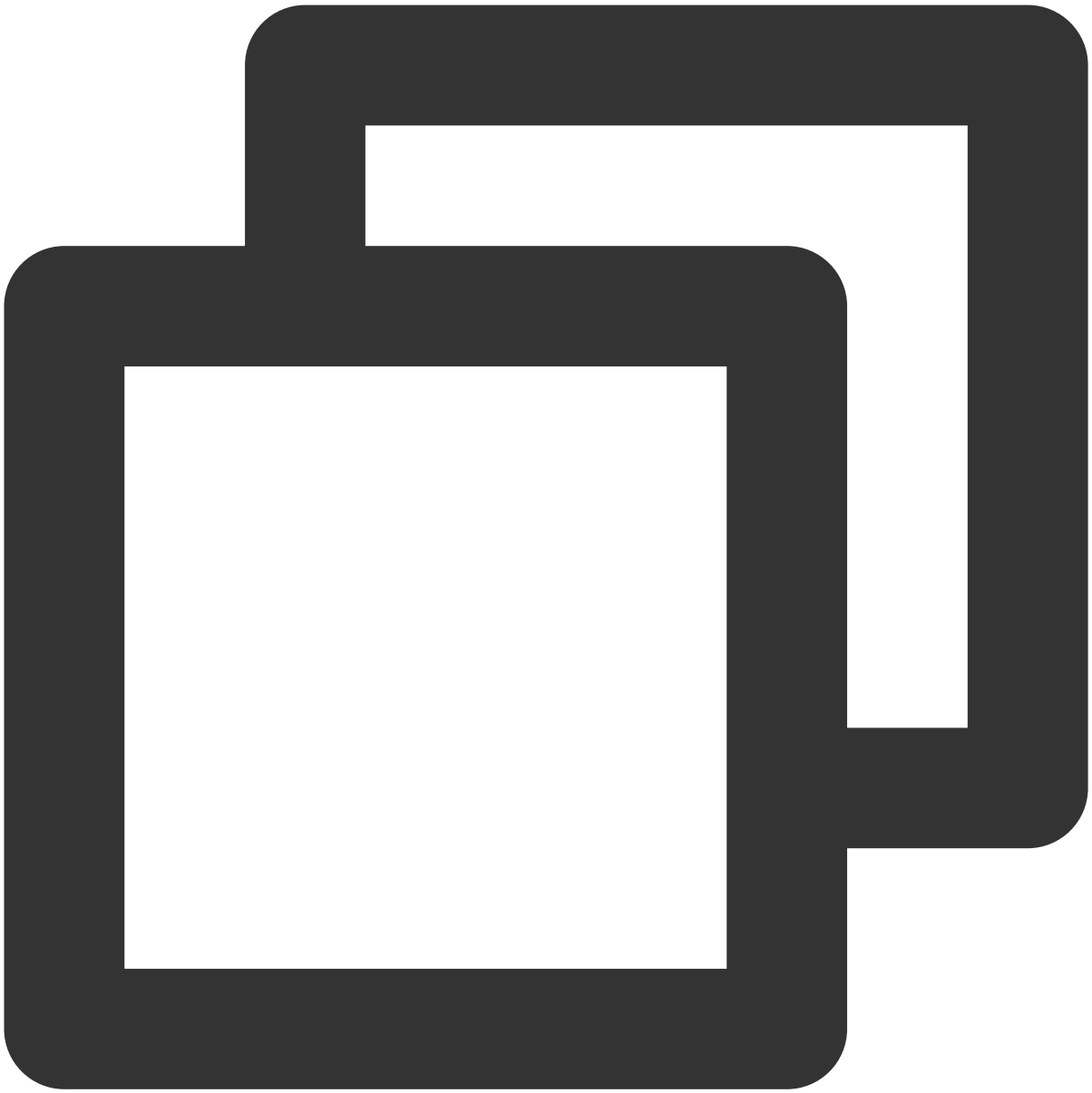
注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C

Swift

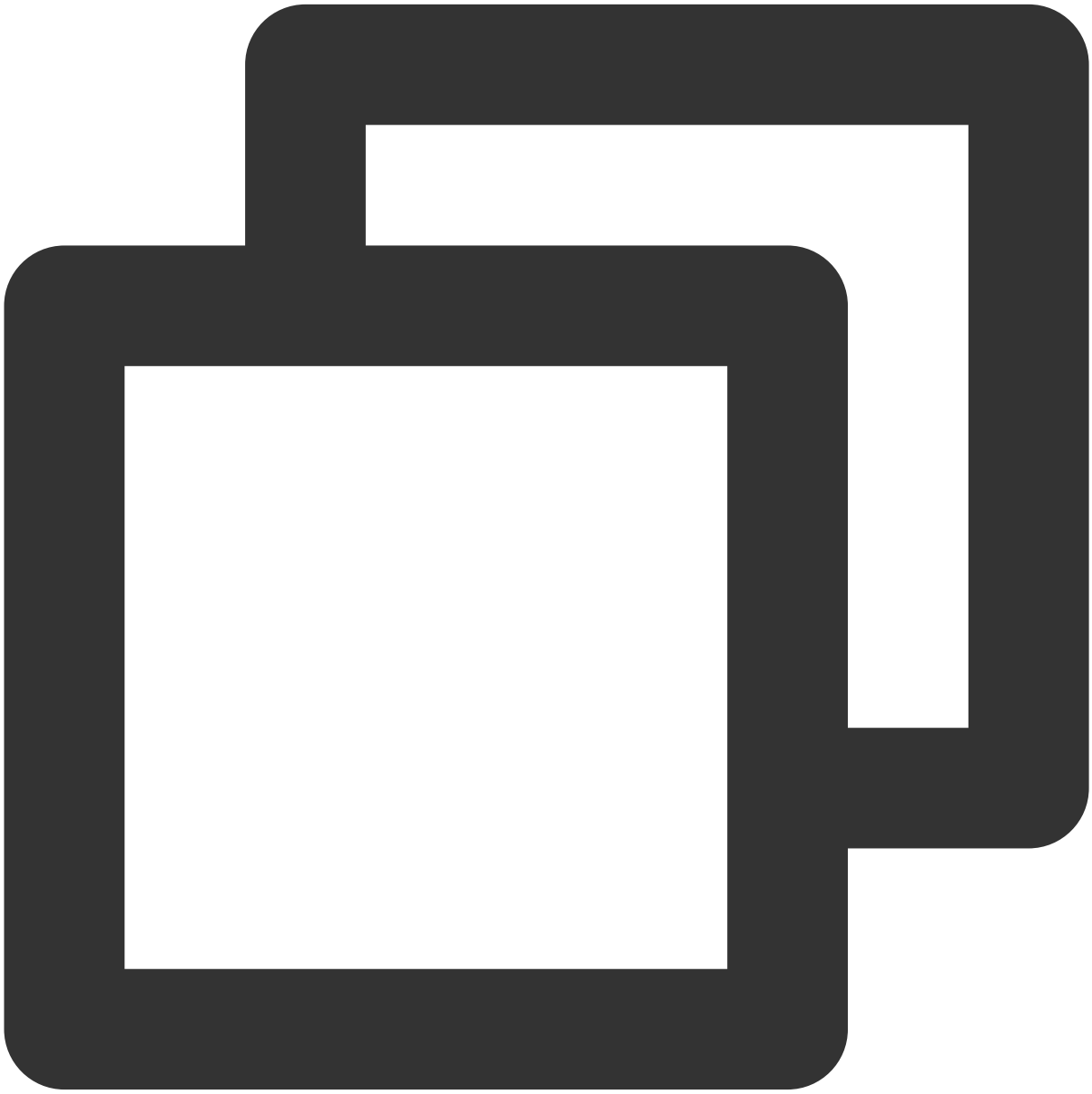


```
QCloudGetVideoRecognitionRequest * request = [[QCloudGetVideoRecognitionRequest all

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// QCloudPostVideoRecognitionRequest 接口返回的 jobid
request.jobId = @"jobid";
[request setFinishBlock:^(QCloudVideoRecognitionResult * _Nullable result, NSError
    // outputObject 审核结果 包含用于查询的 job id，详细字段请查看 api 文档或者 SDK 源码
```

```
    // QCloudVideoRecognitionResult 类；
}];
[[QCloudCOSXMLService defaultCOSXML] GetVideoRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。



```
let request : QCloudGetVideoRecognitionRequest = QCloudGetVideoRecognitionRequest()

// 存储桶名称，格式为 BucketName-APPID
```

```
request.bucket = "examplebucket-1250000000";

// QCloudPostVideoRecognitionRequest 接口返回的 jobid
request.jobId = "jobid";

request.finishBlock = { (result, error) in
    // outputObject 审核结果 包含用于查询的 job id，详细字段请查看 api 文档或者 SDK 源码
    // QCloudVideoRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().getVideoRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

音频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于音频审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
提交音频审核任务	用于提交一个音频审核任务
查询音频审核任务结果	用于查询指定的音频审核任务

创建任务

功能说明

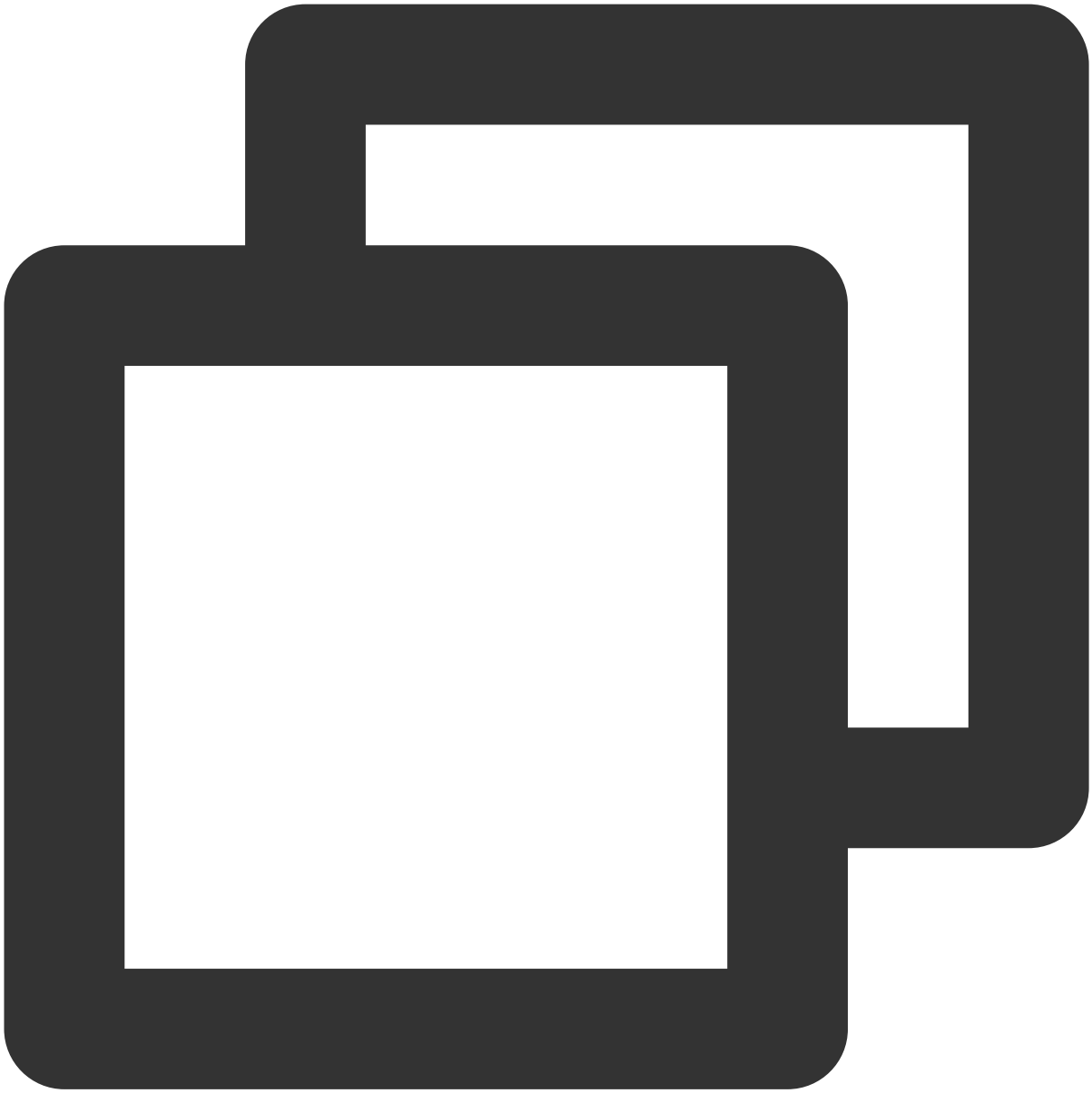
QCloudPostAudioRecognitionRequest 接口用于提交一个音频审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



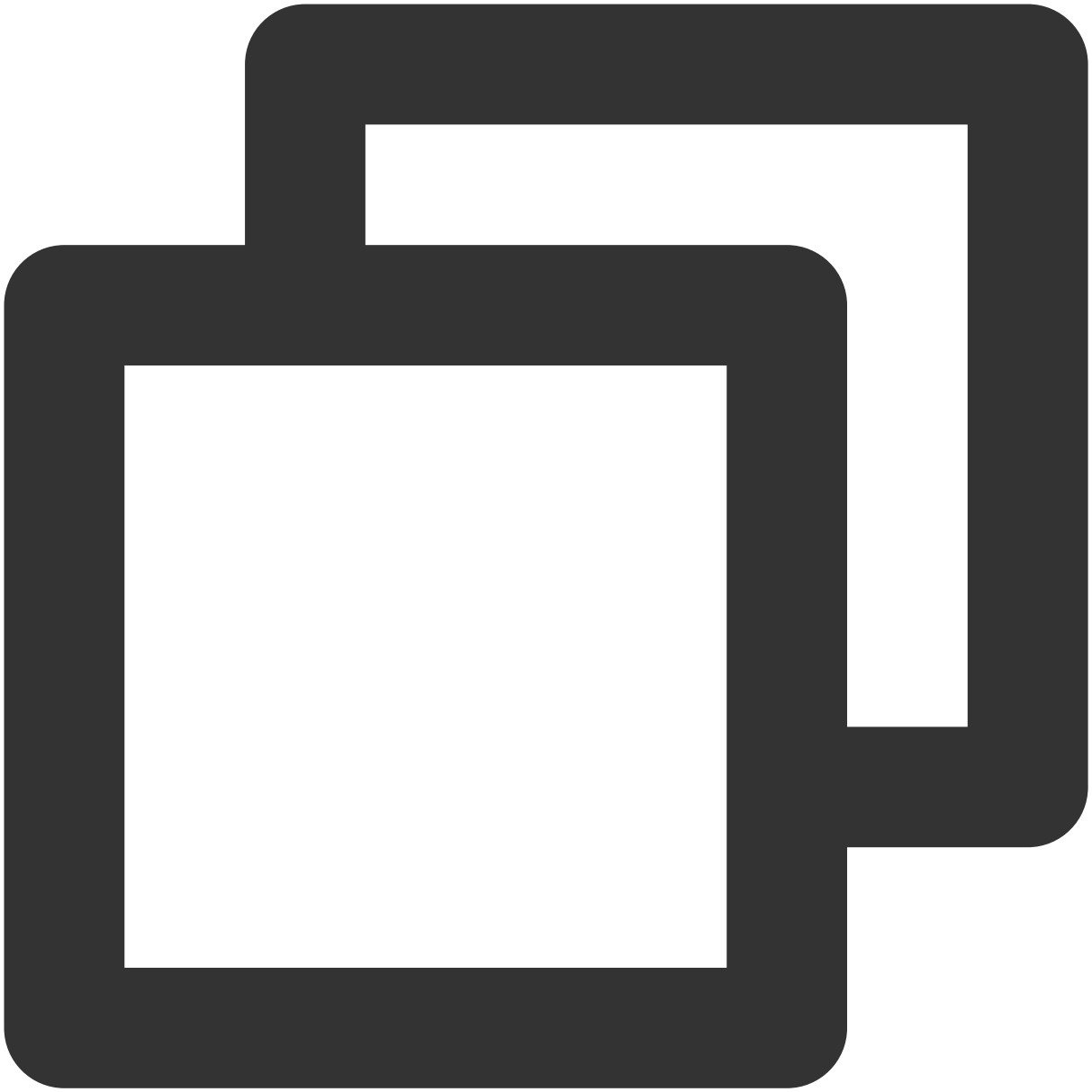
```
QCloudPostAudioRecognitionRequest * request = [[QCloudPostAudioRecognitionRequest a

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = @"exampleobject";

// 文件所在地域
request.regionName = @"regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";
```

```
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionAds;

request.bizType = BizType;

request.finishBlock = ^(QCloudPostAudioRecognitionResult * outputObject, NSError *e
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostAudioRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] PostAudioRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudPostAudioRecognitionRequest();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = "exampleobject";

// 文件所在地域
request.regionName = "regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";
```

```
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

request.bizType = BizType;

request.setFinish { outputObject, error in
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostAudioRecognitionResult 类；
};
QCloudCOSXMLService.defaultCOSXML().postAudioRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询任务

功能说明

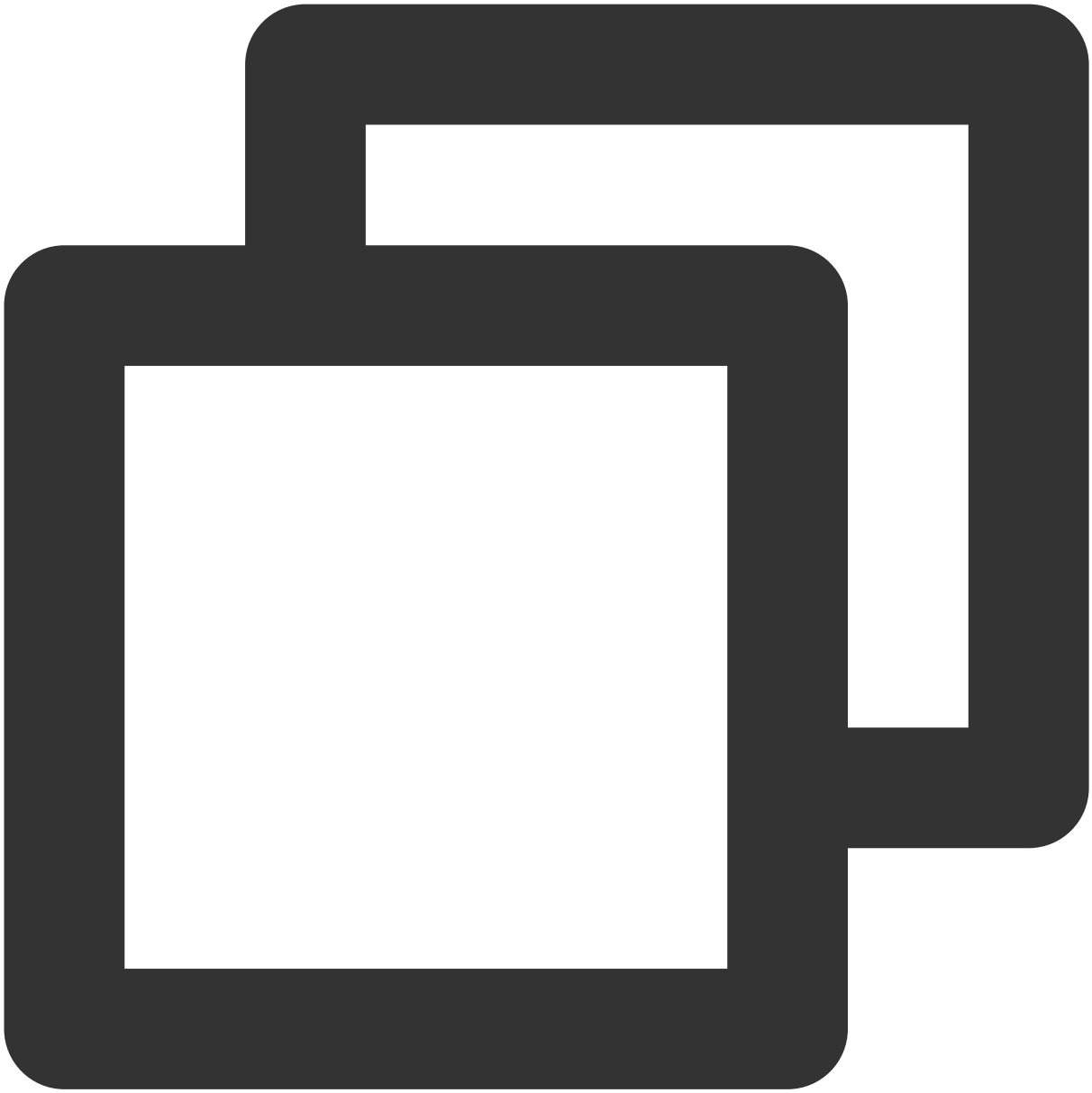
QCloudGetAudioRecognitionRequest 接口用于查询指定的音频审核任务。您可以根据音频审核任务的 JobId 来查询音频审核结果。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



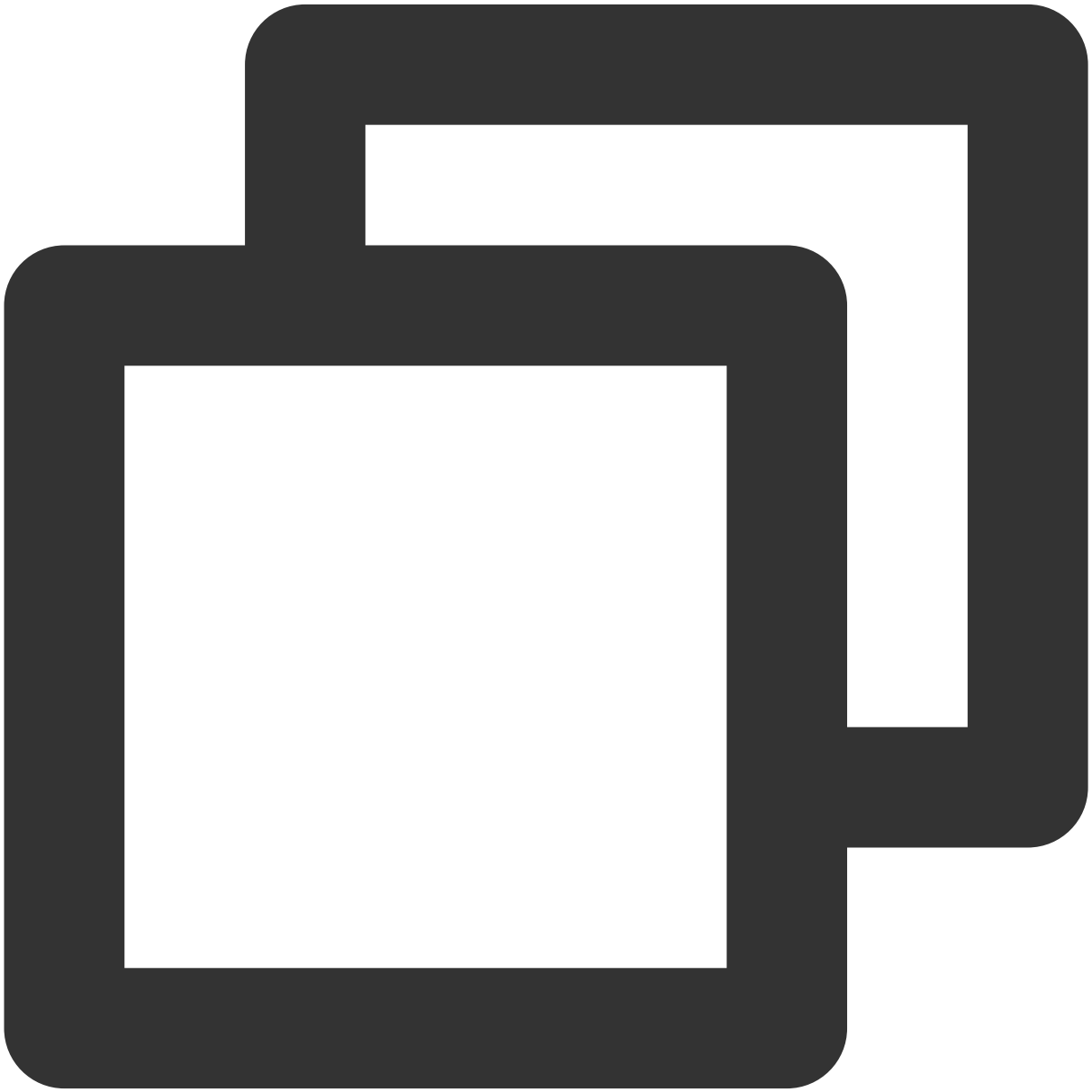
```
QCloudGetAudioRecognitionRequest * request = [[QCloudGetAudioRecognitionRequest all

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

// QCloudPostAudioRecognitionRequest接口返回的jobid
request.jobId = @"jobid";
```

```
request.finishBlock = ^(QCloudAudioRecognitionResult * outputObject, NSError *error
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudAudioRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetAudioRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudGetAudioRecognitionRequest();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// 文件所在地域
request.regionName = "regionName";

// QCloudPostAudioRecognitionRequest接口返回的jobid
request.jobId = "jobid";

request.setFinish { outputObject, error in
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudAudioRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().getAudioRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

文本审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文本审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
提交文本审核任务	用于提交一个文本审核任务
查询文本审核任务结果	用于查询指定的文本审核任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

创建任务

功能说明

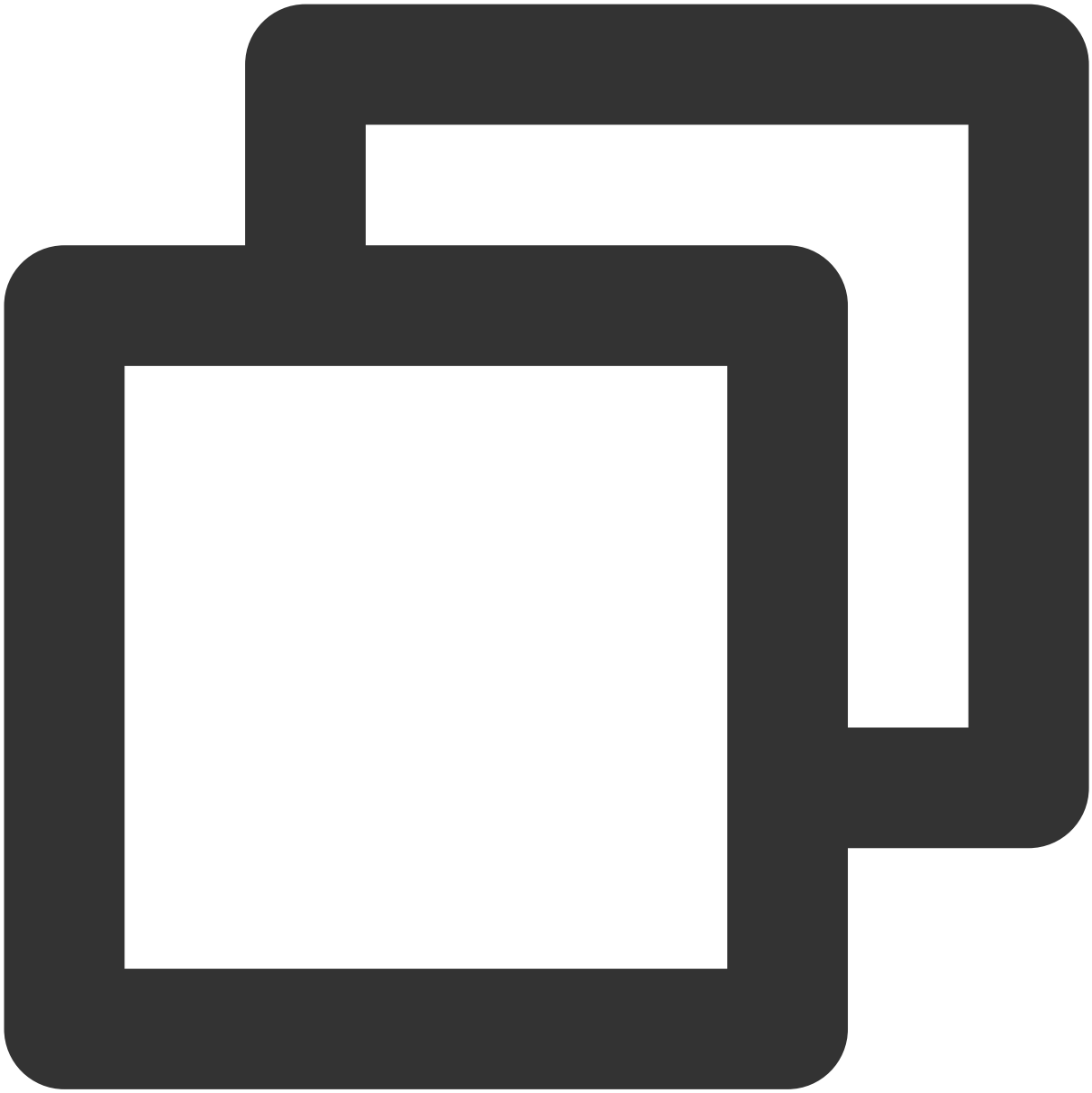
QCloudPostTextRecognitionRequest 接口用于提交一个文本审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



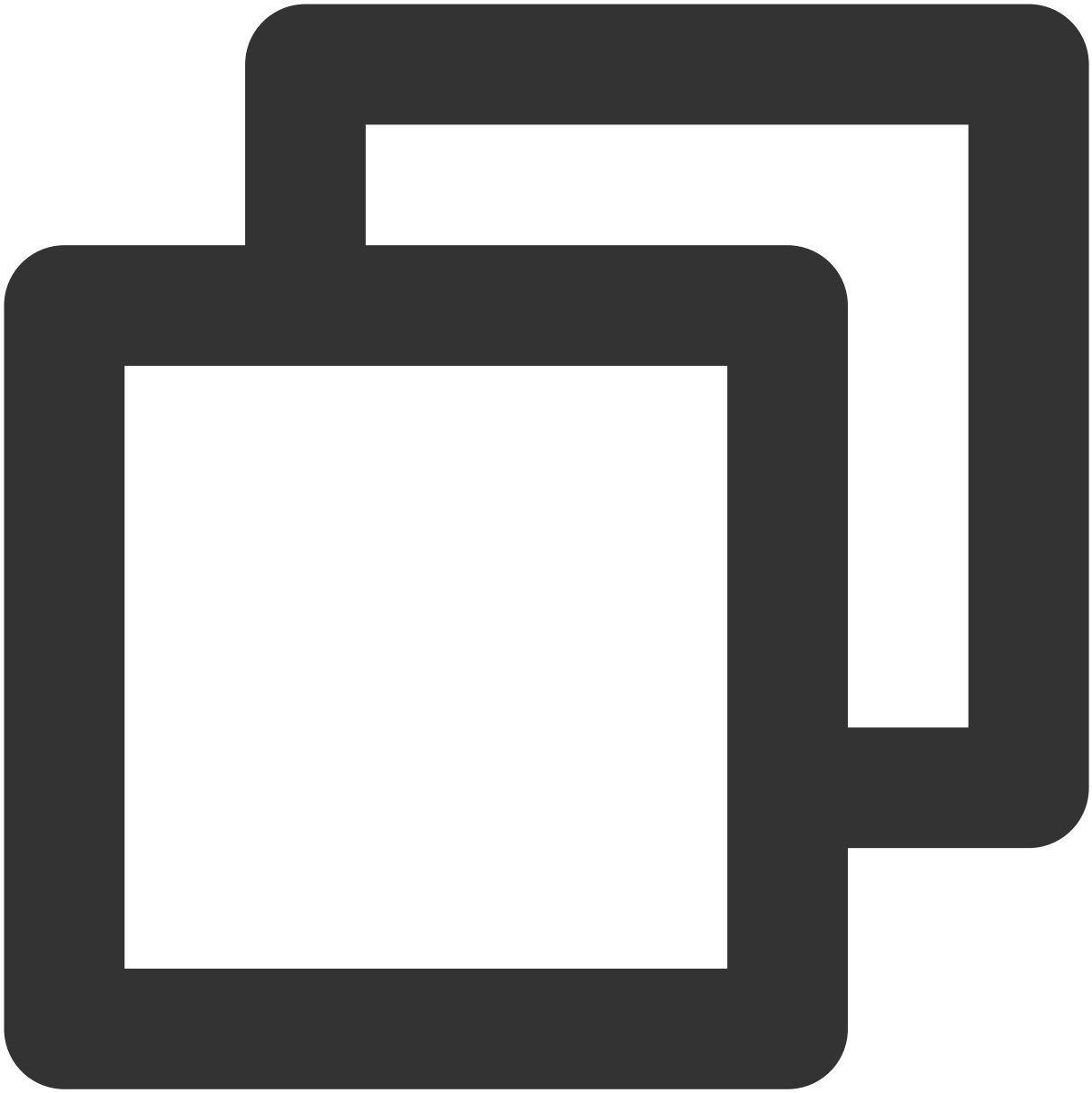
```
QCloudPostTextRecognitionRequest * request = [[QCloudPostTextRecognitionRequest all

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = @"exampleobject";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";
```

```
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionAds | QCloudRecogniti

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.bizType = BizType;

request.finishBlock = ^(QCloudPostTextRecognitionResult * outputObject, NSError *er
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostTextRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] PostTextRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudPostTextRecognitionRequest();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = "exampleobject";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// 文件所在地域
request.regionName = "regionName";
```

```
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.setFinish { outputObject, error in
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostTextRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().postTextRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询任务

功能说明

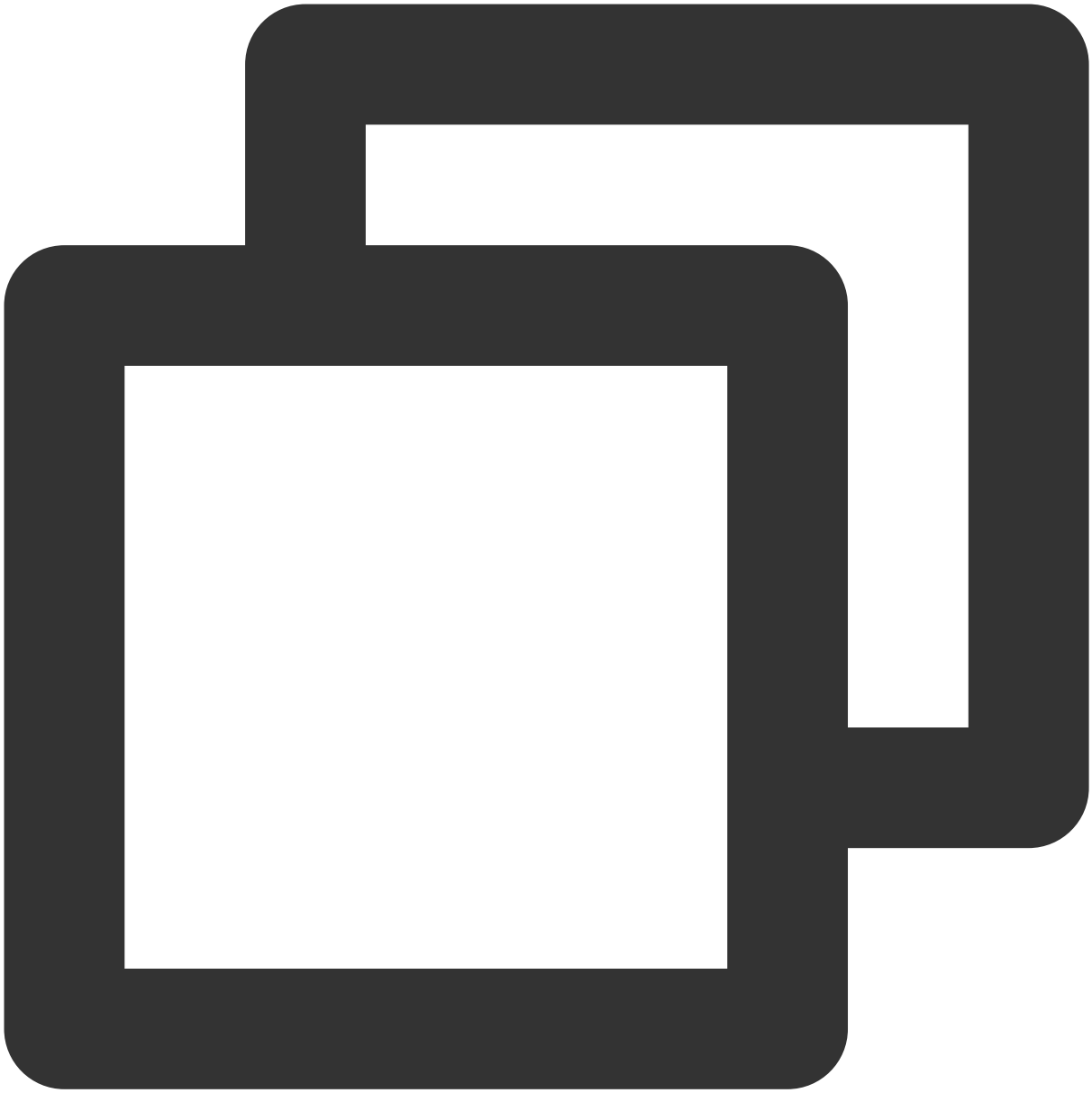
QCloudGetTextRecognitionRequest 接口用于查询指定的文本审核任务。您可以根据文本审核任务的 JobId 来查询文本审核结果。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



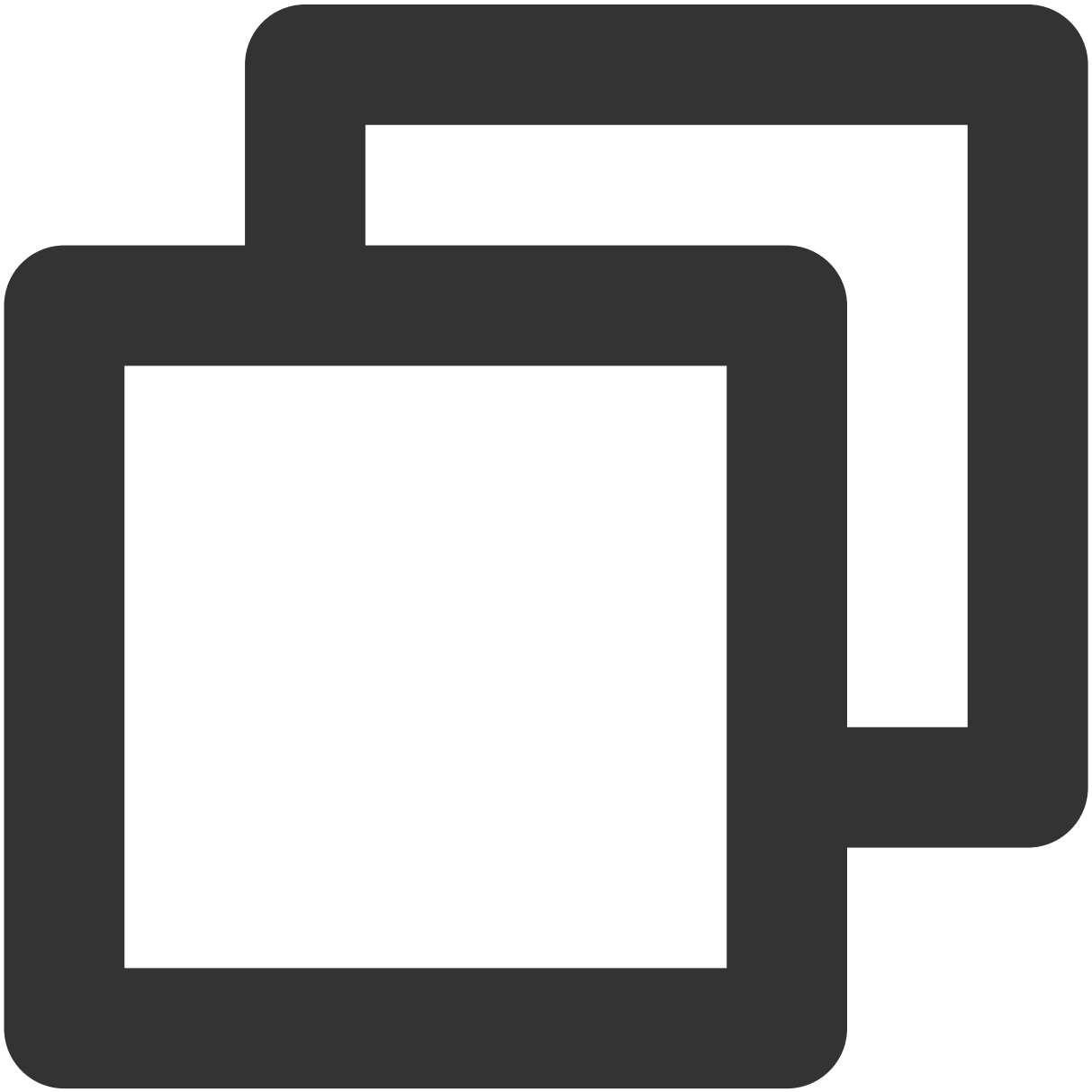
```
QCloudGetTextRecognitionRequest * request = [[QCloudGetTextRecognitionRequest alloc

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

// QCloudPostTextRecognitionRequest接口返回的jobid
request.jobId = @"jobid";
```

```
request.finishBlock = ^(QCloudTextRecognitionResult * outputObject, NSError *error)
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudTextRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetTextRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudGetTextRecognitionRequest();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// 文件所在地域
request.regionName = "regionName";

// QCloudPostTextRecognitionRequest接口返回的jobid
request.jobId = "jobid";

request.setFinish { outputObject, error in
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudTextRecognitionResult 类；
};
QCloudCOSXMLService.defaultCOSXML().getTextRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

文档审核

最近更新时间：2024-03-13 10:07:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文档审核的相关的 API 概览以及 SDK 示例代码。

API	操作描述
提交文档审核任务	用于提交一个文档审核任务
查询文档审核任务结果	用于查询指定的文档审核任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

创建任务

功能说明

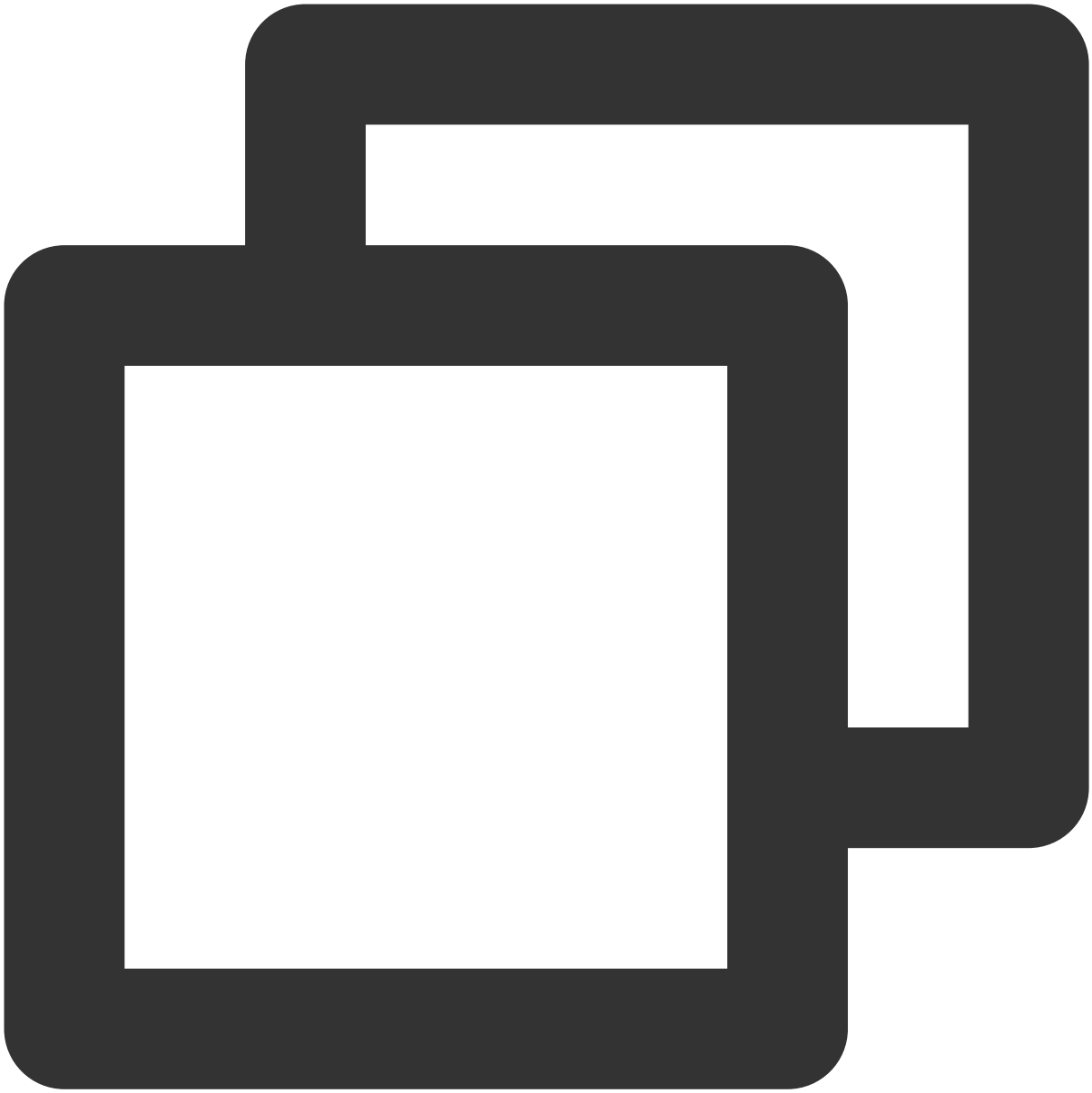
QCloudPostDocRecognitionRequest 接口用于提交一个文档审核任务。您可以通过主动设置回调地址接收审核信息，也可以通过 JobId 进行查询。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



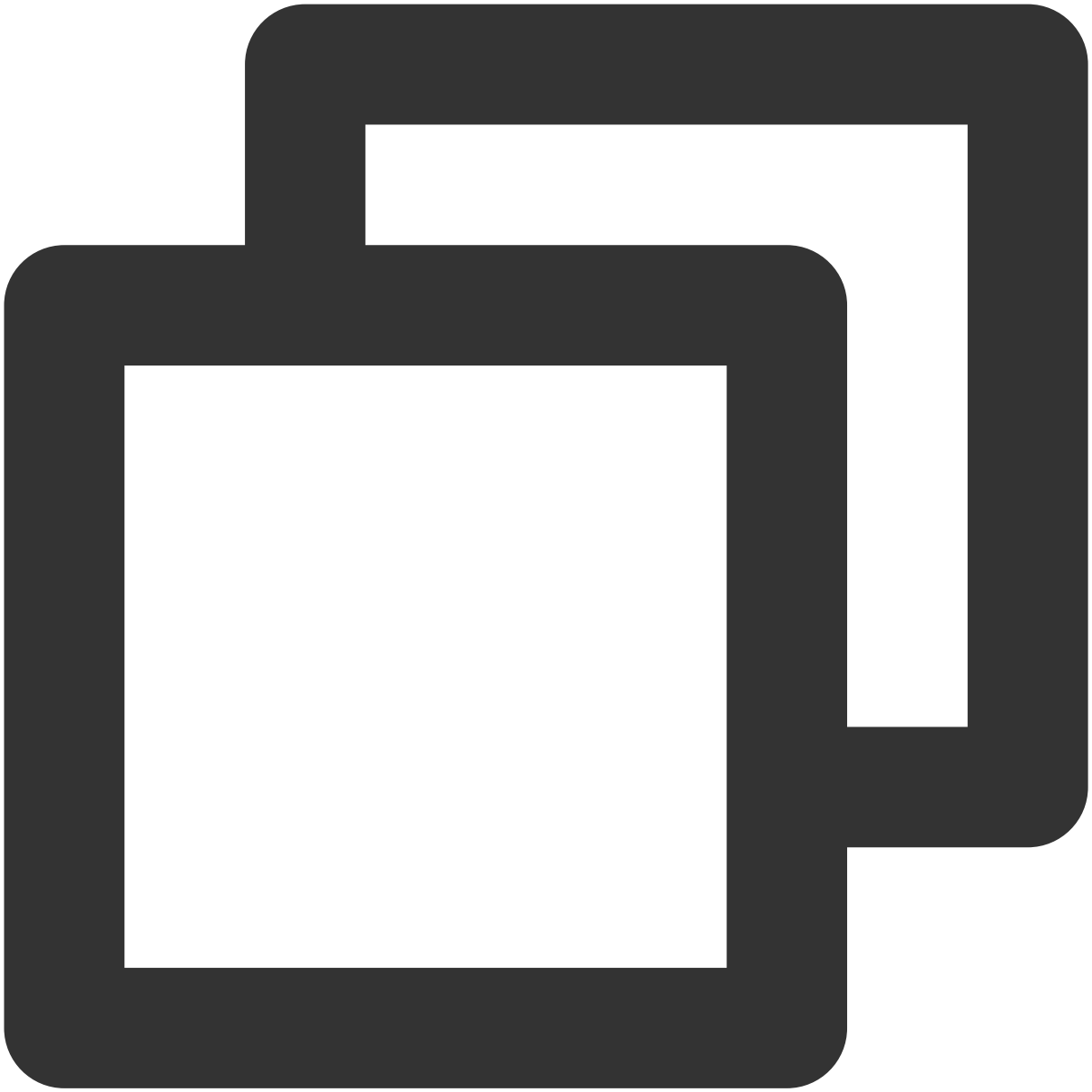
```
QCloudPostDocRecognitionRequest * request = [[QCloudPostDocRecognitionRequest alloc

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = @"exampleobject";

// 文件所在地域
request.regionName = @"regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";
```

```
request.type = @"doc";

// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionAds | QCloudRecogniti

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.bizType = BizType;

request.finishBlock = ^(QCloudPostDocRecognitionResult * outputObject, NSError *err
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostDocRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] PostDocRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudPostDocRecognitionRequest();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.object = "exampleobject";

// 文件所在地域
request.regionName = "regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";
```

```
request.type = "doc";

// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.setFinish { outputObject, error in
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostDocRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().postDocRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询任务

功能说明

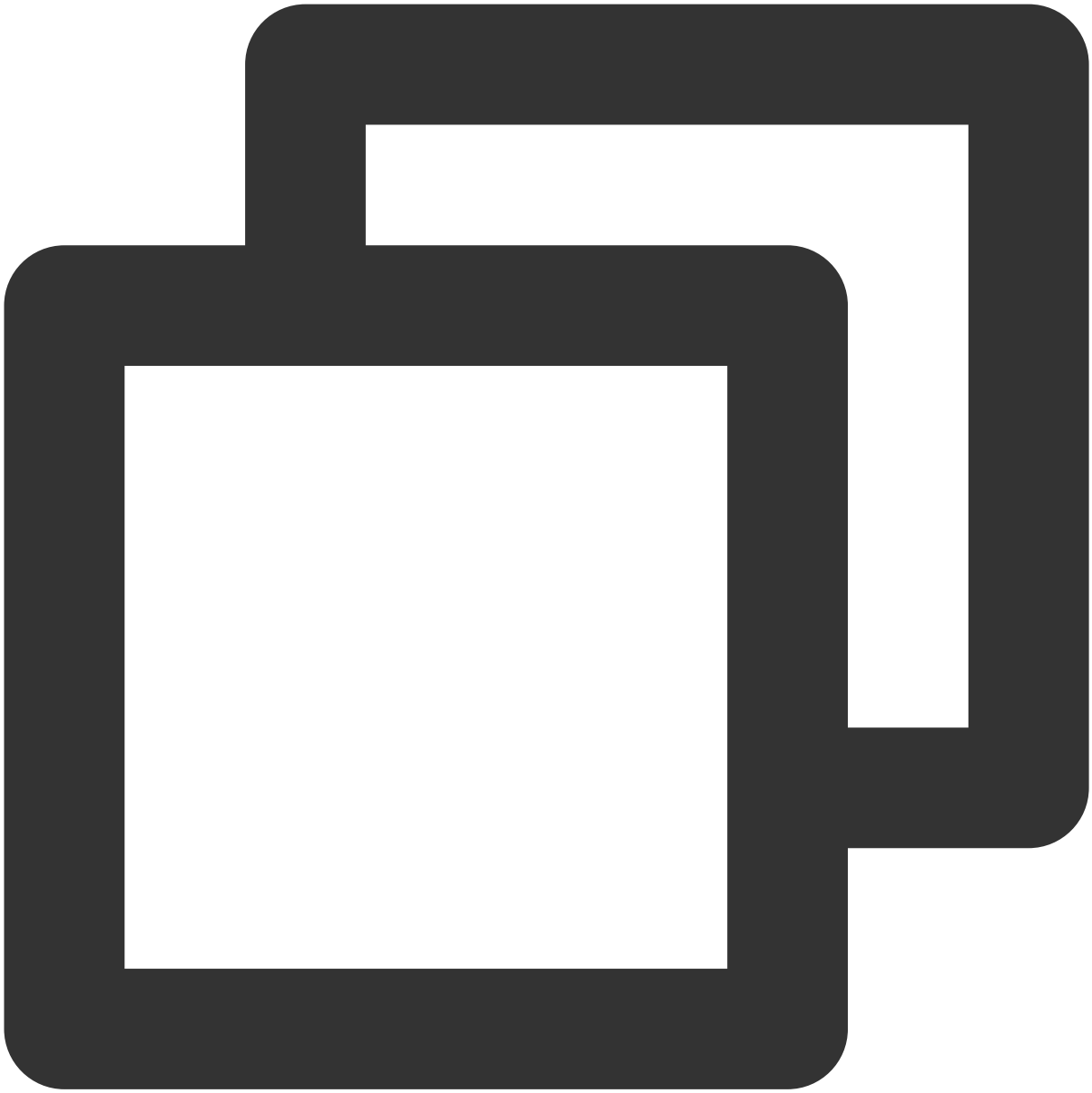
QCloudGetDocRecognitionRequest 接口用于查询指定的文档审核任务。您可以根据文档审核任务的 JobId 来查询文档审核结果。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



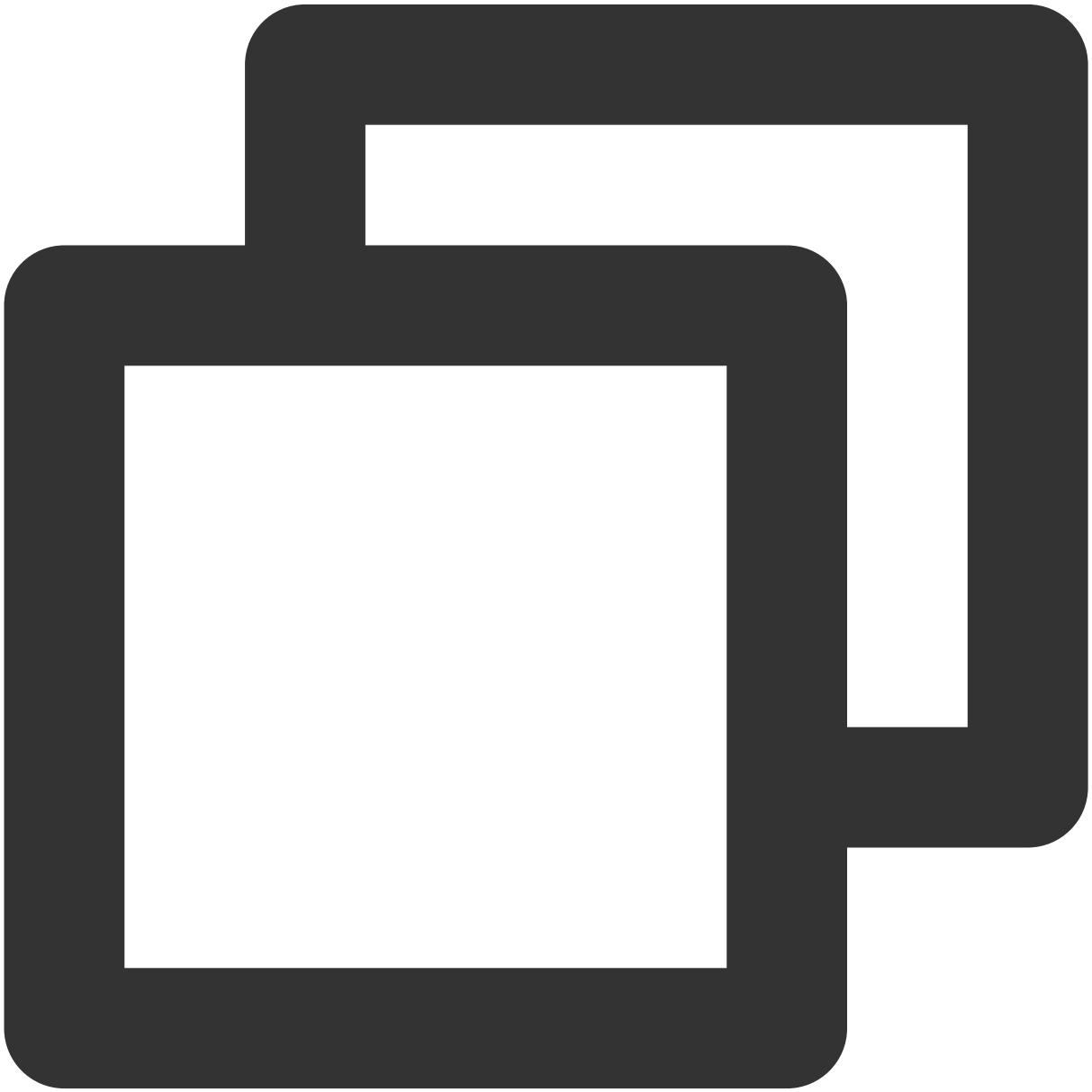
```
QCloudGetDocRecognitionRequest * request = [[QCloudGetDocRecognitionRequest alloc]i

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

// QCloudPostDocRecognitionRequest接口返回的jobid
request.jobId = @"jobid";
```

```
request.finishBlock = ^(QCloudDocRecognitionResult * outputObject, NSError *error)
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudDocRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetDocRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudGetDocRecognitionRequest();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// 文件所在地域
request.regionName = "regionName";

// QCloudPostDocRecognitionRequest接口返回的jobid
request.jobId = "jobid";

request.setFinish { outputObject, error in
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudDocRecognitionResult 类；
};
QCloudCOSXMLService.defaultCOSXML().getDocRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

网页审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于网页审核的 API 概览和 SDK 示例代码。

API	操作描述
提交网页审核任务	用于提交一个网页审核任务
查询网页审核任务结果	用于查询指定的网页审核任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

创建任务

功能说明

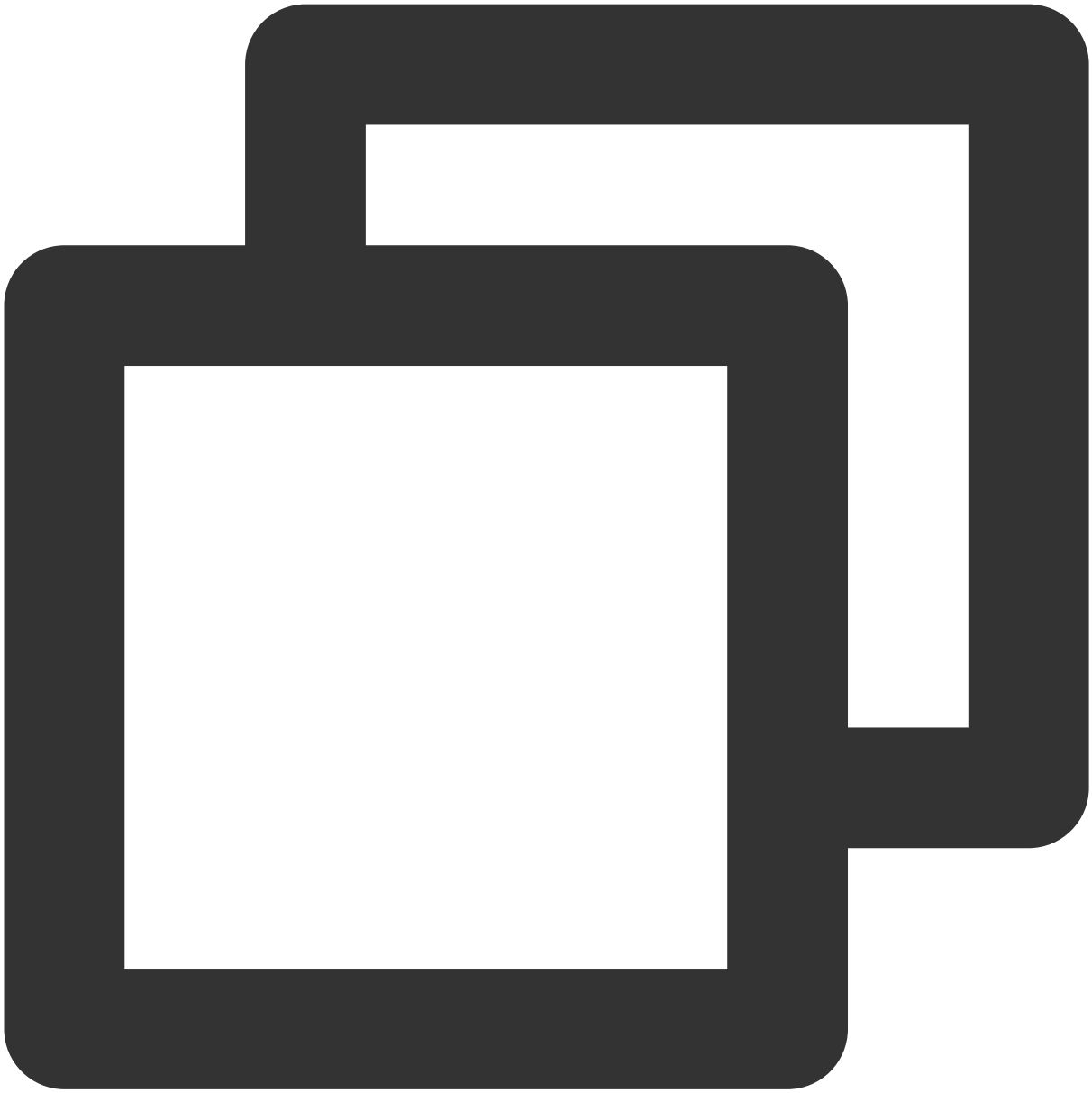
QCloudPostWebRecognitionRequest 接口用于提交一个网页审核任务。可以通过 JobId 进行查询。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



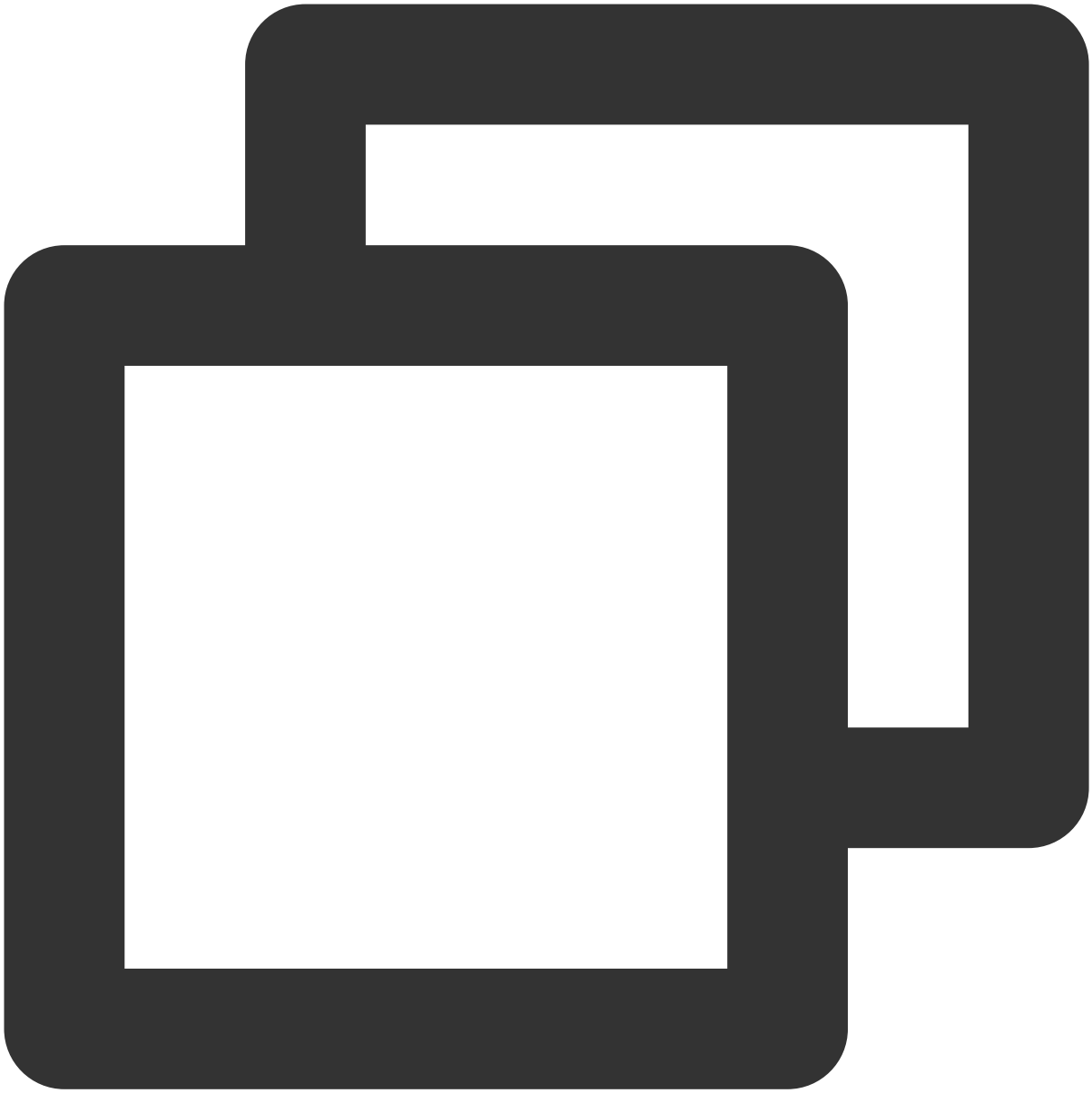
```
QCloudPostWebRecognitionRequest * request = [[QCloudPostWebRecognitionRequest alloc

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

request.url = @"www.****.com";
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
```

```
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionPorn | QCloudRecognitionAds | QCloudRecogniti

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.bizType = BizType;

request.finishBlock = ^(QCloudPostWebRecognitionResult * outputObject, NSError *err
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostTextRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] PostWebRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudPostWebRecognitionRequest();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
request.url = "www.***.com";

// 文件所在地域
request.regionName = "regionName";

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";
```

```
// 审核类型，拥有 porn（涉黄识别）、terrorist（涉暴恐识别）、politics（涉政识别）、ads（广告识
// 用户可选择多种识别类型，例如 detect-type=porn,ads 表示对图片进行涉黄及广告审核
// 可以使用或进行组合赋值 如： QCloudRecognitionPorn | QCloudRecognitionTerrorist
request.detectType = QCloudRecognitionEnum(rawValue: QCloudRecognitionEnum.porn.raw

// 审核策略，不带审核策略时使用默认策略。具体查看 https://cloud.tencent.com/document/produc
request.setFinish { outputObject, error in
    // outputObject 提交审核反馈信息 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudPostTextRecognitionResult 类；
}
QCloudCOSXMLService.defaultCOSXML().postWebRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询任务

功能说明

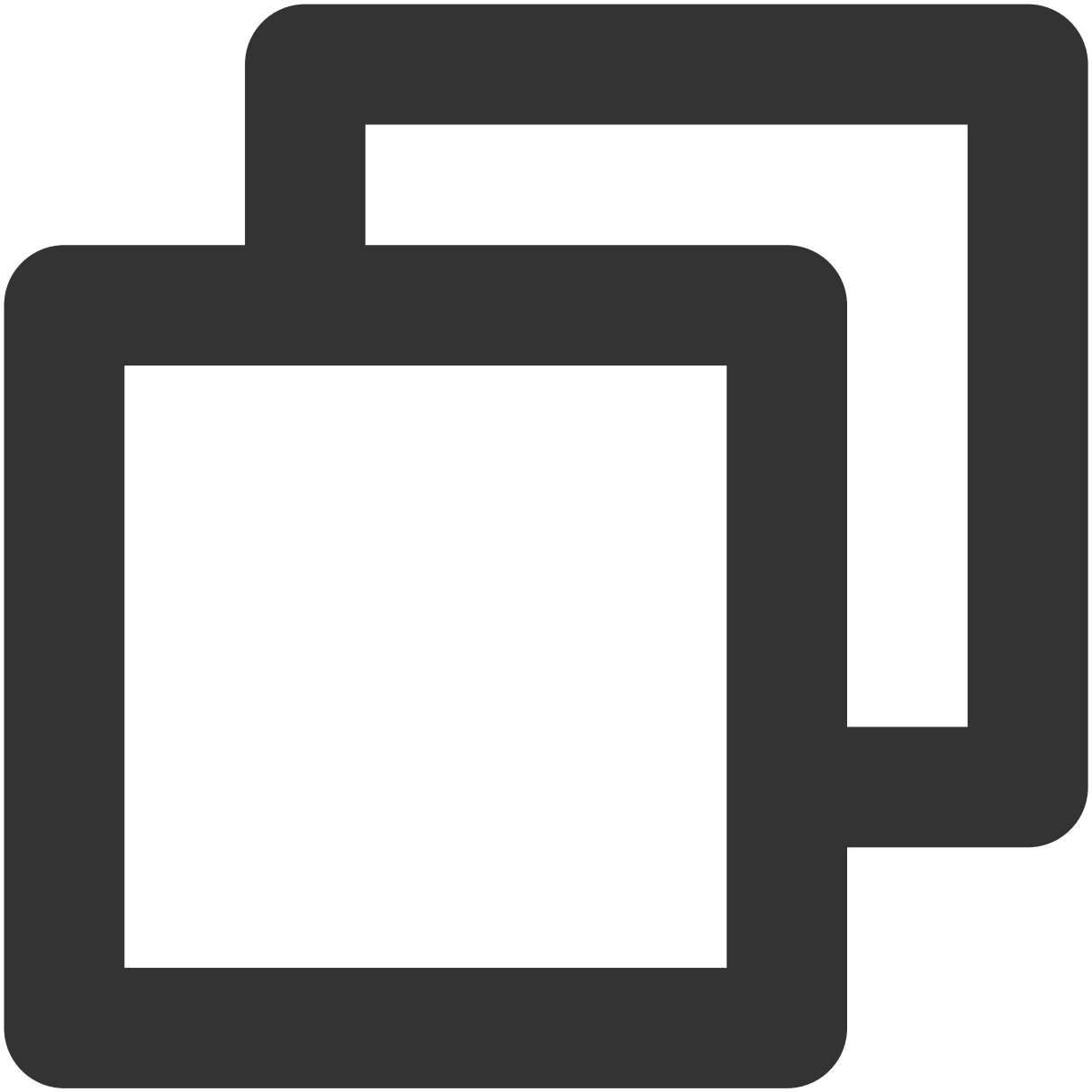
QCloudGetWebRecognitionRequest 接口用于查询指定的网页审核任务。您可以根据网页审核任务的 JobId 来查询网页审核结果。

注意：

COS iOS SDK 版本需要大于等于 v6.0.9。

请求示例

Objective-C



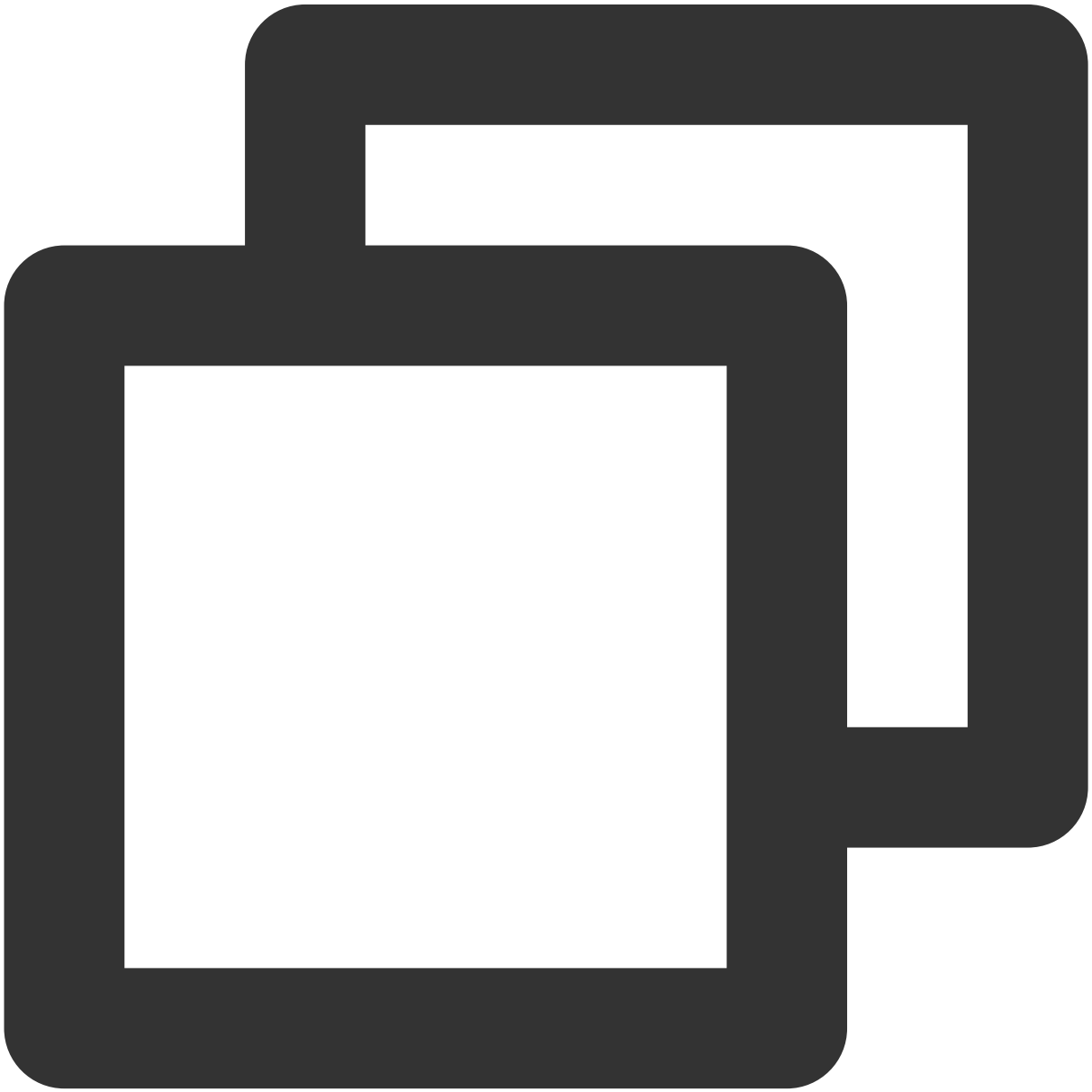
```
QCloudGetWebRecognitionRequest * request = [[QCloudGetWebRecognitionRequest alloc]i

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 文件所在地域
request.regionName = @"regionName";

// QCloudPostWebRecognitionRequest接口返回的jobid
request.jobId = @"jobid";
```

```
request.finishBlock = ^(QCloudWebRecognitionResult * outputObject, NSError *error)
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudWebRecognitionResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetWebRecognition:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift



```
let request = QCloudGetWebRecognitionRequest();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// 文件所在地域
request.regionName = "regionName";

// QCloudPostWebRecognitionRequest接口返回的jobid
request.jobId = "jobid";

request.setFinish { outputObject, error in
    // outputObject 审核结果 包含用于查询的job id，详细字段请查看api文档或者SDK源码
    // QCloudWebRecognitionResult 类；
};
QCloudCOSXMLService.defaultCOSXML().getWebRecognition(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

Java SDK
快速入门

最近更新时间：2024-03-13 10:07:10

简介

数据万象的媒体处理及文档预览接口集成至对象存储服务 XML Java SDK。

下载与安装

相关资源

对象存储服务的 XML Java SDK 源码下载地址：XML Java SDK。
SDK 快速下载地址：XML Java SDK。
示例 Demo 下载地址：COS XML Java SDK 示例。
SDK 文档中的所有示例代码请参见 SDK 代码示例。

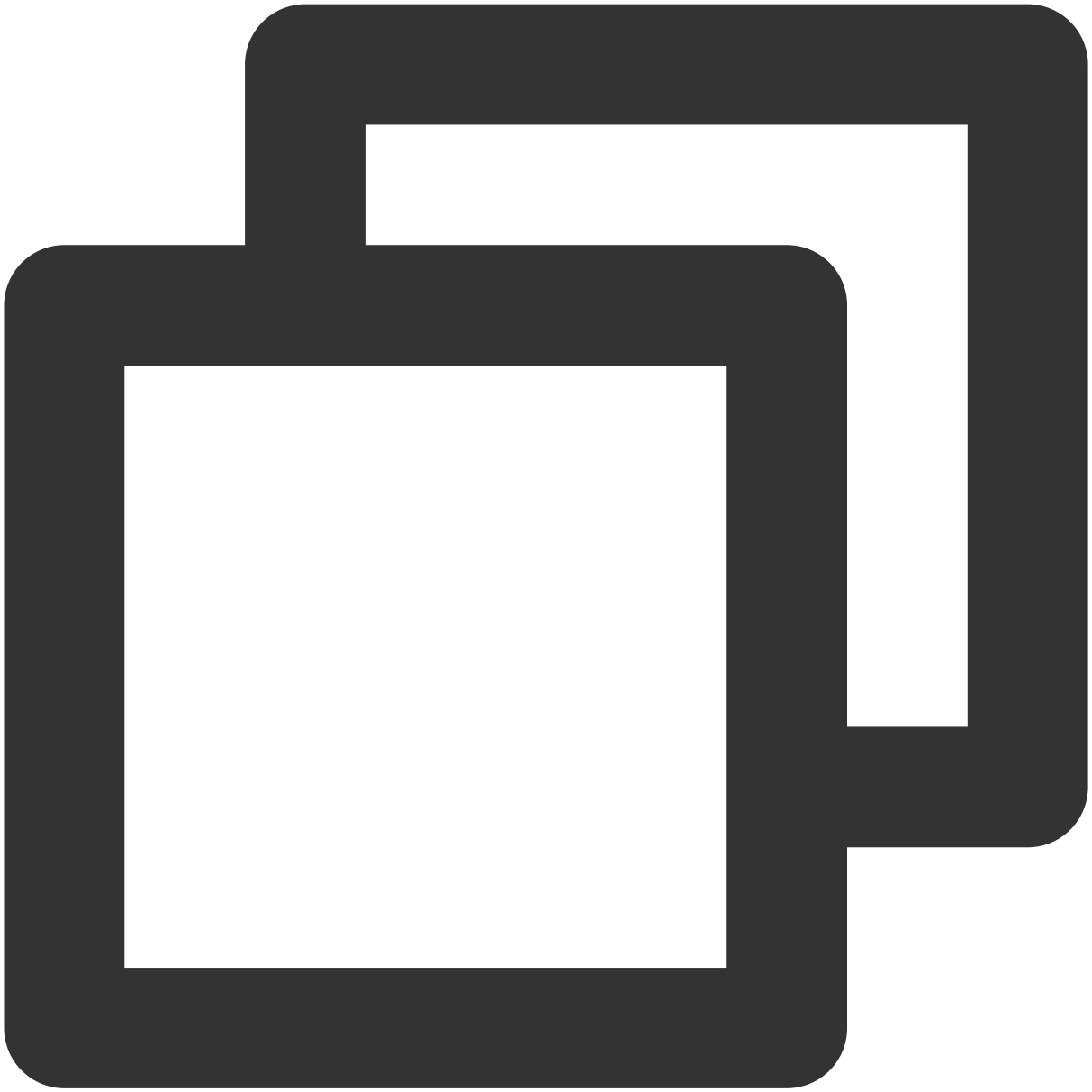
环境依赖

SDK 支持 JDK 1.8及以上版本。
JDK 安装方式请参见 Java 安装与配置。
说明
关于文章中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。
COS Java SDK 中的常见类所在包分别为：
客户端配置相关类在包 com.qcloud.cos.* 下。
权限相关类在 com.qcloud.cos.auth.* 子包下。
异常相关类在 com.qcloud.cos.exception.* 子包下。
请求相关类在 com.qcloud.cos.model.* 子包下。
地域相关类在 com.qcloud.cos.region.* 子包下。
高级 API 接口在 com.qcloud.cos.transfer.* 子包下。

安装 SDK

用户可以通过 maven 和源码两种方式安装 Java SDK：
maven 安装
在 maven 工程的 pom.xml 文件中添加相关依赖，内容如下：

版权所有：腾讯云计算（北京）有限责任公司



```
<dependency>
          <groupId>com.qcloud</groupId>
          <artifactId>cos_api</artifactId>
          <version>5.6.133</version>
</dependency>
```

说明

依赖坐标可能并非最新版本，请 单击此处 获取最新版本。

源码安装

从 Github XML Java SDK 或 快速下载地址 下载源码，通过 maven 导入。例如 eclipse，依次选择 **File > Import >**

maven > Existing Maven Projects。

卸载 SDK

通过删除 pom 依赖或源码即可卸载 SDK。

开始使用

下面为您介绍如何使用 Java SDK 完成一个基础操作，例如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。数据万象的媒体处理接口集成至对象存储服务 XML Java SDK，您可完成 模板操作、任务操作、工作流操作、队列操作 等相关操作。

导入类名

COS Java SDK 的包名为 `com.qcloud.cos.*`，您可以通过 Eclipse 或者 Intellij 等 IDE 工具，导入程序运行所需要的类。

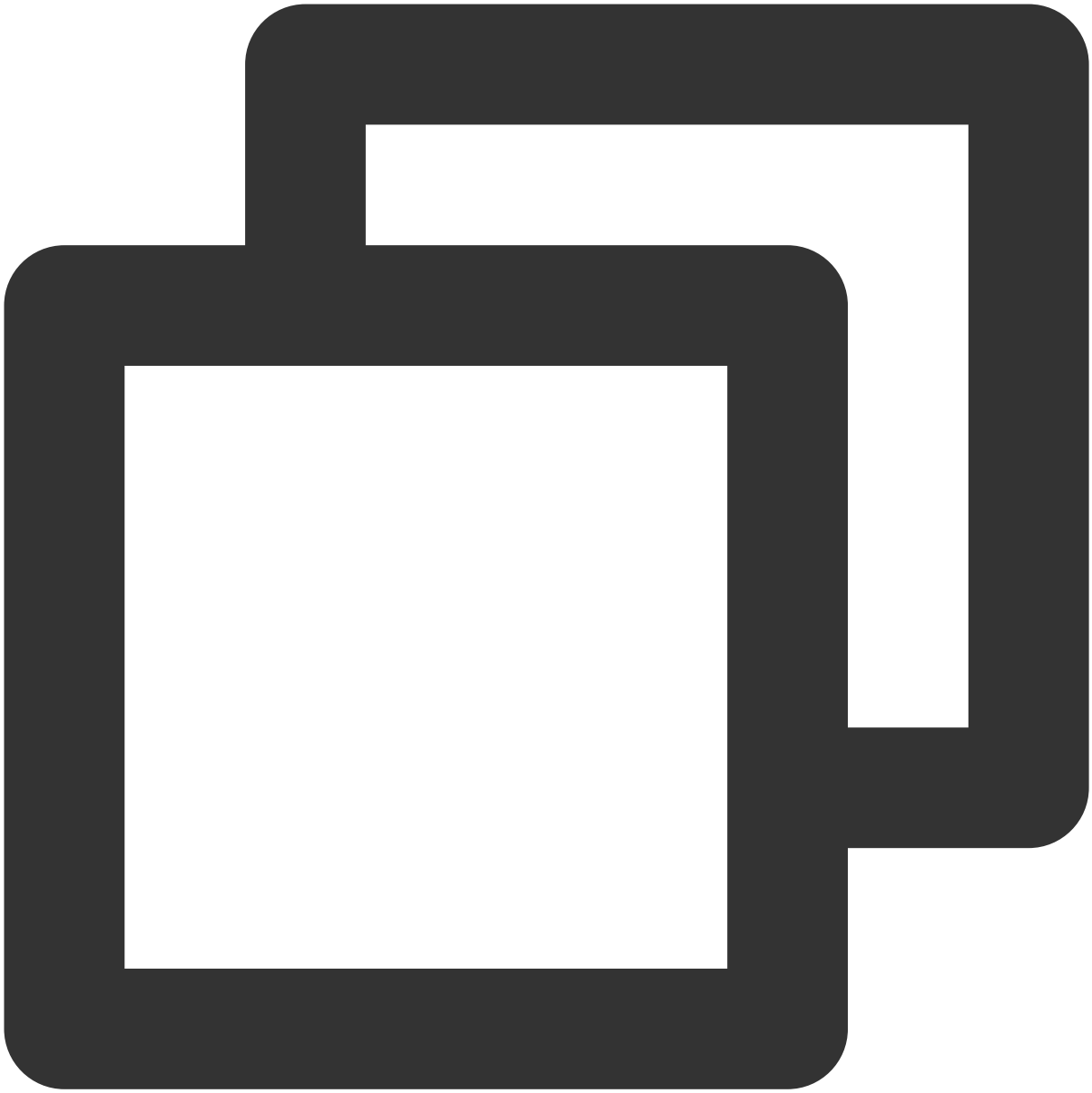
初始化客户端

在执行任何和 COS 服务相关请求之前，都需要先生成 COSClient 类的对象， COSClient 是调用 COS API 接口的对象。

注意

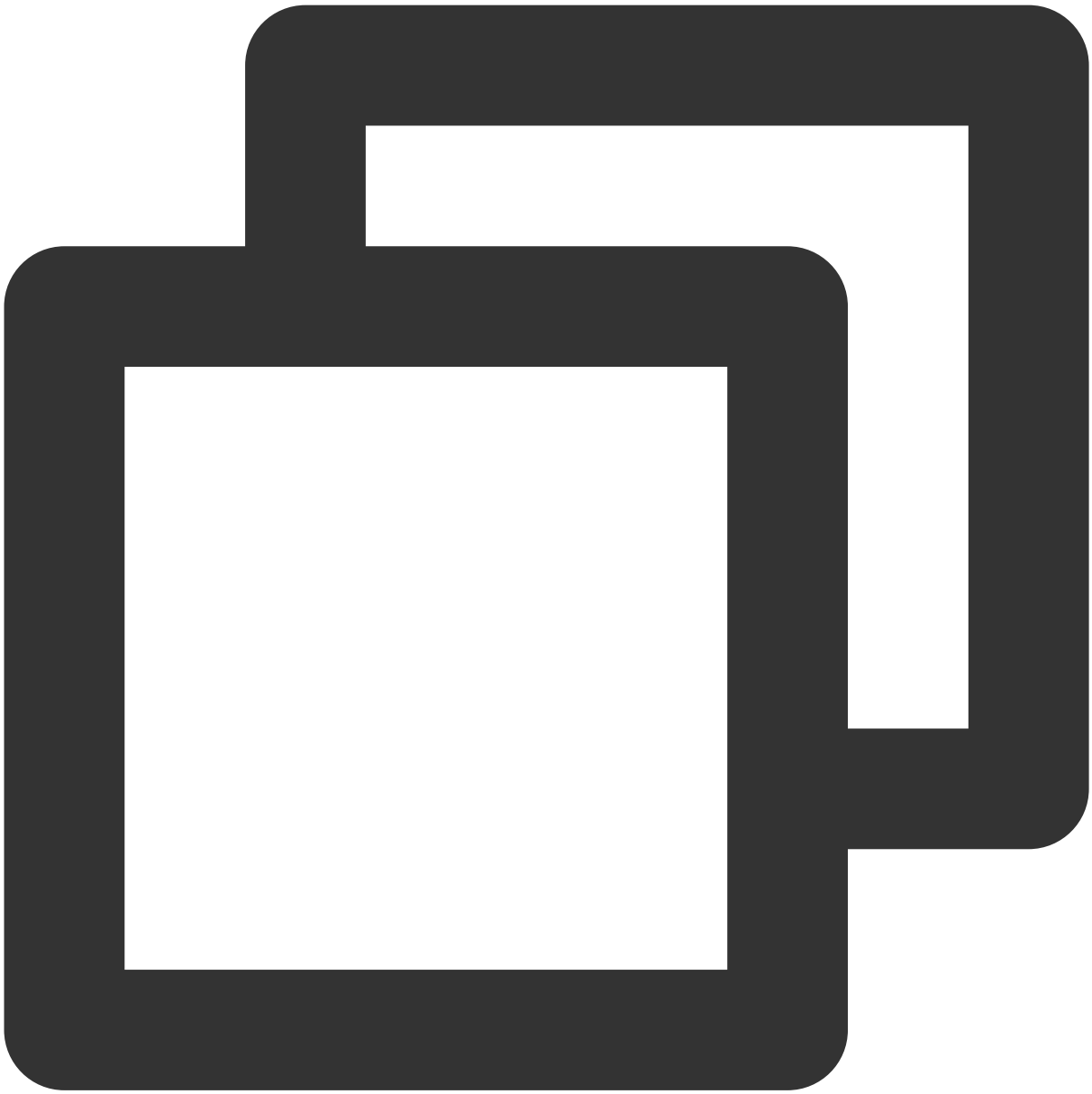
COSClient 是线程安全的类，允许多线程访问同一实例。因为实例内部维持了一个连接池，创建多个实例可能导致程序资源耗尽，**请确保程序生命周期内实例只有一个**，并在不再需要使用时，调用 shutdown 方法将其关闭。如果需要新建实例，请先将之前的实例关闭。

若您使用永久密钥初始化 COSClient，可以先在访问管理控制台中的 API 密钥管理 页面获取 SecretId、SecretKey，使用永久密钥适用于大部分的应用场景，参考示例如下：



```
// 1 初始化用户身份信息（secretId, secretKey）。
String secretId = "COS_SECRETID";
String secretKey = "COS_SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 2 设置 bucket 的区域, COS 地域的简称请参照 https://www.tencentcloud.com/document/proc
// 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
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// 3 生成 cos 客户端。
COSClient cosClient = new COSClient(cred, clientConfig);
```

您也可以使用临时密钥初始化 COSClient，临时密钥生成和使用可参见 临时密钥生成及使用指引，参考示例如下：



```
// 1 传入获取到的临时密钥 (tmpSecretId, tmpSecretKey, sessionToken)
String tmpSecretId = "COS_SECRETID";
String tmpSecretKey = "COS_SECRETKEY";
String sessionToken = "COS_TOKEN";
BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKe
// 2 设置 bucket 的区域, COS 地域的简称请参阅 https://www.tencentcloud.com/document/proc
// clientConfig 中包含了设置 region, https(默认 http), 超时, 代理等 set 方法, 使用可参阅源
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 3 生成 cos 客户端
COSClient cosClient = new COSClient(cred, clientConfig);
```

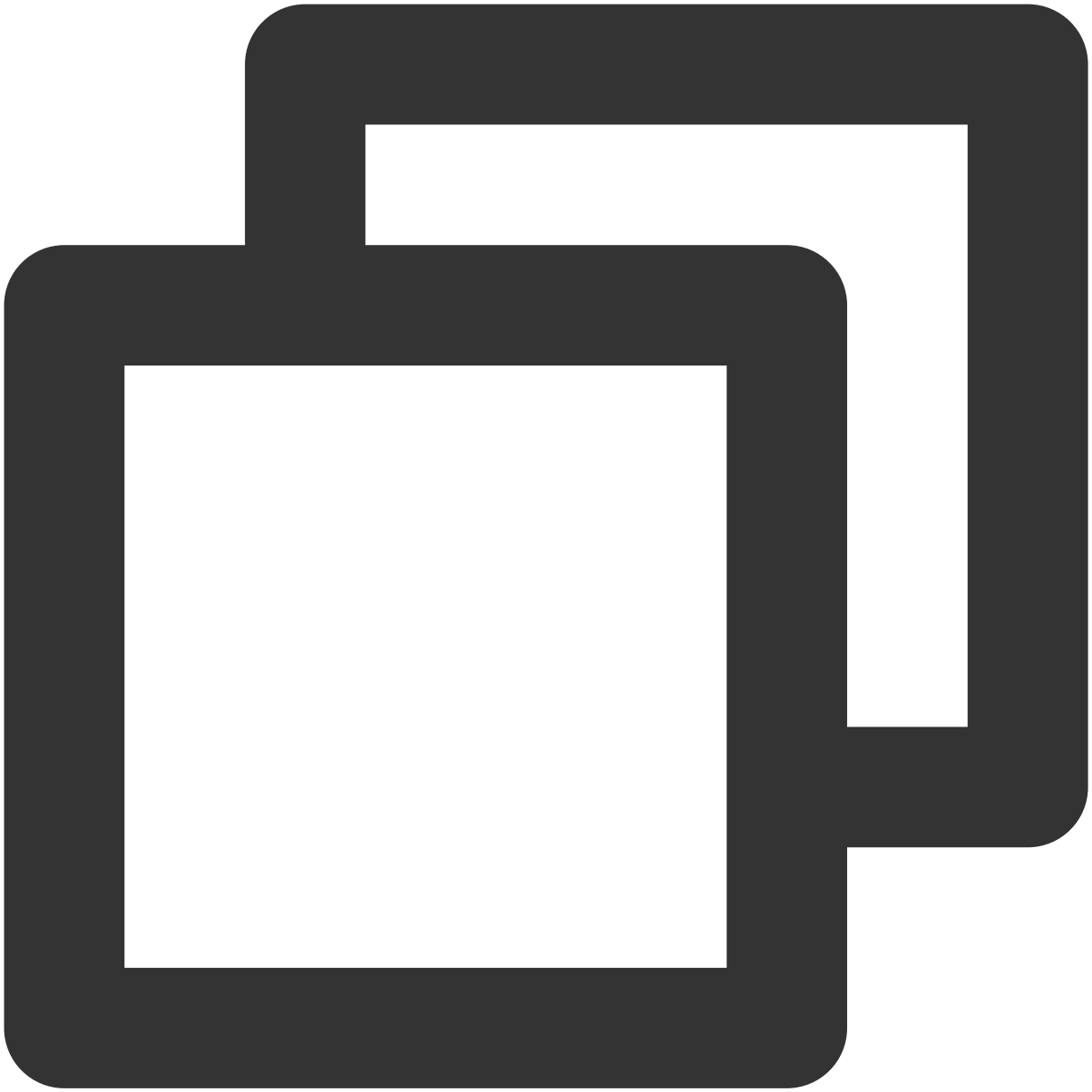
ClientConfig 类为配置信息类，主要的成员如下：

成员名	设置方法	描述	类型
region	构造函数或 set 方法	存储桶所在的区域，COS 地域的简称请参见 地域和访问域名 文档	Region
httpProtocol	set 方法	请求所使用的协议，默认使用 HTTP 协议与 COS 交互	HttpProtocol
signExpired	set 方法	请求签名的有效时间，单位：秒，默认为3600s	int
connectionTimeout	set 方法	连接 COS 服务的超时时间，单位：毫秒，默认为30000ms	int
socketTimeout	set 方法	客户端读取数据的超时时间，单位：毫秒，默认为30000ms	int
httpProxyIp	set 方法	代理服务器的 IP	String
httpProxyPort	set 方法	代理服务器的端口	int

查询开通媒体处理功能的桶列表

查询当前账号下已经开通媒体处理功能的桶列表。

请求示例

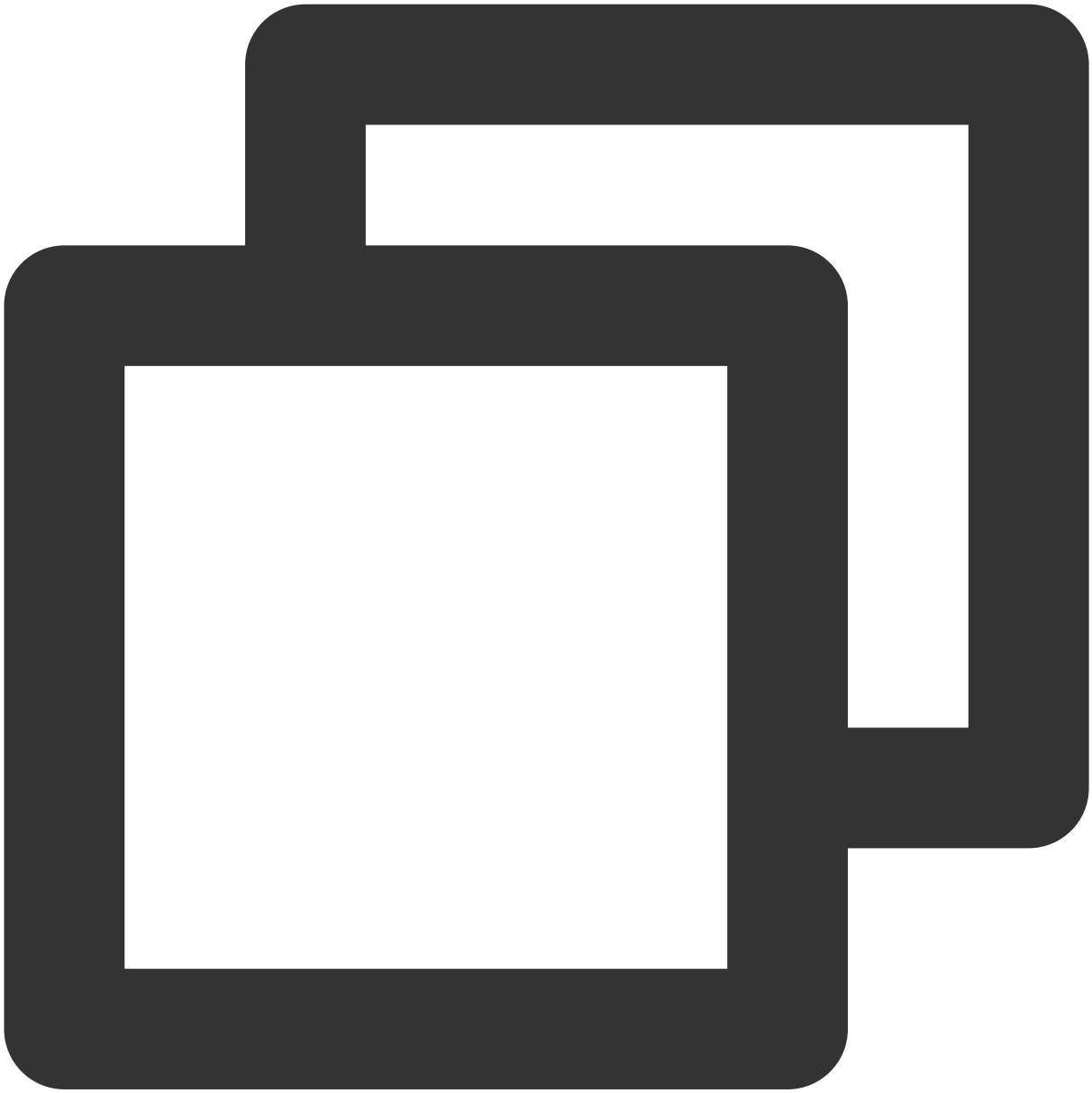


```
 //1.创建模板请求对象
MediaBucketRequest request = new MediaBucketRequest();
//2.添加请求参数 参数详情请见 API 接口文档
request.setBucketName("examplebucket-1250000000");
//3.调用接口,获取桶响应对象
MediaBucketResponse response = client.describeMediaBuckets(request);
```

创建任务

创建一个媒体处理任务。

请求示例



```
//1.创建任务请求对象
MediaJobsRequest request = new MediaJobsRequest();
//2.添加请求参数 参数详情请见 API 接口文档
request.setBucketName("examplebucket-1250000000");
request.setTag("Transcode");
request.getInput().setObject("1.mp4");
request.getOperation().setTemplateId("t0e09a9456d4124542b1f0e44d501*****");
request.getOperation().getOutput().setBucket("examplebucket-1250000000");
request.getOperation().getOutput().setRegion("ap-chongqing");
```

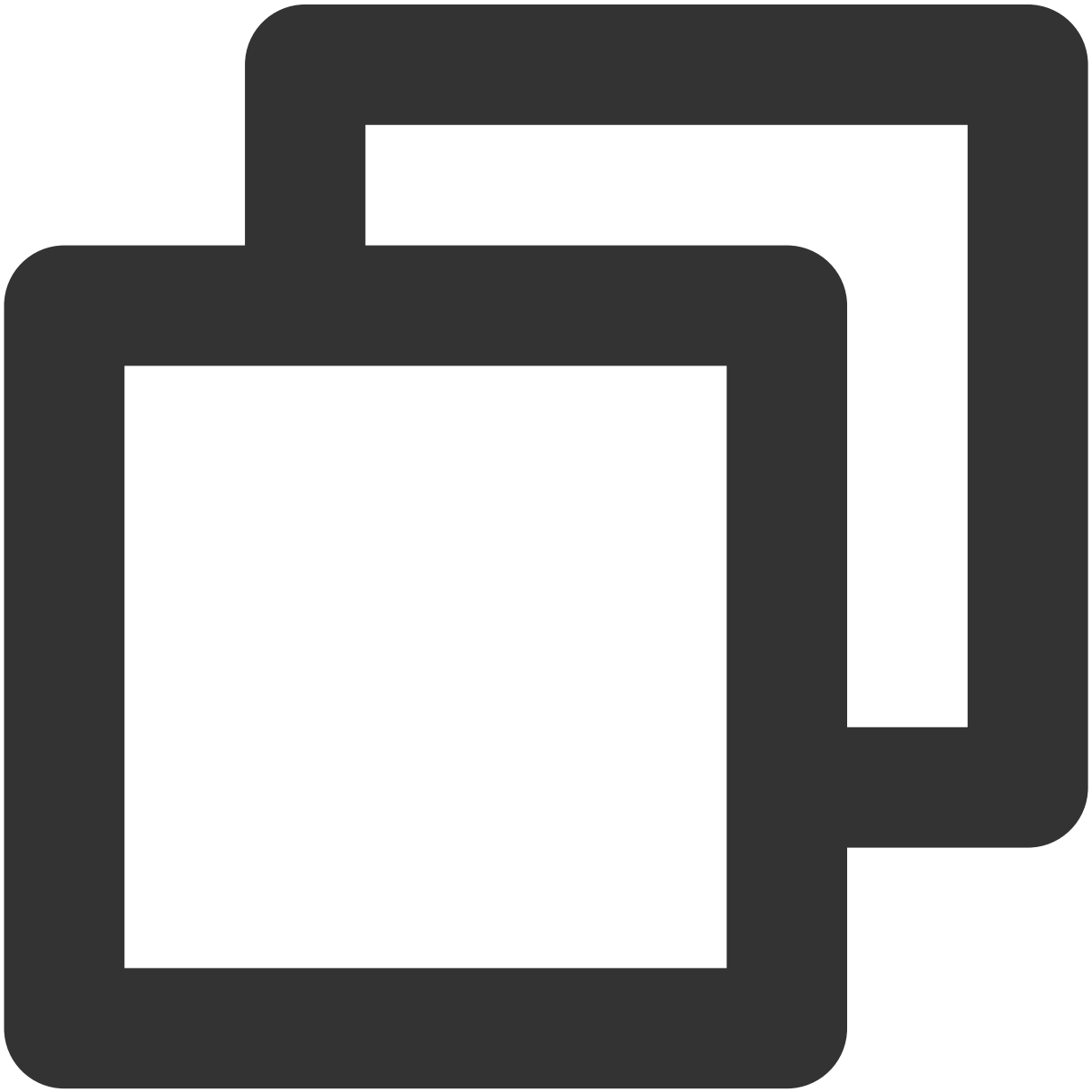
版权所有：腾讯云计算（北京）有限责任公司

```
request.getOperation().getOutput().setObject("2.mp4");
request.setQueueId("p9900025e4ec44b5e8225e70a521*****");
//3.调用接口,获取任务响应对象
MediaJobResponse response = client.createMediaJobs(request);
```

取消任务

功能说明

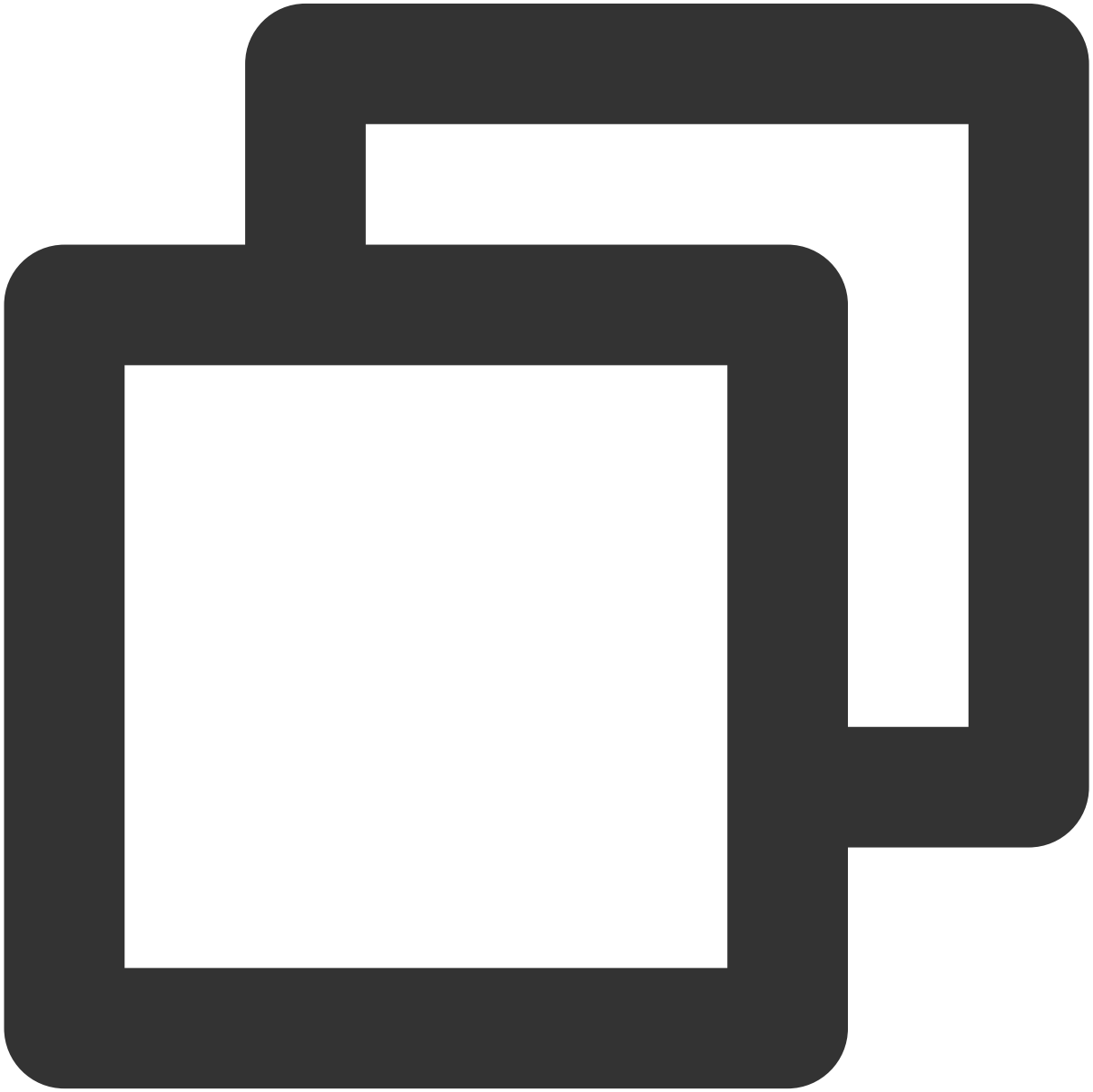
取消一个未在处理中的任务。



```
MediaJobsRequest request = new MediaJobsRequest();
request.setBucketName("examplebucket-1250000000");
request.setJobId("jae776cb4ec3011eab2cdd3817d4*****");
Boolean response = client.cancelMediaJob(request);
```

查询任务

根据任务 id 查询任务详情。



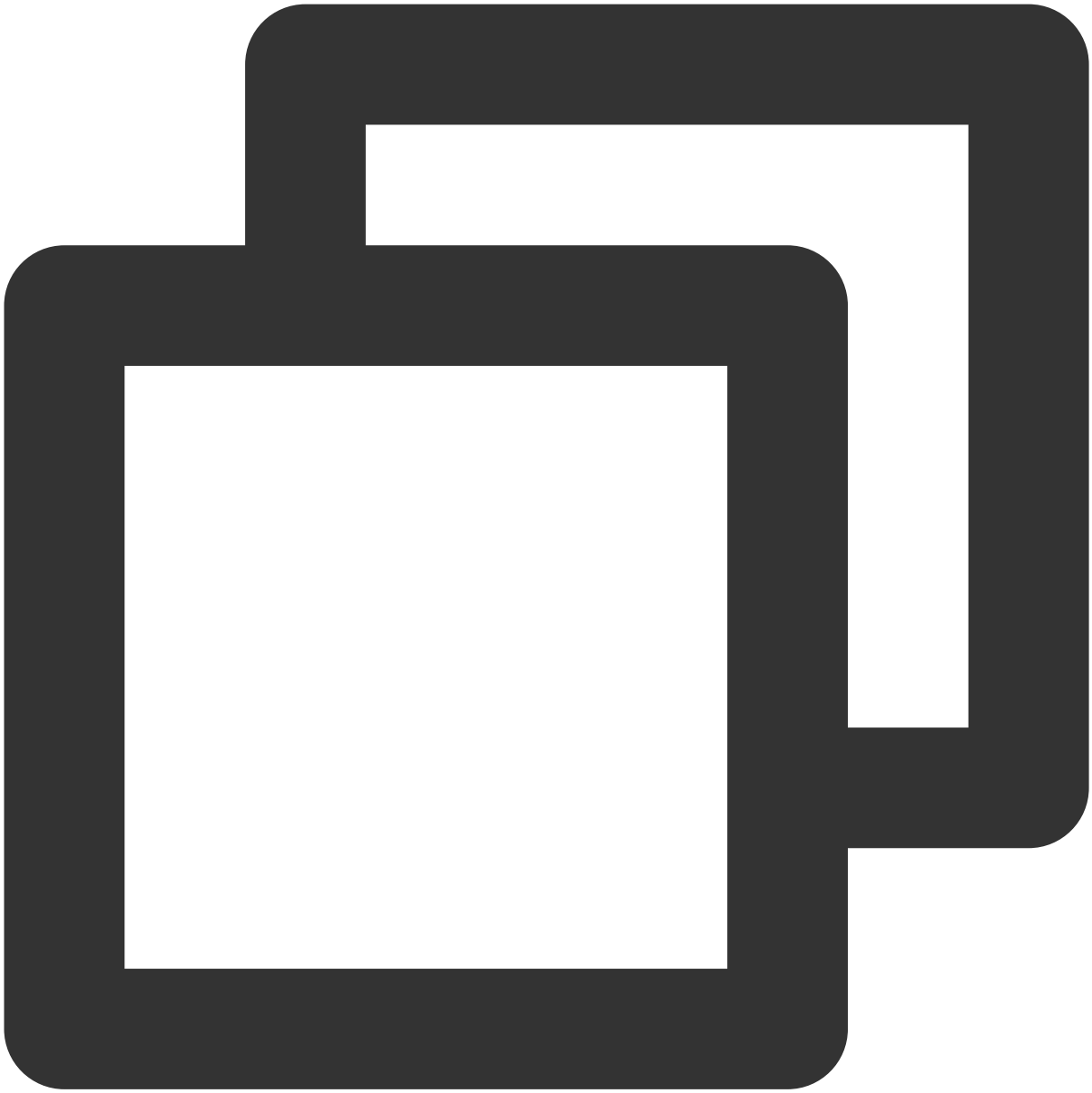
```
//1.创建任务请求对象
```

```
MediaJobsRequest request = new MediaJobsRequest();
//2.添加请求参数 参数详情请见 API 接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("j29a82fea08ba11ebb54bc9d1c05*****");
//3.调用接口,获取任务响应对象
MediaJobResponse response = client.describeMediaJob(request);
```

查询任务列表

查询队列中的任务列表。

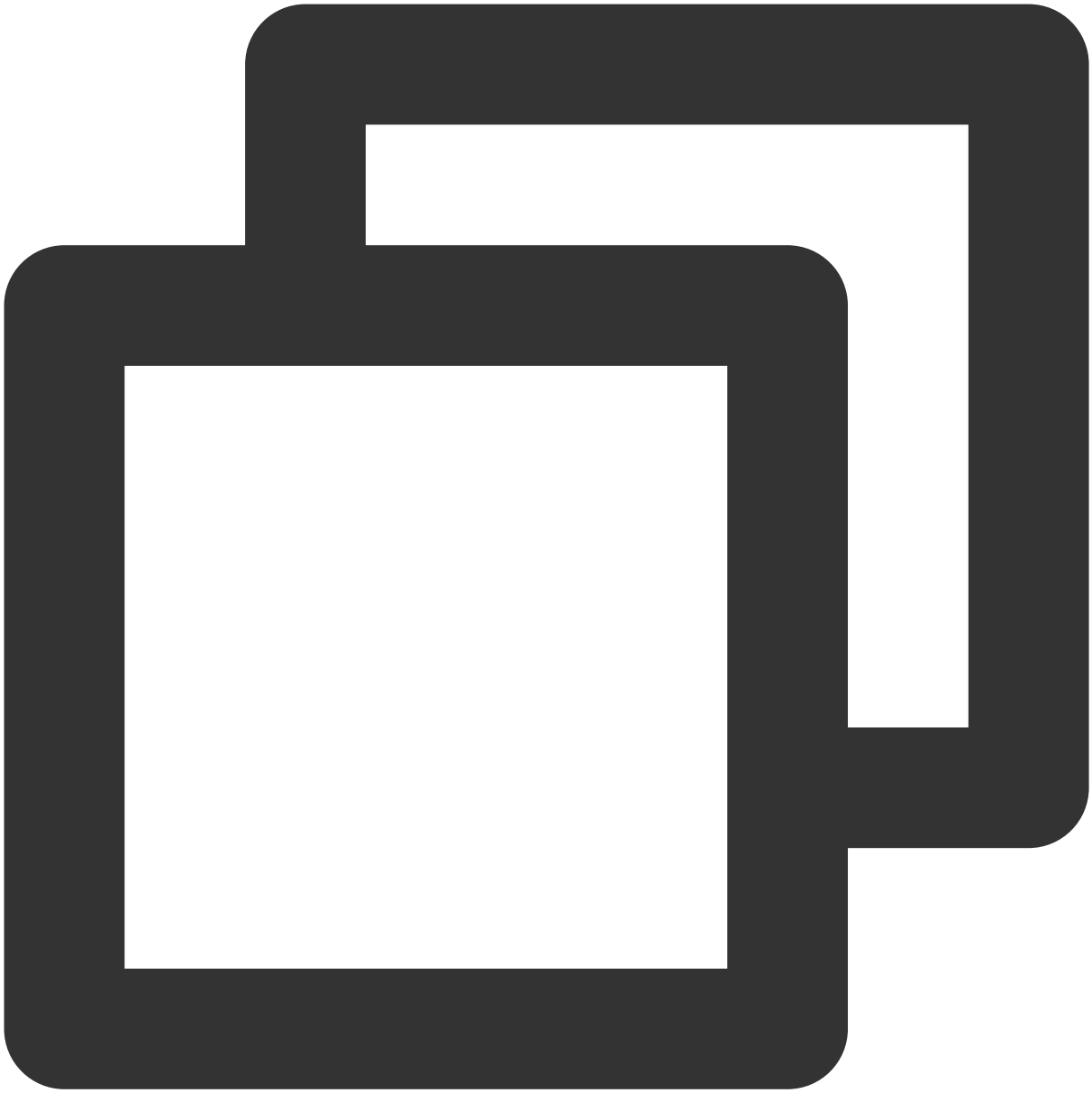
请求示例



```
MediaJobsRequest request = new MediaJobsRequest();
request.setBucketName("examplebucket-1250000000");
request.setQueueId("p9900025e4ec44b5e8225e70a521*****");
request.setTag("Transcode");
MediaListJobResponse response = client.describeMediaJobs(request);
List<MediaJobObject> jobsDetail = response.getJobsDetail();
```

关闭客户端

关闭 cosClient，并释放 HTTP 连接的后台管理线程，代码如下：



```
// 关闭客户端(关闭后台线程)
cosClient.shutdown();
```

常见问题

最近更新时间：2024-03-13 10:07:10

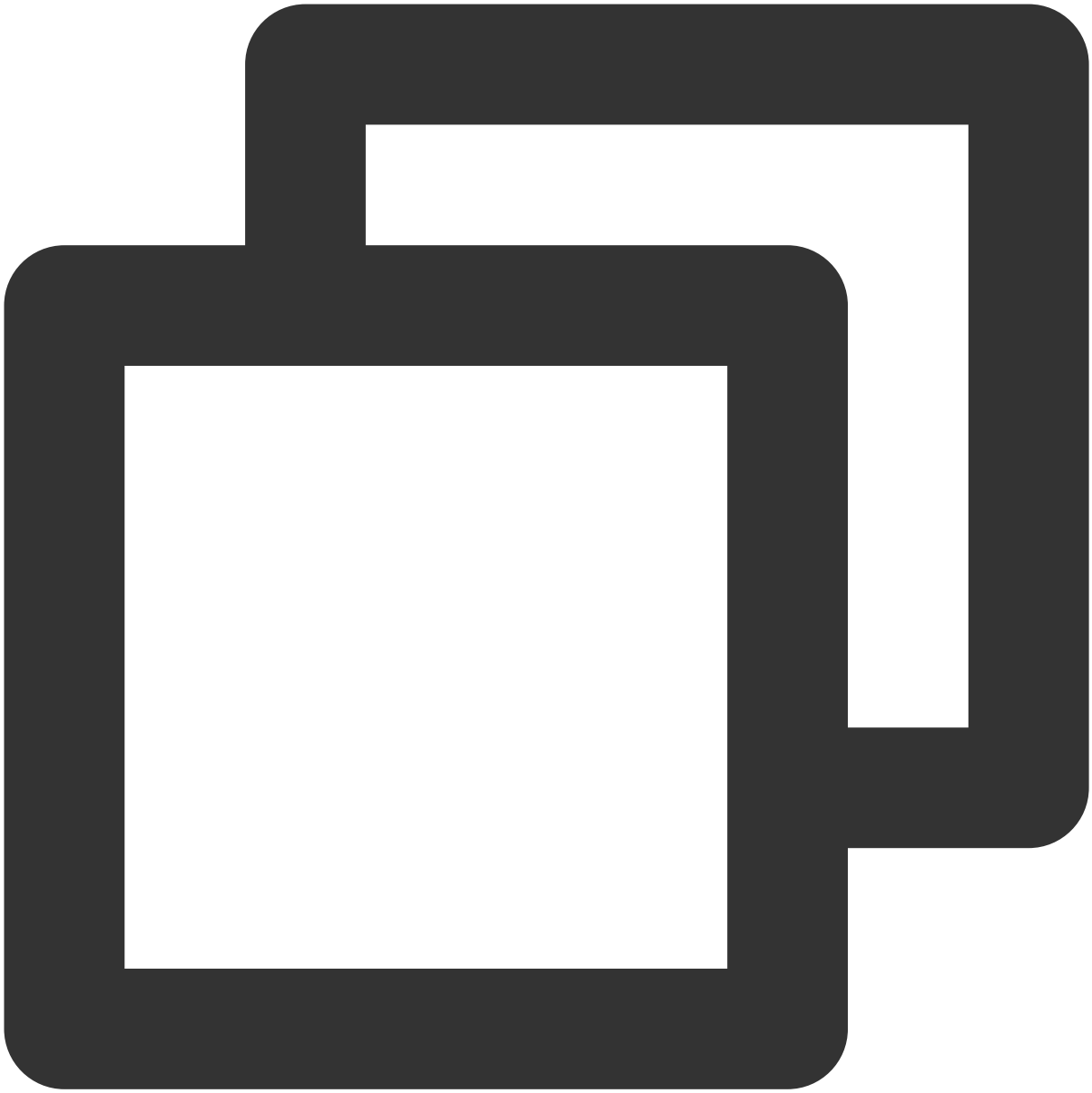
客户端网络正常，但是通过 HTTP 访问 COS 非常慢，或者报错 Connection reset，该如何处理？

部分区域的运营商可能会对 COS 的域名进行劫持，因此建议通过 HTTPS 来访问 COS。

引入 SDK 运行后，出现 java.lang.NoSuchMethodError 的异常，该如何处理？

原因：一般是发生了 JAR 包冲突，例如，用户的工程中的 httpclient 库中 的 JAR 包版本没有 A 方法，但是 SDK 依赖的 JAR 包使用了 A 方法。此时，由于运行时加载顺序的问题，加载了用户工程中的 httpclient 库，运行时便会抛出 NoSuchMethodError 的异常。
解决方法：
方式一：将工程中引起 NoSuchMethodError 包的版本，改成和 SDK 中 pom.xml 里的对应库的版本一致。
方式二：将 cos-java-sdk 换成 cos_api-bundle。此方案会将 cos-java-sdk 的所有依赖都独立安装，所以会占用更多的空间。

版权所有：腾讯云计算（北京）有限责任公司



```
<groupId>com.qcloud</groupId>
     <artifactId>cos_api-bundle</artifactId>
<version>5.6.35</version>
```

Java SDK 的默认超时时间是多少？

Java SDK 默认连接超时时间为30000ms，默认读写超时时间为30000ms，可以使用 SDK 中的 SetConnectionTimeoutMs 方法和 setSocketTimeout 来进行调整。

Java SDK 上传速度慢，日志频繁打印 IOException，该如何处理？

原因与解决办法如下：

1. 确认是否是通过公网访问 COS 或 CI。

目前同地域 CVM 访问 COS 一般通过内网访问（内网域名解析出的 IP 是10、100、169网段，有关 COS 域名请参见地域和访问域名）。如果是通过公网访问，需确认出口带宽是否较小，或者是否有其他程序占用带宽资源。

2. 确保在生产环境中的日志级别不是 DEBUG，推荐使用 INFO 日志。

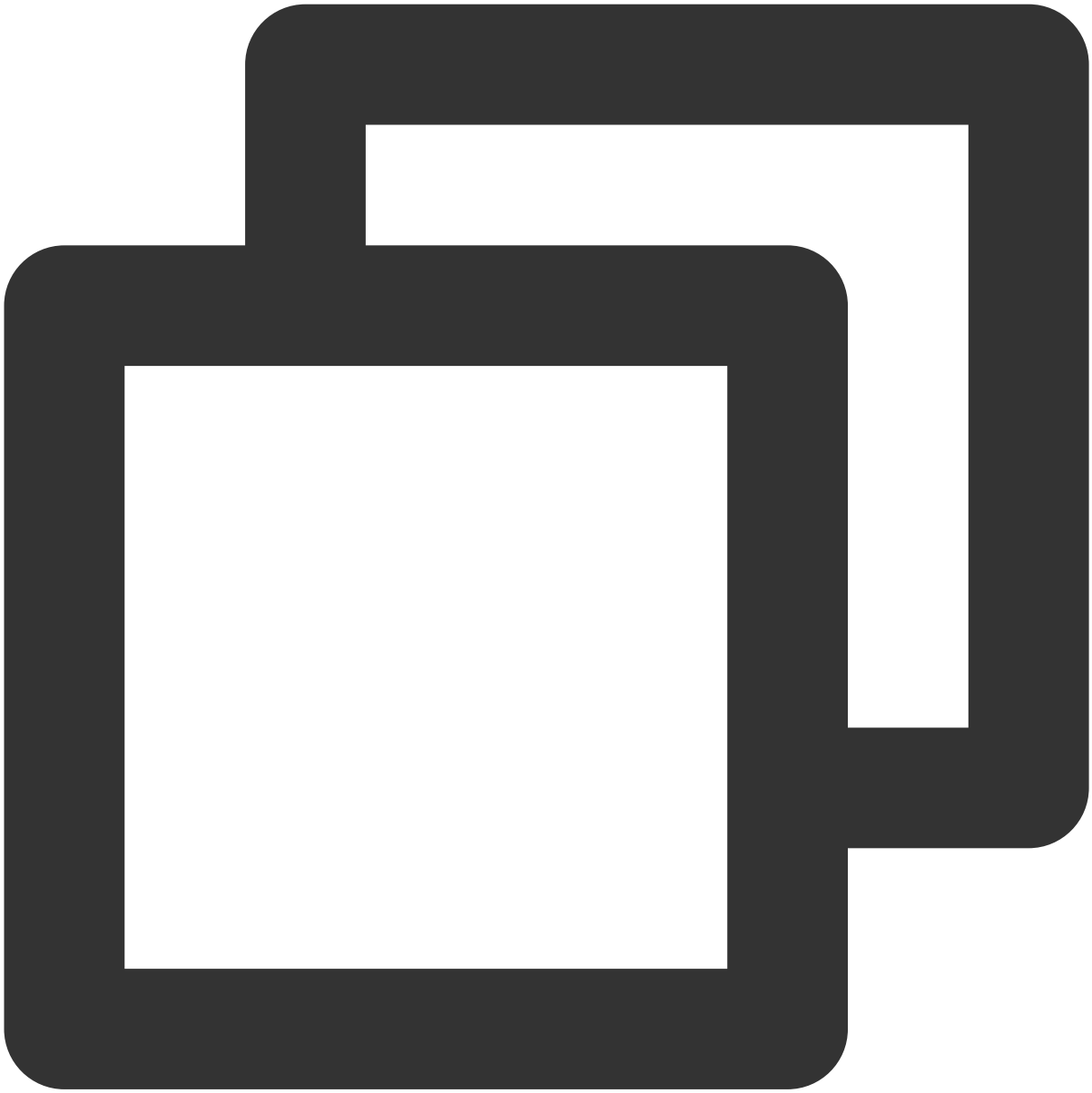
3. 如果 WARN 日志打印 IOException 可以忽略，SDK 会进行重试。IOException 的原因可能是网速过慢，解决办法可参考步骤1和步骤2。

上传文件请求路径中带有“+”，报错 Code: 403 SignatureDoesNotMatch，如何处理？

可能原因：用户 Java 环境的 httpclient 版本引起的 urlencode 编码错误。

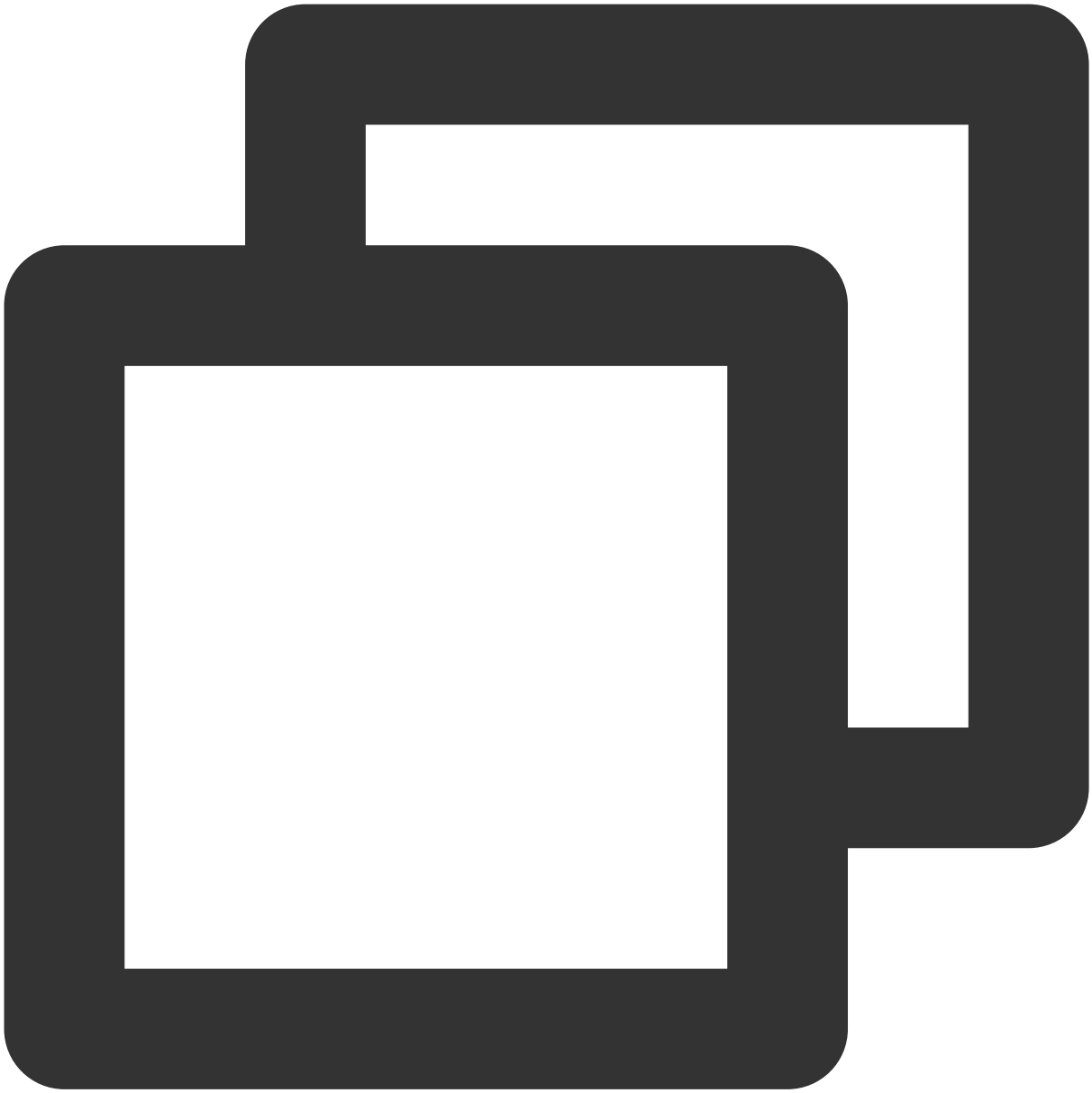
解决方法：

方式一：使用 httpclient 4.5.3 版本



```
<groupId>org.apache.httpcomponents</groupId>
     <artifactId>httpclient</artifactId>
 <version>4.5.3</version>
```

方式二：将 cos-java-sdk 换成 cos_api-bundle。此方案会将 cos-java-sdk 的所有依赖都独立安装，所以会占用更多的空间。



```
<groupId>com.qcloud</groupId>
     <artifactId>cos_api-bundle</artifactId>
<version>5.6.35</version>
```

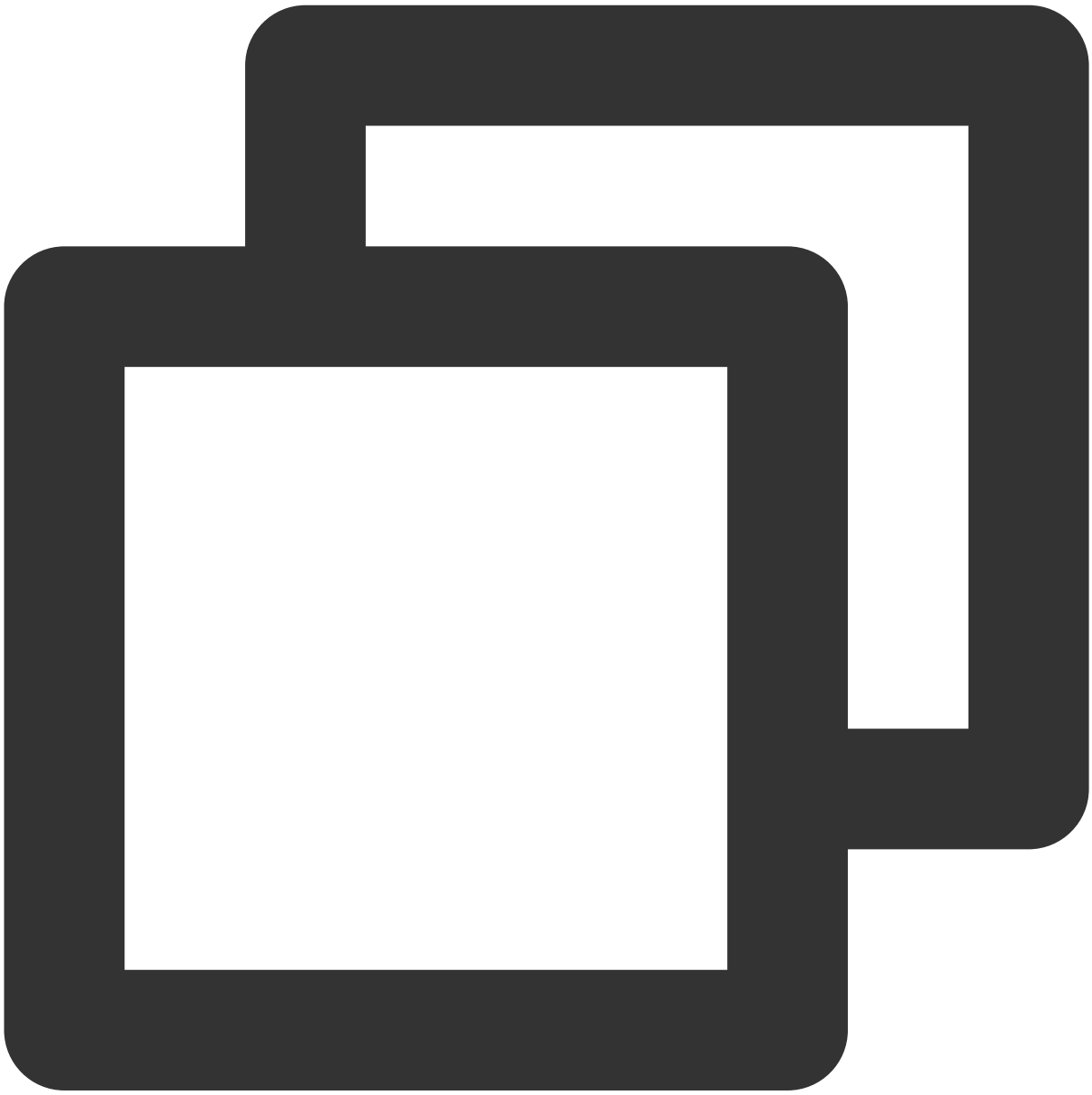
使用 Java SDK 报错提示某个依赖版本太低，如何处理？

原因：用户 Java 环境的依赖版本和 Java SDK 需要的依赖版本有冲突。

解决方法：

方式一：按照提示，升级对应依赖到要求的版本。

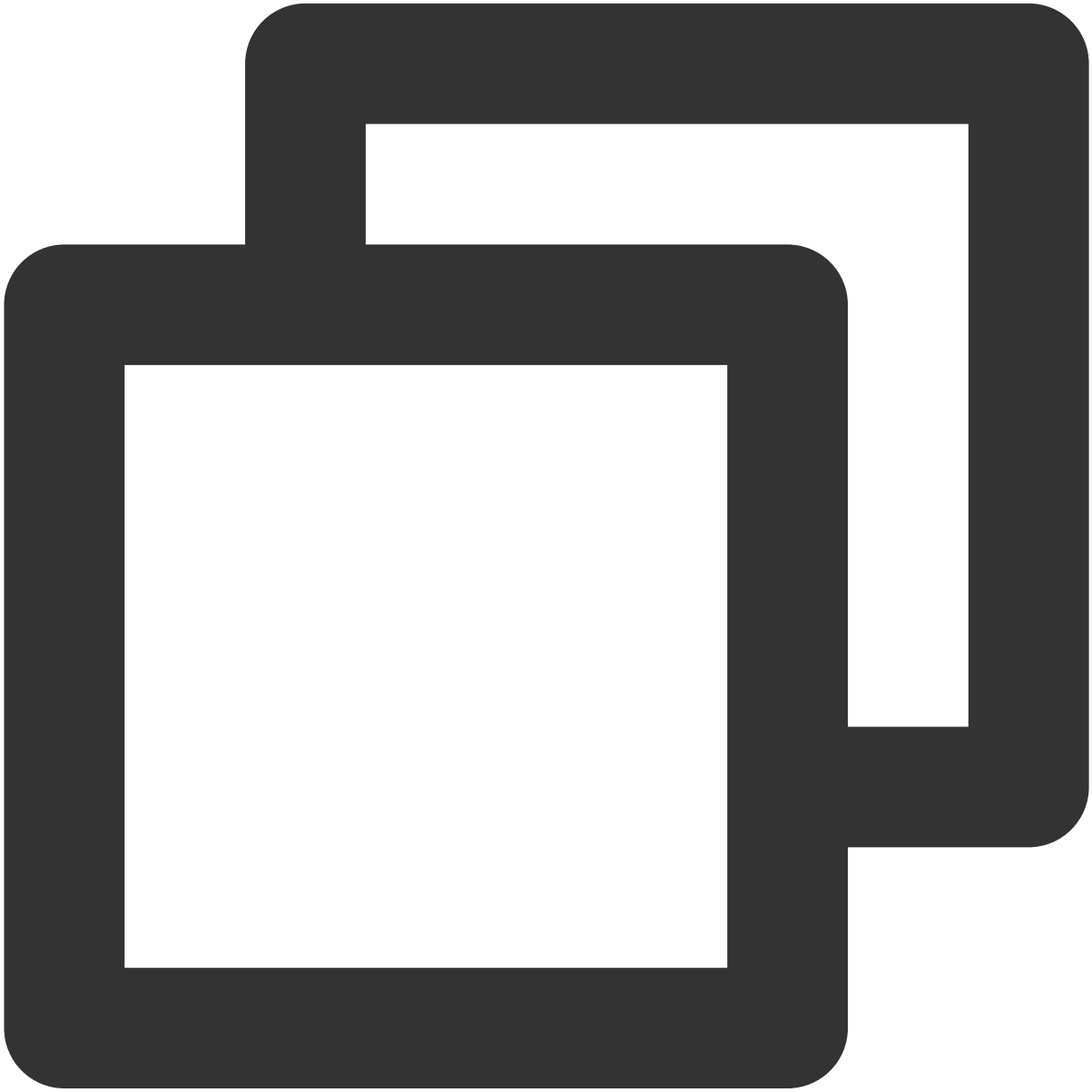
方式二：将 cos-java-sdk 换成 cos_api-bundle。此方案会将 cos-java-sdk 的所有依赖都独立安装，所以会占用更多的空间。



```
<groupId>com.qcloud</groupId>
     <artifactId>cos_api-bundle</artifactId>
<version>5.6.35</version>
```

SDK 如何创建COS的文件目录？

对象存储中文件和目录都是对象，目录只是以 `/` 结尾的对象。创建文件时，不需要创建目录。如创建一个对象键为 `xxx/yyy/zzz.txt` 的文件，只用把 key 设置为 `xxx/yyy/zzz.txt` 即可，不用建立 `xxx/yyy/` 这个对象。在控制台上展示时，也会以 `/` 作为分隔，展示出目录的层级效果。但这些目录对象是不存在的。如果想创建一个目录对象，可使用以下的示例代码：

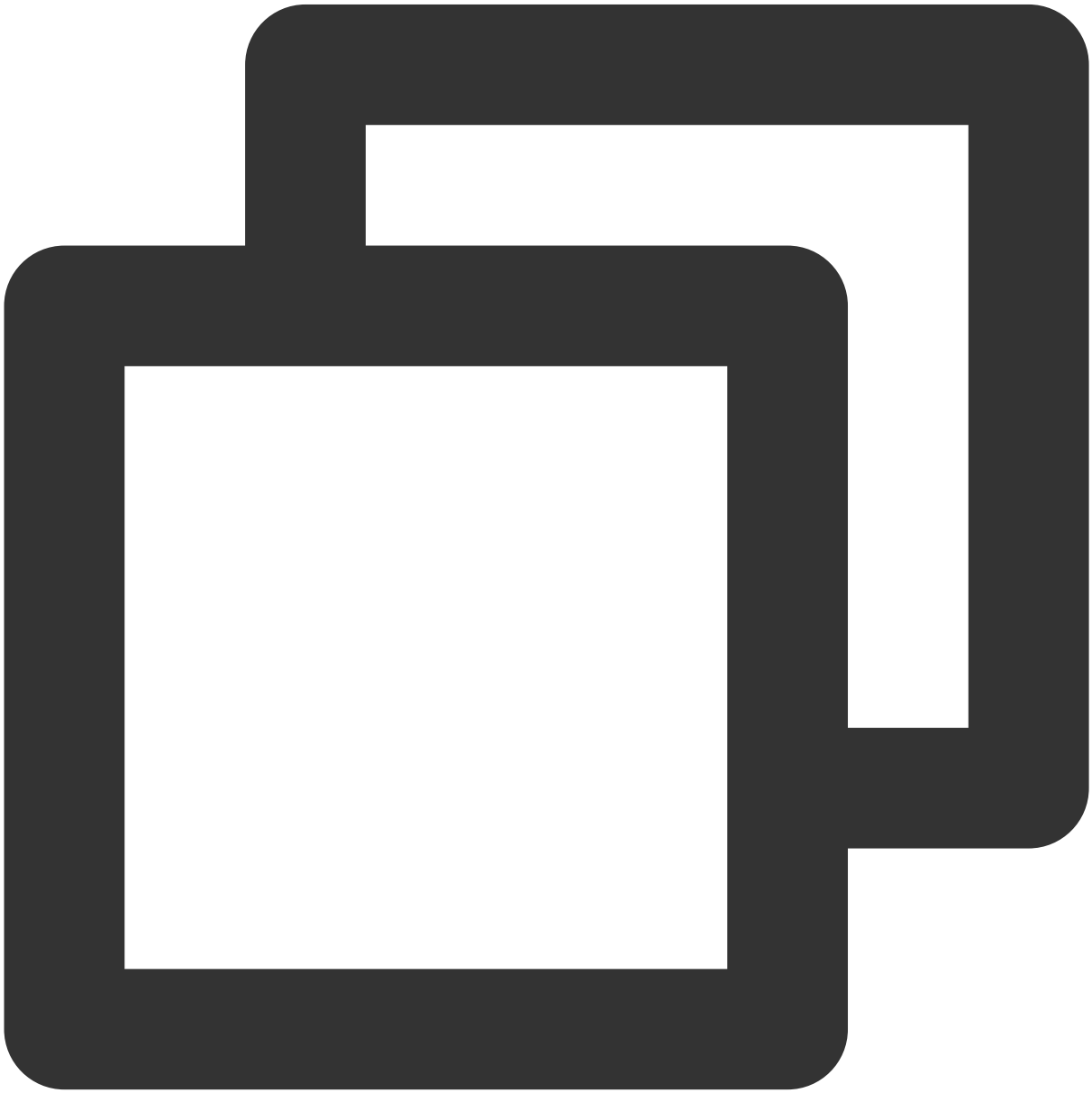


```
String bucketName = "examplebucket-1250000000";
String key = "folder/images/";
// 目录对象即是一个/结尾的空文件，上传一个长度为 0 的 byte 流
InputStream input = new ByteArrayInputStream(new byte[0]);
ObjectMetadata objectMetadata = new ObjectMetadata();
```

```
objectMetadata.setContentLength(0);

PutObjectRequest putObjectRequest =
new PutObjectRequest(bucketName, key, input, objectMetadata);
PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
```

SDK 如何使用 HTTPS?

SDK 中相关的配置都统一放在 ClientConfig 类中，示例代码如下：

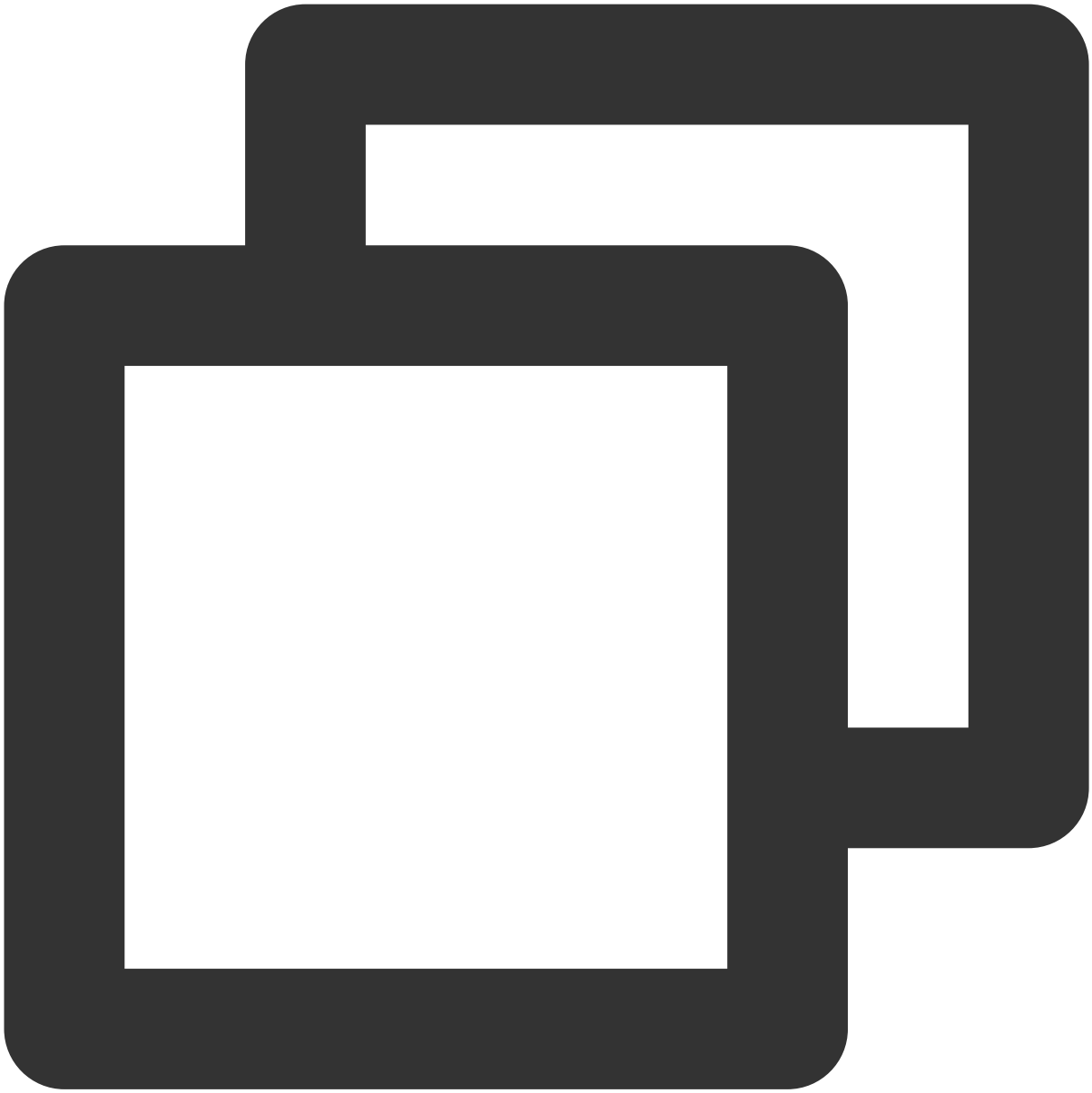


```
// 初始化用户身份信息(secretId, secretKey)
```

```
String secretId = "COS_SECRETID";
String secretKey = "COS_SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 设置 bucket 的区域, COS 地域的简称请参照 https://www.tencentcloud.com/document/produ
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));
// 配置使用 https
clientConfig.setHttpProtocol(HttpProtocol.https);
// 生成 cos 客户端
COSClient cosClient = new COSClient(cred, clientConfig);
```

SDK 如何使用代理？

对于需要使用代理访问 COS 或 CI 的用户，可在 ClientConfig 类中，配置使用代理 IP （或域名）以及端口，示例代码如下：

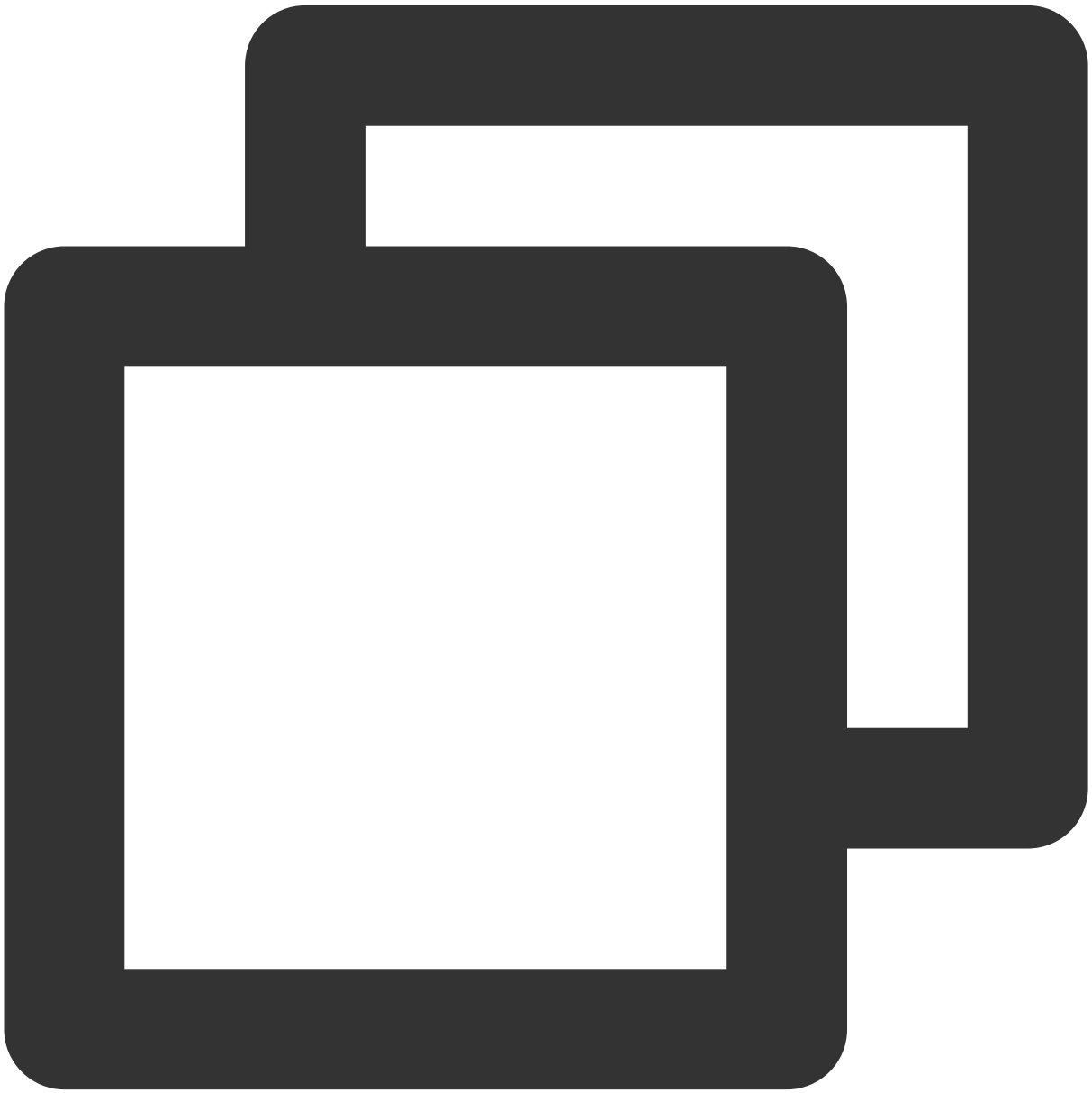


```
// 初始化用户身份信息(secretId, secretKey)
String secretId = "COS_SECRETID";
String secretKey = "COS_SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 设置 bucket 的区域, COS 地域的简称请参照 https://www.tencentcloud.com/document/produ
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));
// 配置使用代理(IP 和端口需要同时设置)
// 设置代理 IP (也可传入域名)
clientConfig.setHttpProxyIp("192.168.2.3");
// 设置代理端口
clientConfig.setHttpProxyPort(8080);
```

```
// 生成 cos 客户端
COSClient cosClient = new COSClient(cred, clientConfig);
```

如何设置自定义 EndpointBuilder？

您的场景也许需要指定 API 请求的 Endpoint。此时，您需要实现 EndpointBuilder 接口中的 buildGeneralApiEndpoint 和 buildGetServiceApiEndpoint 中的两个函数，分别为普通 API 请求和 GETService 请求指定远端的 Endpoint。使用示例如下：



```
// 步骤1：实现 EndpointBuilder 接口中的两个函数
class SelfDefinedEndpointBuilder implements EndpointBuilder {
```

```
    @Override
    public String buildGeneralApiEndpoint(String bucketName) {
        return String.format("%s.%s", bucketName, "mytest.com");
    }

    @Override
    public String buildGetServiceApiEndpoint() {
        return "service.mytest.com";
    }
}

// 步骤2：初始化客户端
String secretId = "COS_SECRETID";
String secretKey = "COS_SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
SelfDefinedEndpointBuilder selfDefinedEndpointBuilder = new SelfDefinedEndpointBuil
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing"));
clientConfig .setEndpointBuilder(selfDefinedEndpointBuilder);
COSClient cosClient = new COSClient(cred, clientConfig);
```

SDK 的上传、下载、批量删除等操作中，使用的 key 值是否需要添加 / 前缀？

对象存储的 key 值无需携带 / 前缀。例如，您将对象 key 值设置为 exampleobject 上传的对象，可以通过 URL：`http://cos.ap-guangzhou.myqcloud.com/exampleobject` 进行访问。

注意：

在批删请求中，请勿传入 / 前缀的 key，这将导致对象删除失败。

使用 Java SDK 上传报错 please make sure bucket name must contain legal appid when appid is missing. example: bucektname-1250000000，该如何处理？

COS Java SDK 旧版本仅支持例如125开头的 APPID，如果您使用的是130开头的 APPID，建议您将 SDK 更新至最新版本。Java SDK 下载安装请参见 Java SDK 文档。

使用 Java SDK 上传大文件报错，该如何处理？

使用 Java SDK 上传大于5G的文件时，建议使用 高级上传接口。高级上传接口根据用户文件的长度和数据类型，自动选择以简单上传或分块上传方式。

Java SDK 如何获取文件上传进度？

需获取上传进度，建议使用 高级上传 接口，通过调用 Upload 的方法 getProgress() 获取上传的进度 TransferProgress 类。

Java SDK 能否使用长连接?

Java SDK 默认使用长连接。

如何使用 Java SDK 获取指定目录所有路径的同时，判断路径是文件还是文件夹？

可以使用 Java SDK 中的 查询对象列表 功能来列出存储桶内的对象，**使用 prefix 参数可以指定目录前缀**。

Java SDK 报错连接池已关闭，抛出异常： java.lang.IllegalStateException: Connection pool shut down，该如何处理？

1. COSClient 是线程安全的类，允许多线程访问同一实例。
因为实例内部维持了一个连接池，创建多个实例可能导致程序资源耗尽。请确保程序生命周期内实例只有一个，且在不再需要使用时，调用 COSClient.shutdown() 方法将其关闭。
如果需要新建实例，请先将之前的实例关闭。**推荐一个进程里只使用一个 COSClient，在程序全部结束退出时才调用 COSClient.shutdown()**。
2. 高级接口封装类的 TransferManager 中也使用了 COSClient，建议在程序全部结束退出时再调用 TransferManager.shutdownNow(false)。
此外，如果在创建 TransferManager 实例时复用了其他 COSClient，请在关闭时加入参数 false（即 TransferManager.shutdownNow(false)），避免关闭复用的 COSClient，导致其他使用 COSClient 的地方报错。

图片处理
基础图片处理

最近更新时间：2024-03-13 10:07:10

简介

腾讯云对象存储（Cloud Object Storage，COS）集成了 数据万象（Cloud Infinite，CI）专业的一体化多媒体解决方案，涵盖以下图片处理功能，详情可见 图片处理概述。

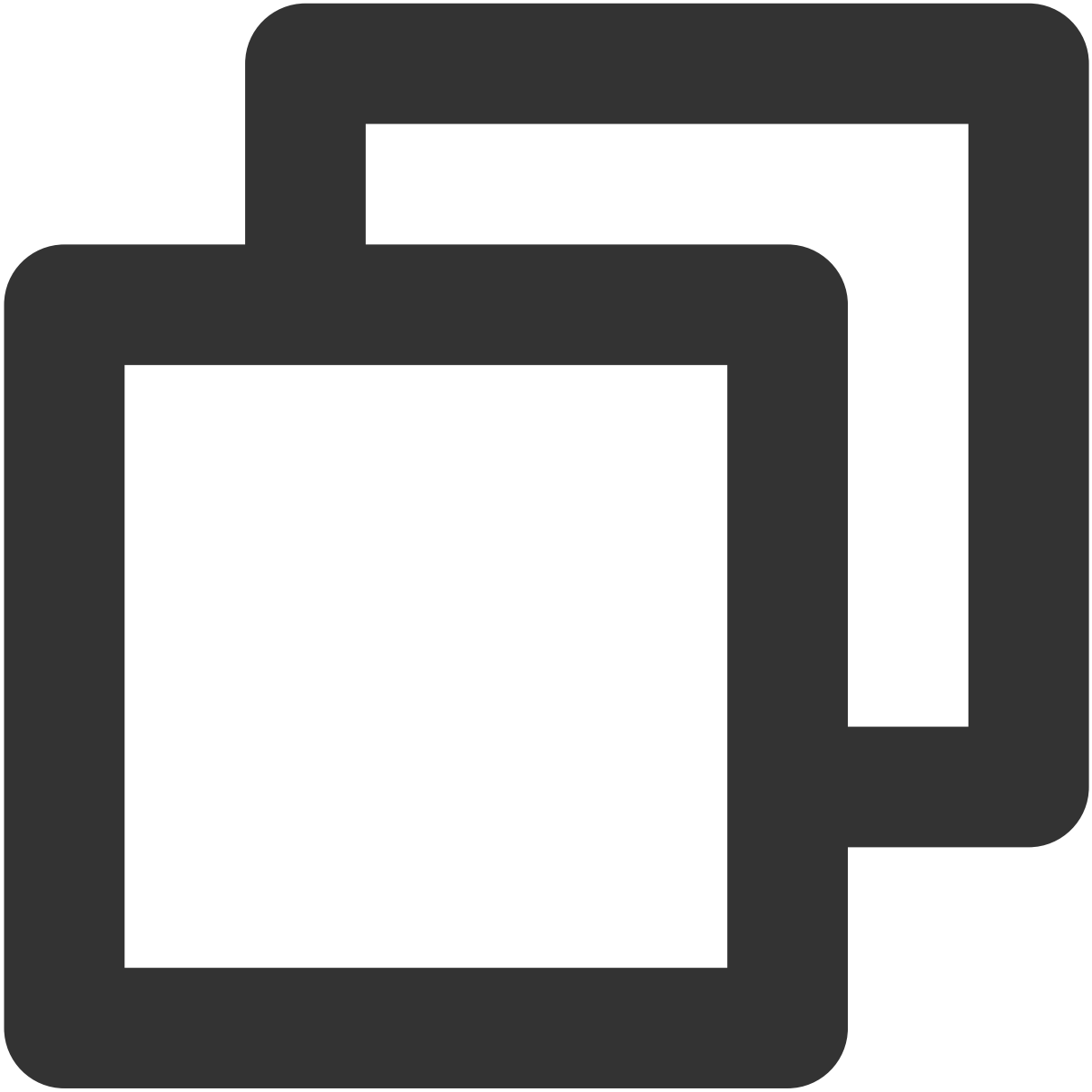
服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

基础图片处理

进行基础图片处理时，需要给定图片处理规则 rule，详情可见 图片处理概述。
所有示例代码请参见 SDK 基础图片处理示例。
部分示例如下：

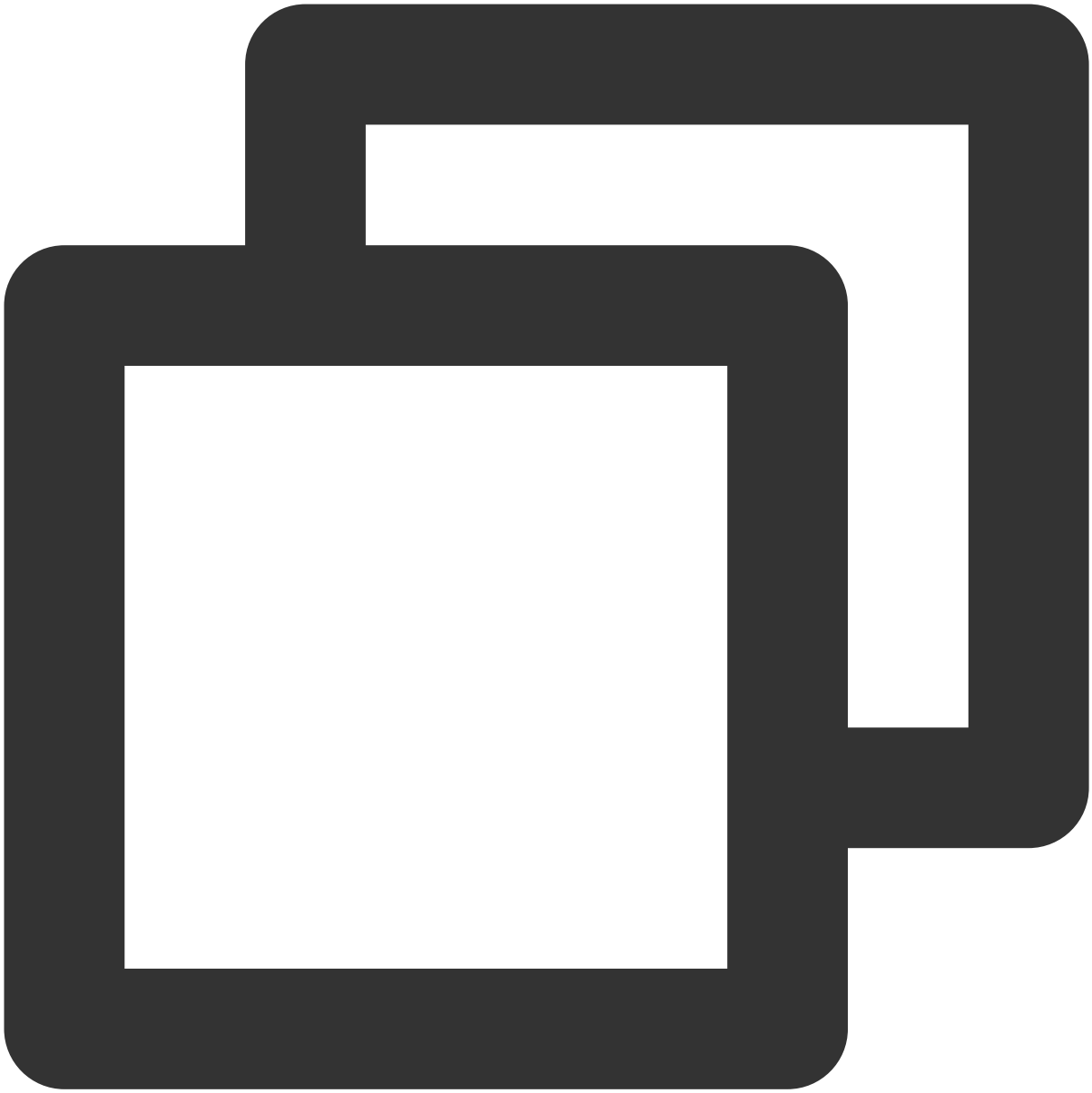
版权所有：腾讯云计算（北京）有限责任公司

缩放



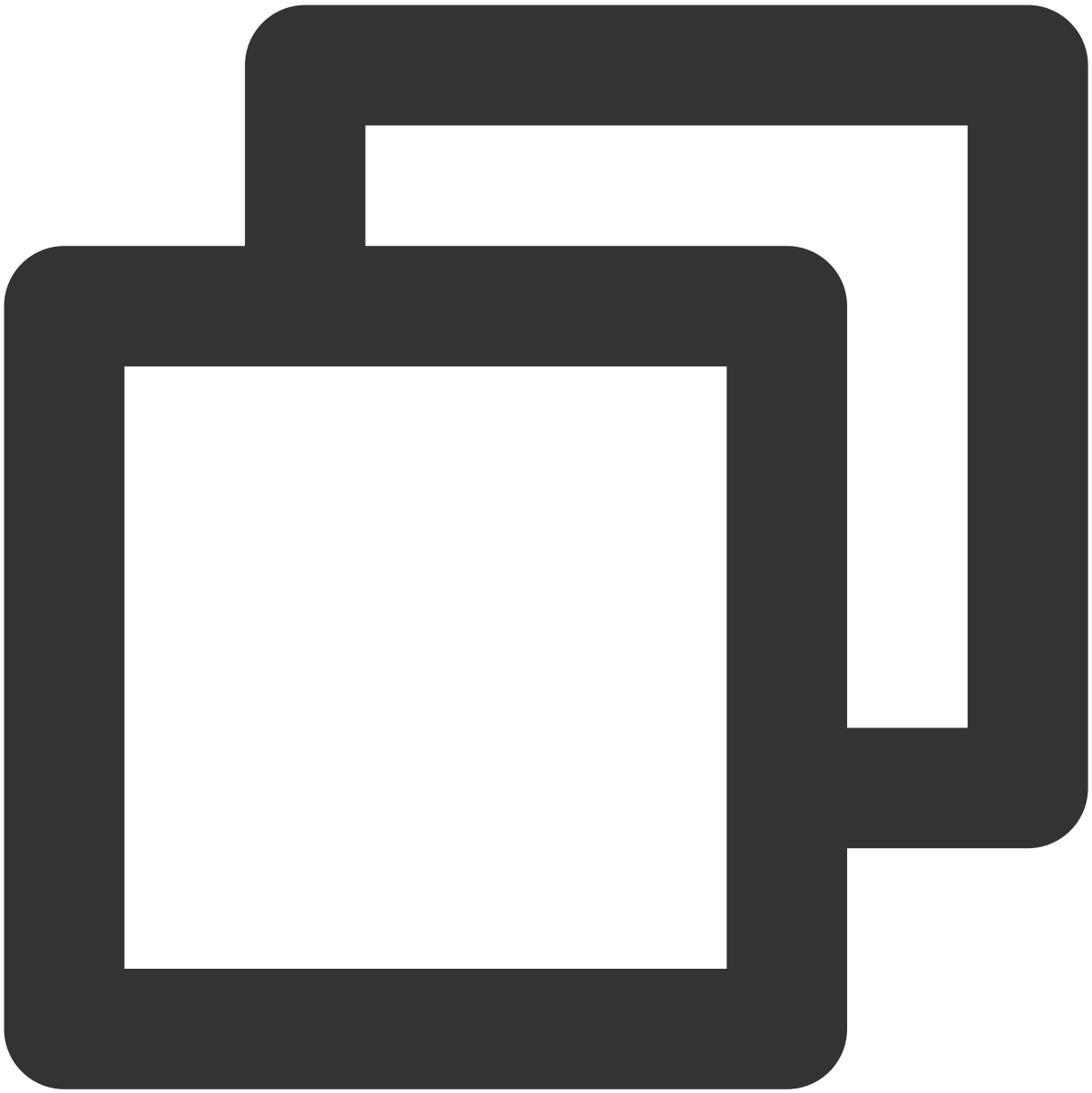
```
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 宽高缩放为原图的50%
String rule = "imageMogr2/thumbnail/!50p";
getObj.putCustomQueryParameter(rule, null);
cosClient.getObject(getObj, new File("qrcode-50p.png"));
```

裁剪



```
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 裁剪内切圆的半径，取值范围为大于0且小于原图最小边一半的整数。内切圆的圆心为图片的中心
String rule = "imageMogr2/iradius/150";
getObj.putCustomQueryParameter(rule, null);
cosClient.getObject(getObj, new File("qrcode-cropping.png"));
```

旋转



```
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 顺时针旋转90度
String rule = "imageMogr2/rotate/90";
getObj.putCustomQueryParameter(rule, null);
cosClient.getObject(getObj, new File("qrcode-rotate.png"));
```

图片持久化处理

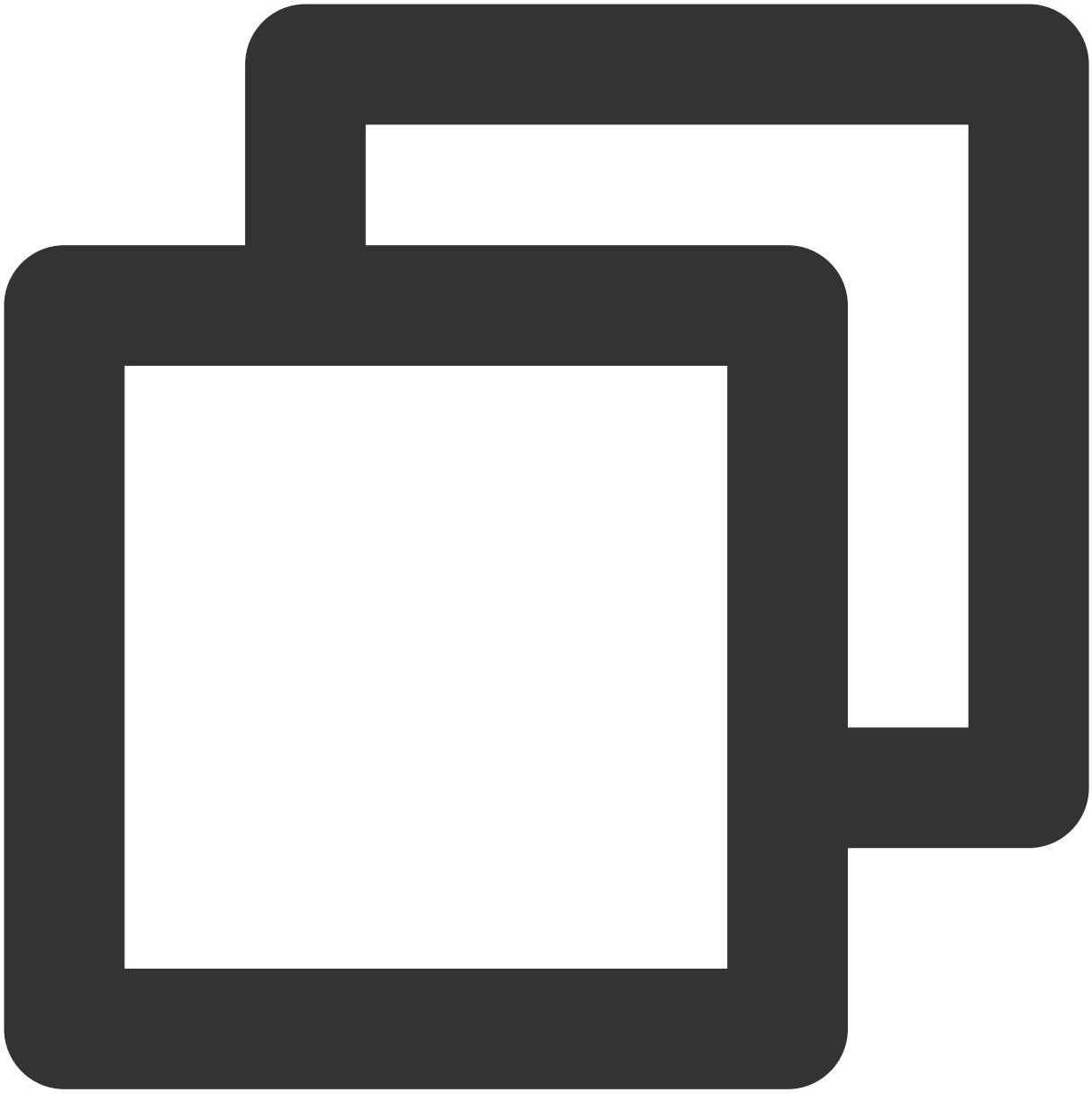
最近更新时间：2024-03-13 10:07:10

简介

本文示例展示了如何在上传图片时自动实现图片处理。
图片上传完成后，对象存储（Cloud Object Storage，COS）会存储原始图片和已处理过的图片。后续用户可以通过普通的下载请求获取处理结果。

上传时图片持久化处理

示例代码



```
  // bucket名需包含appid
// api 请参考 https://cloud.tencent.com/document/product/436/54050
String bucketName = "examplebucket-1250000000";

String key = "test.jpg";
File localFile = new File("E://test.jpg");
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile
PicOperations picOperations = new PicOperations();
picOperations.setIsPicInfo(1);
// 添加图片处理规则
List<PicOperations.Rule> ruleList = new LinkedList<>();
```

```
PicOperations.Rule rule1 = new PicOperations.Rule();
rule1.setBucket(bucketName);
rule1.setFileId("test-1.jpg");
rule1.setRule("imageMogr2/rotate/90");
ruleList.add(rule1);
PicOperations.Rule rule2 = new PicOperations.Rule();
rule2.setBucket(bucketName);
rule2.setFileId("test-2.jpg");
rule2.setRule("imageMogr2/rotate/180");
ruleList.add(rule2);
picOperations.setRules(ruleList);
putObjectRequest.setPicOperations(picOperations);
try {
    PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
    CIUploadResult ciUploadResult = putObjectResult.getCiUploadResult();
    System.out.println(putObjectResult.getRequestId());
    System.out.println(ciUploadResult.getOriginalInfo().getEtag());
    for(CIObject ciObject:ciUploadResult.getProcessResults().getObjectList()) {
        System.out.println(ciObject.getLocation());
        System.out.println(ciObject.getEtag());
    }
} catch (CosServiceException e) {
    e.printStackTrace();
} catch (CosClientException e) {
    e.printStackTrace();
}
```

参数说明

PicOperations 类用于记录图像操作，其主要成员说明如下：

成员名称	描述	类型
isPicInfo	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	int
rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	List

返回参数说明

CIUploadResult 类用于返回图片处理结果信息，其主要成员说明如下：

成员名称	描述	类型
originalInfo	原图信息	OriginalInfo
processResults	图片处理结果	ProcessResults

OriginalInfo 类用于记录原图信息，其主要成员说明如下：

成员名称	描述	类型
key	原图文件名	String
location	图片路径	String
imageInfo	原图图片信息	ImageInfo
etag	原图 ETag 信息（若处理结果图覆盖原图则为结果图 ETag 信息）	String

ImageInfo 类用于记录原图图片信息，其主要成员说明如下：

成员名称	描述	类型
format	格式	String
width	图片宽度	Integer
height	图片高度	Integer
quality	图片质量	Integer
ave	图片主色调	String
orientation	图片旋转角度	Integer

ProcessResults 类用于记录图片处理结果，其主要成员说明如下：

成员名称	描述	类型
objectList	每一个图片处理结果	List

CIObject 类用于记录一个图片处理结果，其主要成员说明如下：

成员名称	描述	类型
key	文件名	String
location	图片路径	String
format	图片格式	String
width	图片宽度	Integer
height	图片高度	Integer

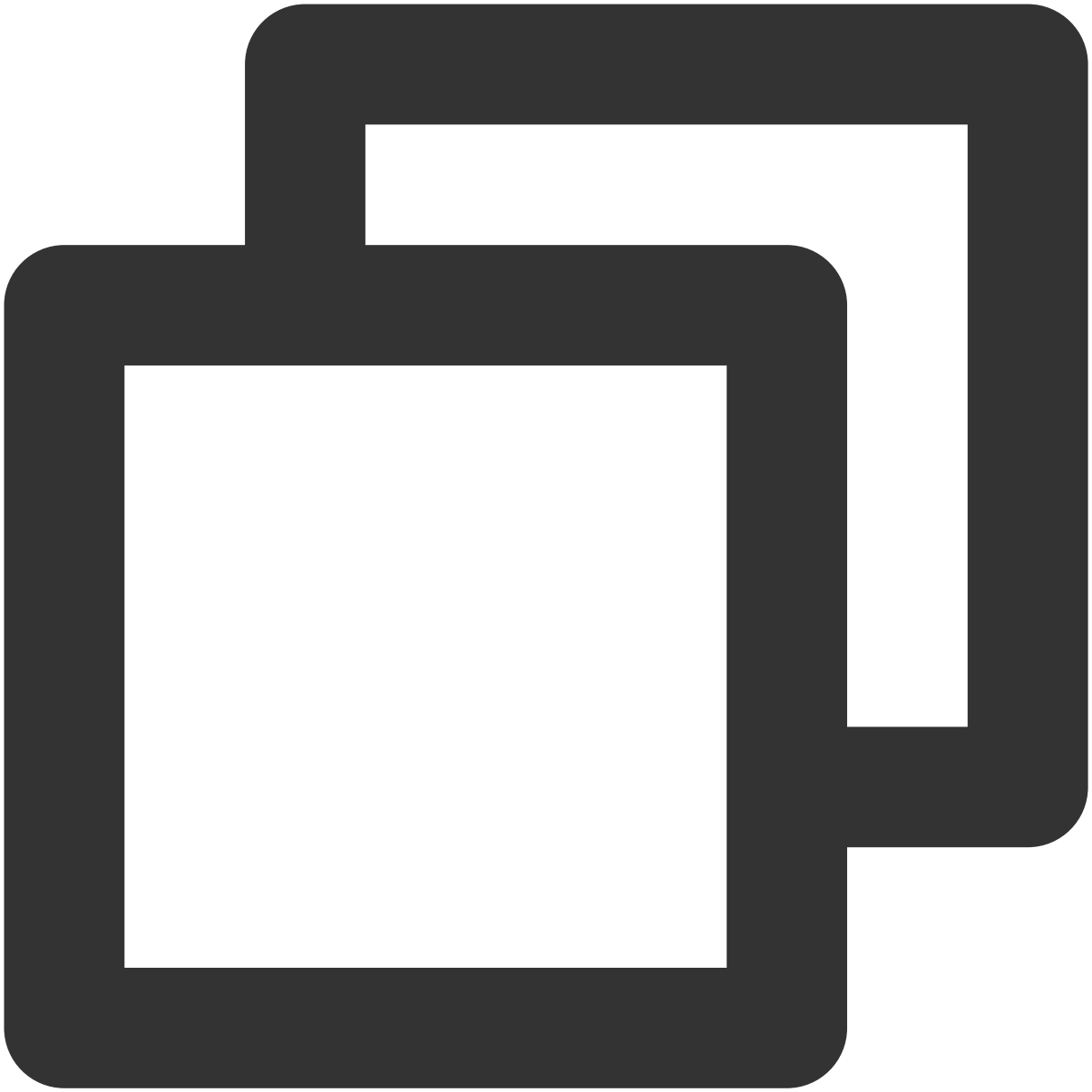
版权所有：腾讯云计算（北京）有限责任公司

size	图片大小	Integer
quality	图片质量	Integer
etag	处理结果图 ETag 信息	String

对云上数据进行图片处理

下面示例展示了如何对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

示例代码



```
String bucketName = "examplebucket-1250000000";
String key = "test.jpg";
ImageProcessRequest imageReq = new ImageProcessRequest(bucketName, key);

PicOperations picOperations = new PicOperations();
picOperations.setIsPicInfo(1);
List<PicOperations.Rule> ruleList = new LinkedList<>();
PicOperations.Rule rule1 = new PicOperations.Rule();
rule1.setBucket(bucketName);
rule1.setFileId("test-1.jpg");
rule1.setRule("imageMogr2/rotate/90");
```

```
ruleList.add(rule1);
PicOperations.Rule rule2 = new PicOperations.Rule();
rule2.setBucket(bucketName);
rule2.setFileId("test-2.jpg");
rule2.setRule("imageMogr2/rotate/180");
ruleList.add(rule2);
picOperations.setRules(ruleList);

imageReq.setPicOperations(picOperations);

CIUploadResult result = cosClient.processImage(imageReq);
result.getProcessResults();
result.getOriginalInfo();

try {
    CIUploadResult ciUploadResult = cosClient.processImage(imageReq);
    System.out.println(ciUploadResult.getOriginalInfo().getEtag());
    for(CIObject ciObject:ciUploadResult.getProcessResults().getObjectList()) {
        System.out.println(ciObject.getLocation());
        System.out.println(ciObject.getEtag());
    }
} catch (CosServiceException e) {
    e.printStackTrace();
} catch (CosClientException e) {
    e.printStackTrace();
}
```

图片高级压缩

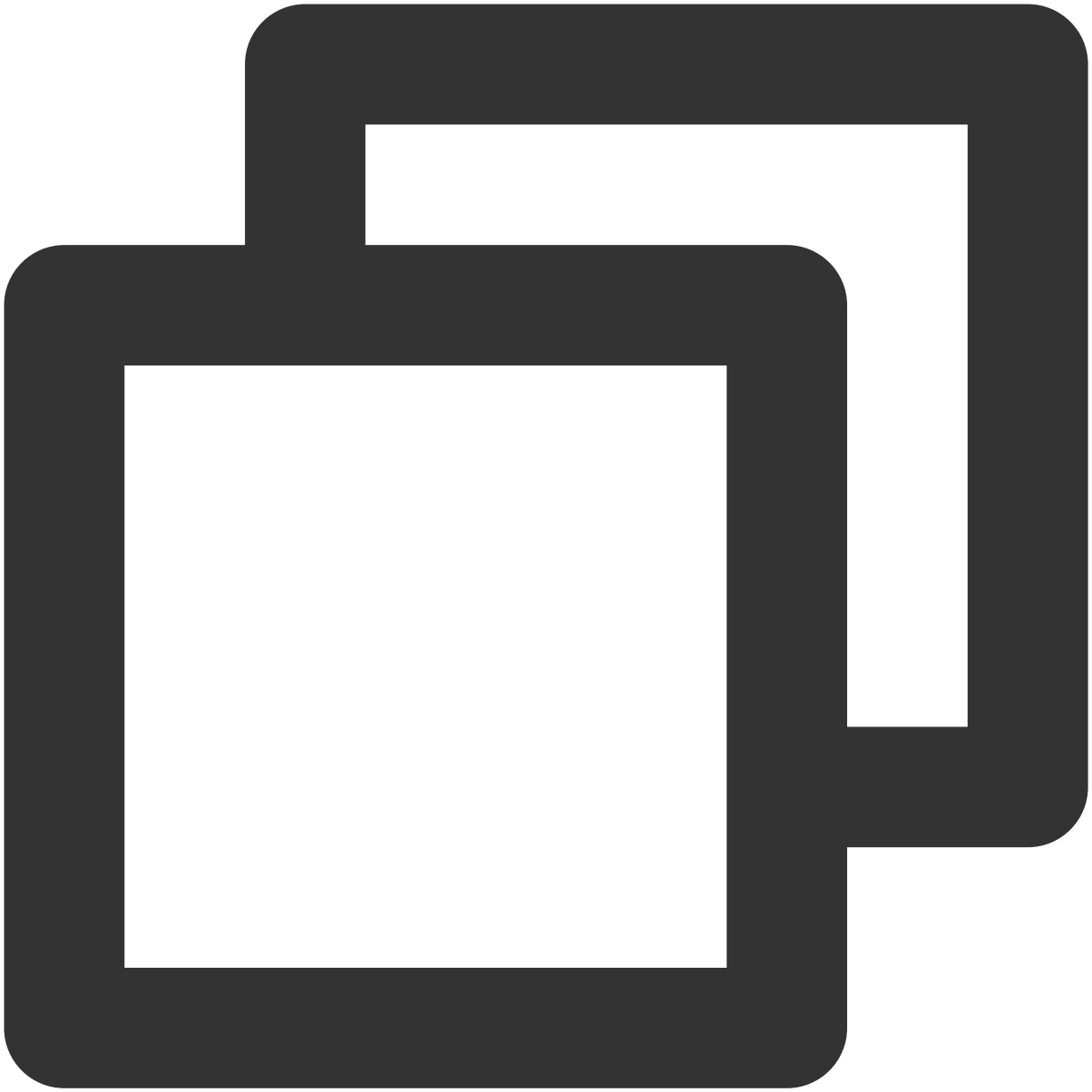
最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于图片高级压缩的 API 概览以及 SDK 示例代码。

API	操作描述
图片高级压缩	图片高级压缩可以更加高效地将图片转换为 TPG 或 HEIF 等高压缩比格式，有效降低图片传输链路及加载耗时，降低带宽及流量成本。

示例代码



```
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 这里是图片压缩参数，具体请参考数据万象API，这里仅是示例
String compress = "imageMogr2/format/tpg";
getObj.putCustomQueryParameter(compress, null);
```

盲水印

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于盲水印的 API 概览以及 SDK 示例代码。

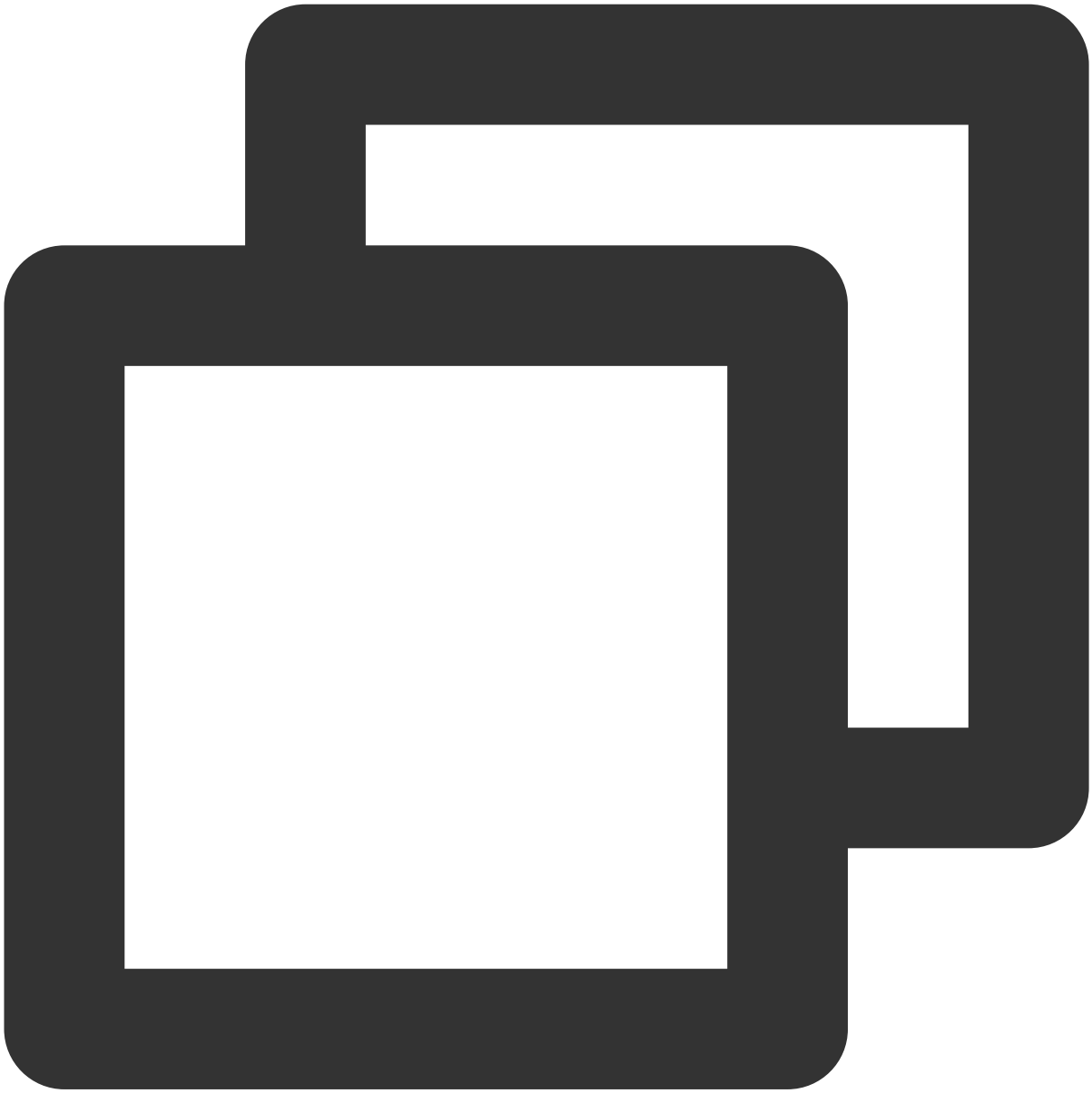
API	操作描述
盲水印	对本地图片添加或提取盲水印并上传至存储桶

添加盲水印

功能说明

盲水印支持在上传时添加以及下载时添加。

示例代码：上传时添加盲水印



```
// bucket名需包含appid
// api 请参考：https://www.tencentcloud.com/document/product/436/49729
String bucketName = "examplebucket-1250000000";

String key = "test.png";
File localFile = new File("E://test.png");
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile
PicOperations picOperations = new PicOperations();
picOperations.setIsPicInfo(1);
List<PicOperations.Rule> ruleList = new LinkedList<>();
PicOperations.Rule rule = new PicOperations.Rule();
```

```
rule.setBucket(bucketName);
rule.setFileId("test-watermark.png");
rule.setRule("watermark/3/type/3/text/dGVuY2VudCBjbG91ZA==");
ruleList.add(rule);
picOperations.setRules(ruleList);
putObjectRequest.setPicOperations(picOperations);
try {
    PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
    CIUploadResult ciUploadResult = putObjectResult.getCiUploadResult();
    System.out.println(putObjectResult.getRequestId());
    System.out.println(ciUploadResult.getOriginalInfo().getEtag());
    for(CIObject ciObject:ciUploadResult.getProcessResults().getObjectList()) {
        System.out.println(ciObject.getLocation());
    }
} catch (CosServiceException e) {
    e.printStackTrace();
} catch (CosClientException e) {
    e.printStackTrace();
}
```

参数说明

PicOperations 类用于记录图像操作，其主要成员说明如下：

成员名称	描述	类型
isPicInfo	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	int
rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	List

返回参数说明

CIUploadResult 类用于返回图片处理结果信息，其主要成员说明如下：

成员名称	描述	类型
originalInfo	原图信息	OriginalInfo
processResults	图片处理结果	ProcessResults

OriginalInfo 类用于记录原图信息，其主要成员说明如下：

成员名称	描述	类型
key	原图文件名	String
location	图片路径	String

imageInfo	原图图片信息	ImageInfo

ImageInfo 类用于记录原图图片信息，其主要成员说明如下：

成员名称	描述	类型
format	格式	String
width	图片宽度	Integer
height	图片高度	Integer
quality	图片质量	Integer
ave	图片主色调	String
orientation	图片旋转角度	Integer

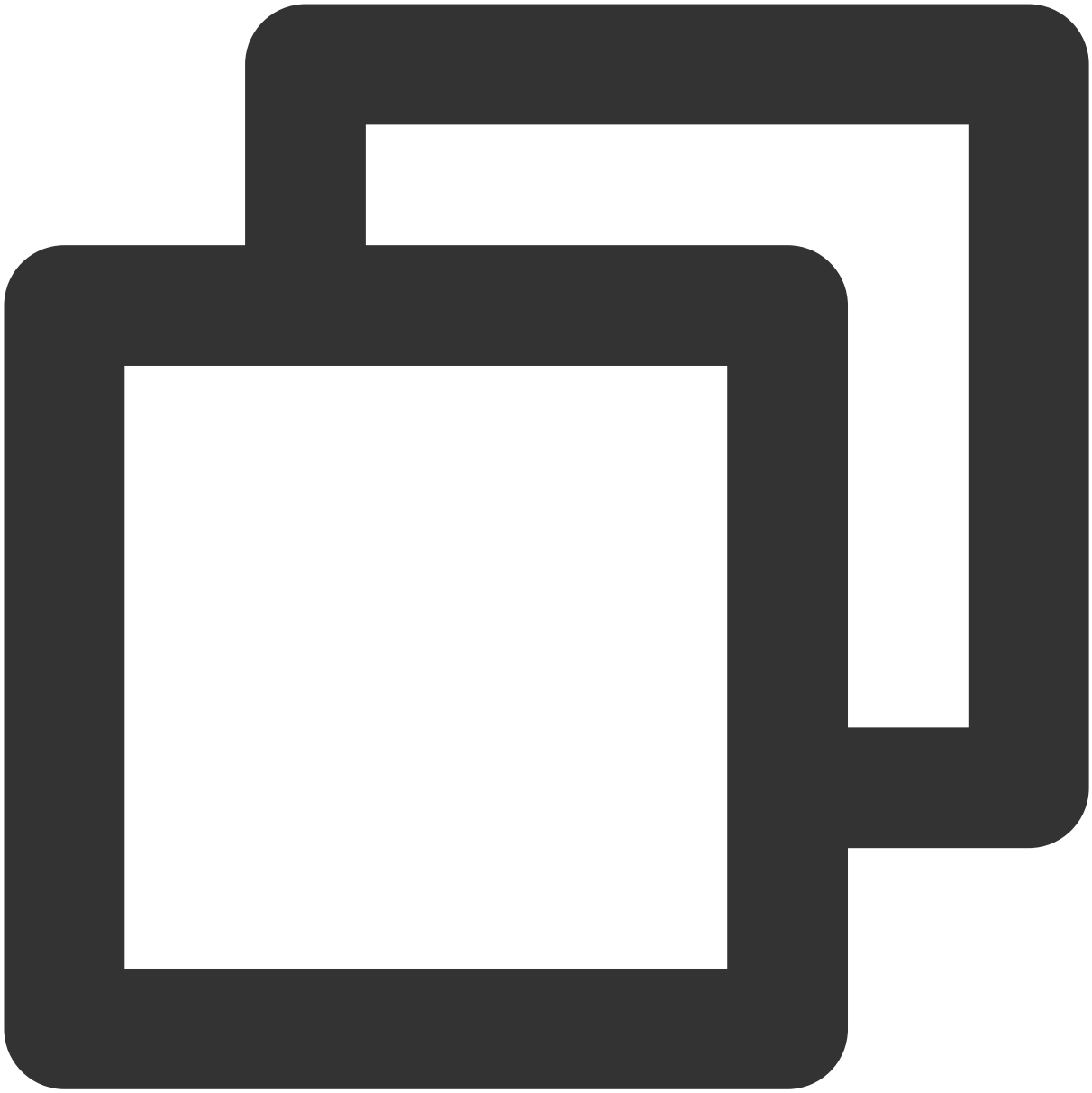
ProcessResults 类用于记录图片处理结果，其主要成员说明如下：

成员名称	描述	类型
objectList	每一个图片处理结果	List

CIObject 类用于记录一个图片处理结果，其主要成员说明如下：

成员名称	描述	类型
key	文件名	String
location	图片路径	String
format	图片格式	String
width	图片宽度	Integer
height	图片高度	Integer
size	图片大小	Integer
quality	图片质量	Integer

示例代码：下载时添加盲水印

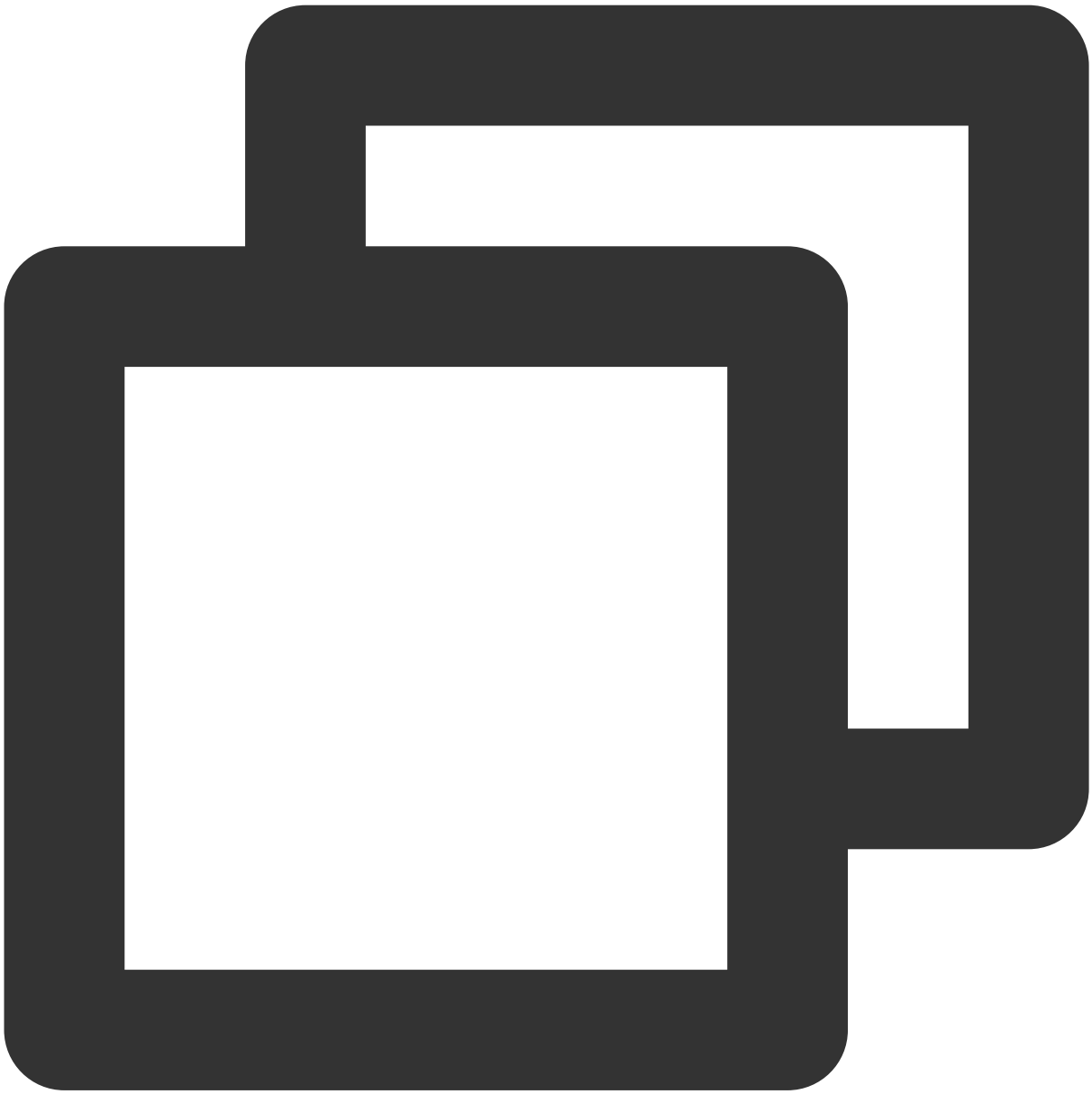


```
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 这里是盲水印规则，具体请参考数据万象API，这里仅是示例
String rule = "watermark/3/type/3/text/dGVuY2VudCBjbG91ZA==";
getObj.putCustomQueryParameter(rule, null);

cosClient.getObject(getObj);
```

提取盲水印

下面示例展示了如何对已添加盲水印的图片提取盲水印。

示例代码



```
String bucketName = "examplebucket-1250000000";
String key = "qrcode-watermark.png";
File localFile = new File("E://qrcode-watermark.png");
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile
PicOperations picOperations = new PicOperations();
picOperations.setIsPicInfo(1);
List<PicOperations.Rule> ruleList = new LinkedList<>();
```

```
PicOperations.Rule rule = new PicOperations.Rule();
rule.setBucket(bucketName);
rule.setFileId("qrcode-watermark-extract.png");
rule.setRule("watermark/4/type/2/image/aHR0cDovL2V4YW1wbGVidWNrZXQtMTI1MDAwMDAwMC5j
ruleList.add(rule);
picOperations.setRules(ruleList);
putObjectRequest.setPicOperations(picOperations);
try {
    PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
    CIUploadResult ciUploadResult = putObjectResult.getCiUploadResult();
    System.out.println(putObjectResult.getRequestId());
    System.out.println(ciUploadResult.getOriginalInfo().getEtag());
    for(CIObject ciObject:ciUploadResult.getProcessResults().getObjectList()) {
        System.out.println(ciObject.getLocation());
        System.out.println(ciObject.getWatermarkStatus());
    }
} catch (CosServiceException e) {
    e.printStackTrace();
} catch (CosClientException e) {
    e.printStackTrace();
}
```

参数说明

PicOperations 类用于记录图像操作，其主要成员说明如下：

成员名称	描述	类型
isPicInfo	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	int
rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	List

返回参数说明

CIUploadResult 类用于返回图片处理结果信息，其主要成员说明如下：

成员名称	描述	类型
originalInfo	原图信息	OriginalInfo
processResults	图片处理结果	ProcessResults

OriginalInfo 类用于记录原图信息，其主要成员说明如下：

成员名称	描述	类型
key	原图文件名	String

版权所有：腾讯云计算（北京）有限责任公司

location	图片路径	String
imageInfo	原图图片信息	ImageInfo

ImageInfo 类用于记录原图图片信息，其主要成员说明如下：

成员名称	描述	类型
format	格式	String
width	图片宽度	Integer
height	图片高度	Integer
quality	图片质量	Integer
ave	图片主色调	String
orientation	图片旋转角度	Integer

ProcessResults 类用于记录图片处理结果，其主要成员说明如下：

成员名称	描述	类型
objectList	每一个图片处理结果	List

CIObject 类用于记录一个图片处理结果，其主要成员说明如下：

成员名称	描述	类型
key	文件名	String
location	图片路径	String
format	图片格式	String
width	图片宽度	Integer
height	图片高度	Integer
size	图片大小	Integer
quality	图片质量	Integer
watermarkStatus	当 type 为2时返回该字段，表示提取到全盲水印的可信度。具体为0 - 100的数字，75分以上表示确定有盲水印，60 - 75表示疑似有盲水印，60以下可认为未提取到盲水印	Integer

版权所有：腾讯云计算（北京）有限责任公司

AI 内容识别
图片标签

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于图片标签的 API 概览和 SDK 示例代码。

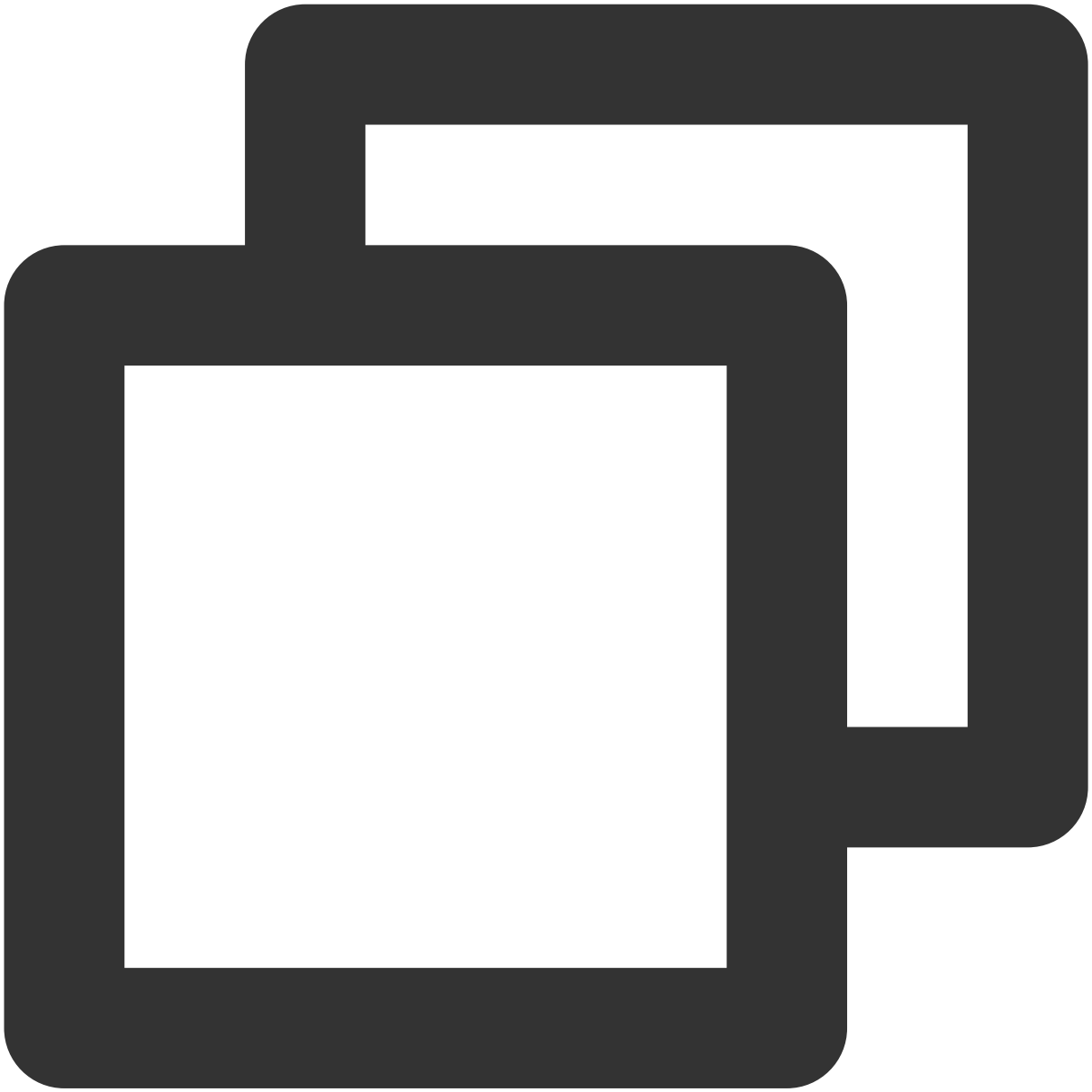
API	操作描述
图片标签	对 COS 存量数据的图片标签识别

图片标签

功能说明

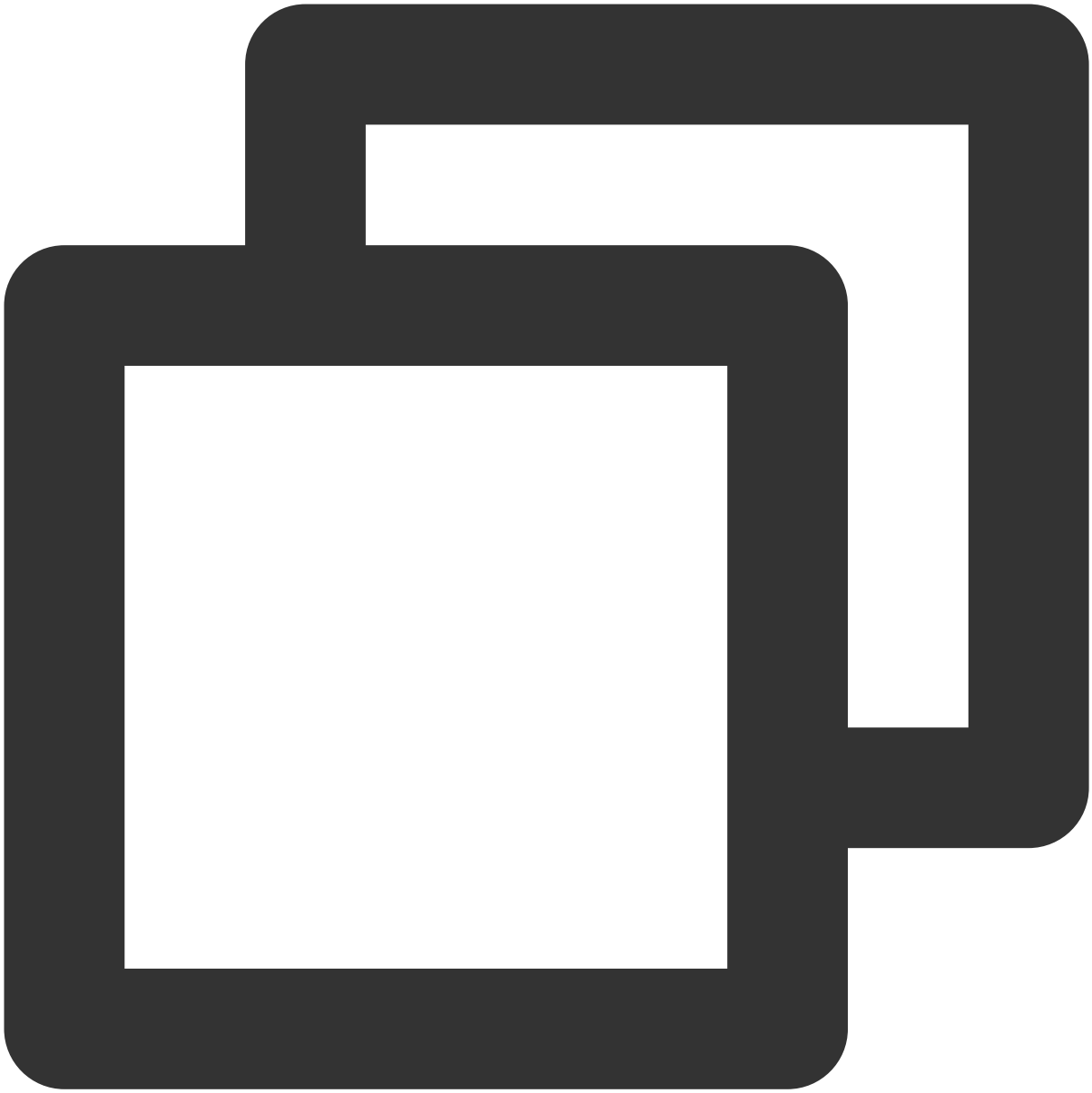
图片标签功能通过借助数据万象的持久化处理接口，实现对 COS 存量数据的图片标签识别，返回图片中置信度较高的主题标签。图片标签识别请求包属于 GET 请求，请求时需要携带签名。

方法原型



```
ImageLabelResponse getImageLabel(ImageLabelRequest request);
```

请求示例



```
//1.创建任务请求对象
ImageLabelRequest request = new ImageLabelRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("demo-123456789");
request.setObjectKey("1.png");
//3.调用接口,获取任务响应对象
ImageLabelResponse response = client.getImageLabel(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
bucketName	Request	Bucket 的命名规则为 BucketName-APPID，详情请参见存储桶概述	String	是
objectKey	Request	图片文件在 bucket 中的位置	String	是

返回结果说明

成功：成功则返回 ImageLabelResponse 实例，内含结果内容。

失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

版权所有：腾讯云计算（北京）有限责任公司

汽车识别

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于汽车识别的 API 概览和 SDK 示例代码。

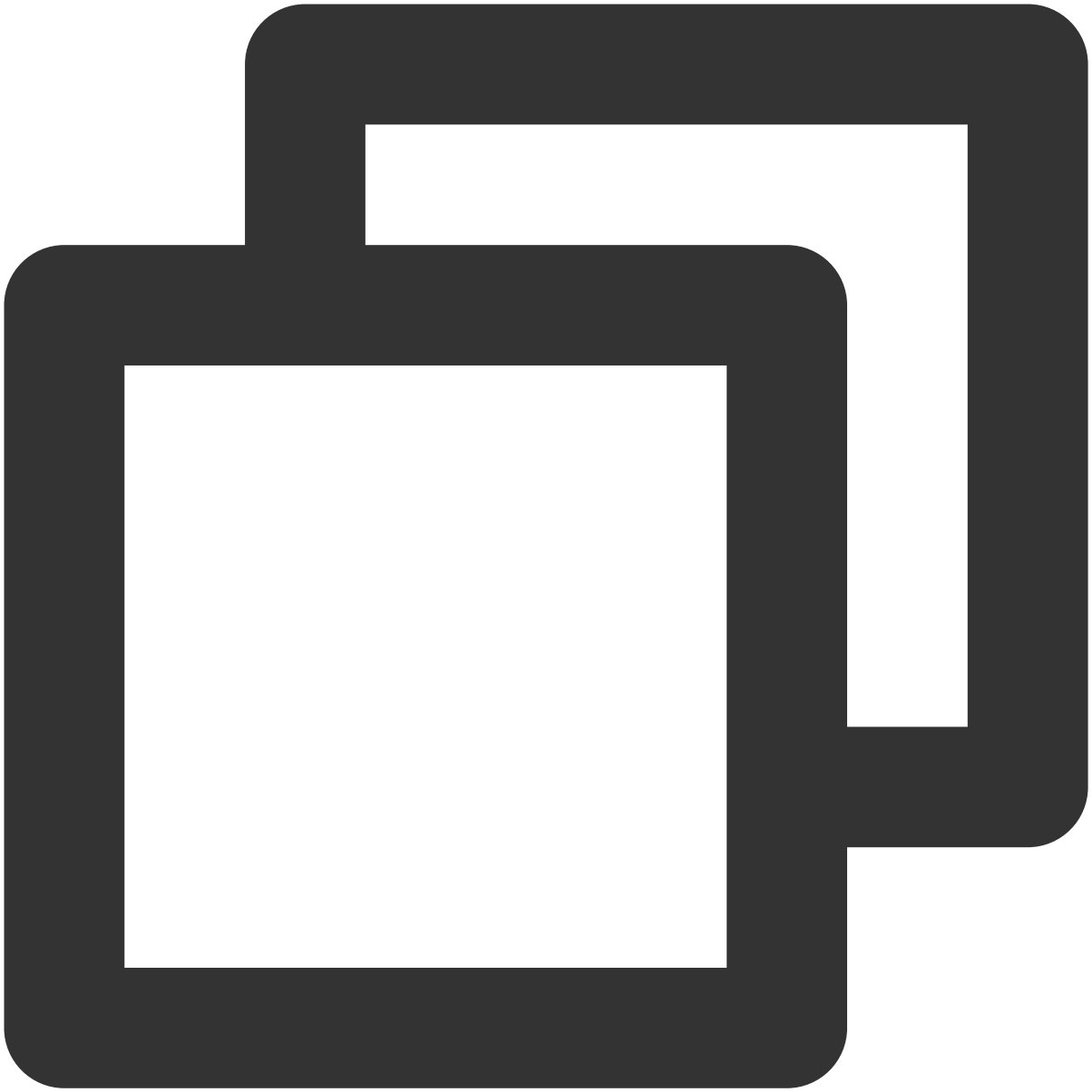
API	操作描述
车辆车牌检测	对图片中的车辆进行识别

车辆车牌检测

功能说明

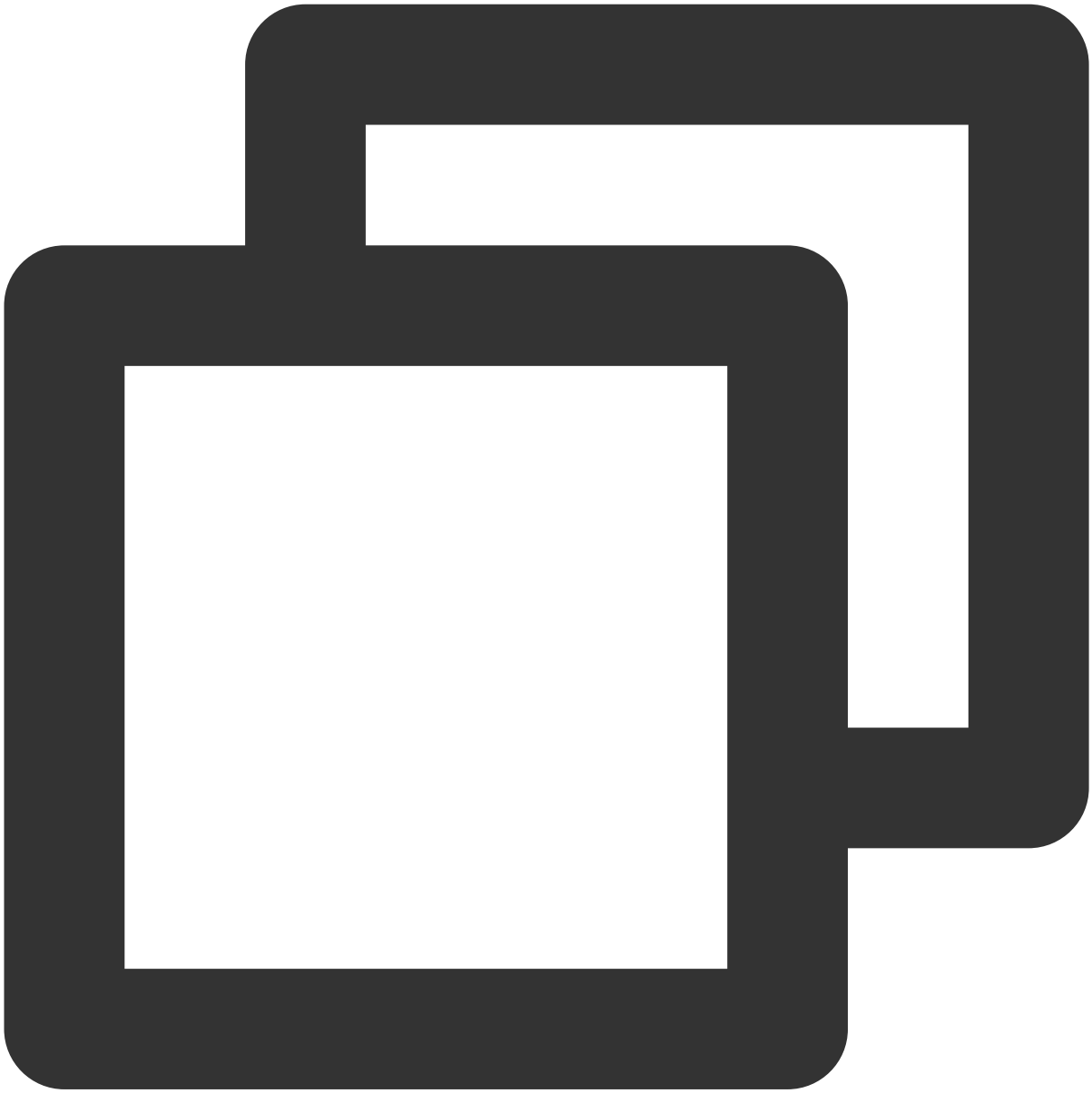
车辆车牌检测功能为同步请求方式，您可以通过本接口检测图片中的车辆，识别出车辆的品牌、颜色、位置、车牌位置等信息。该接口属于 GET 请求。

方法原型



```
DetectCarResponse detectCar(DetectCarRequest request);
```

请求示例



```
//1.创建任务请求对象
DetectCarRequest request = new DetectCarRequest();
//2.添加请求参数 参数详情请见api接口文档
//2.1设置请求bucket
request.setBucketName("demobucket-1234567890");
//2.2设置bucket中的图片位置
request.setObjectKey("car.jpg");
DetectCarResponse response = client.detectCar(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
bucketName	Request	Bucket 的命名规则为 BucketName-APPID，详情请参见存储桶概述	String	是
objectKey	Request	图片文件在 bucket 中的位置	String	是

返回结果说明

成功：成功则返回 DetectCarResponse 任务结果对象。

失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

媒体处理
模板操作

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象媒体处理任务的相关 API 概览以及 SDK 示例代码,此处以动图任务举例。

API	操作名	操作描述
CreateMediaTemplate	创建模板	用于创建一个新的模板
DeleteMediaTemplate	删除模板	删除一个模板
DescribeMediaTemplates	查询模板	查询模板列表
UpdateMediaTemplate	修改模板	修改一个模板

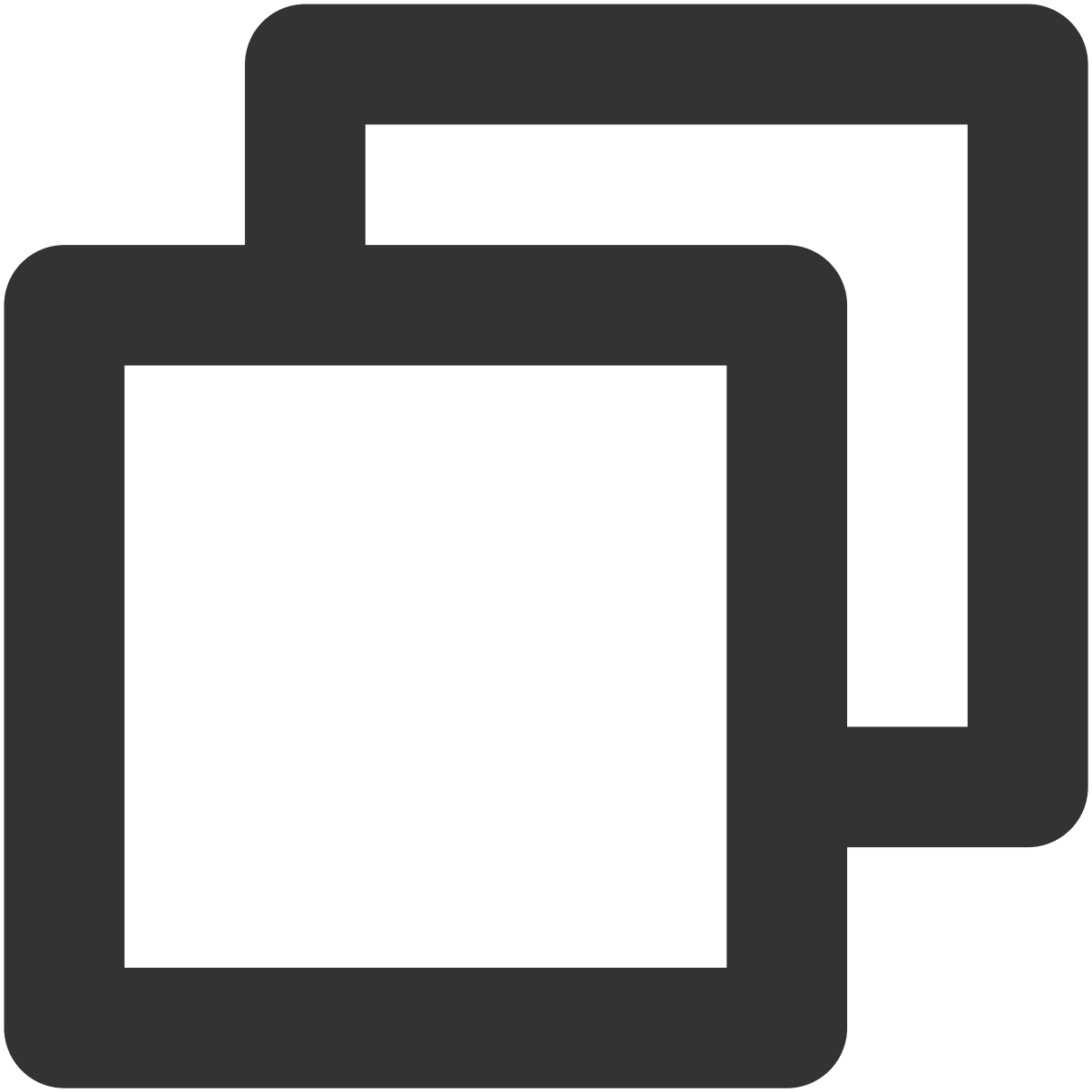
基本操作

创建模板

功能说明

用于创建一个新的模板。

方法原型



```
public MediaTemplateResponse createMediaTemplate(MediaTemplateRequest request);
```

参数说明

Request 的具体参数描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Tag	Request	模板类型：Animation 动图，Snapshot 截图，Transcode 转码，Watermark 水印，SmartCover	String	是

		智能封面		
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
Container	Request	容器格式	Container	是
Video	Request	视频信息	Container	否
TimeInterval	Request	时间区间	Container	否
Snapshot	Request	截图	Container	否
Watermark	Request	水印	Container	否
Audio	Request	音频信息	Container	否
TransConfig	Request	转码配置	Container	否

Request 节点 Container 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Format	Request.Container	容器格式：gif，hgif，webp hgif 为高质量 gif，即清晰度比较高的 gif 格式图	String	是

Request 节点 Video 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Codec	Request.Video	编解码格式	String	是	无	gif、webp
Width	Request.Video	宽	String	否	视频原始宽度	值范围：[128，4096] 单位：px 若只设置 Width 时，按照视频原始比例计算 Height
Height	Request.Video	高	String	否	视频原始高度	值范围：[128，4096] 单位：px 若只设置 Height 时，按照视频原始比例计算 Width
Fps	Request.Video	帧率	String	否	视频原始帧率	值范围：(0，60] 单位：fps

版权所有：腾讯云计算（北京）有限责任公司

						如果不设置，那么播放速度按照原来的时间戳。这里设置 fps 为动图的播放帧率。
AnimateOnlyKeepKeyFrame	Request.Video	动图只保留关键帧	String	否	无	true、false 动图保留关键帧参数
AnimateTimeIntervalOfFrame	Request.Video	动图抽帧间隔时间	String	否	无	（0，视频时长] 动图抽帧时间间隔 若设置 TimeInterval.Duration，则小于该值
AnimateFramesPerSecond	Request.Video	Animation 每秒抽帧帧数	String	否	无	(0，视频帧率) 动图抽帧频率 优先级：AnimateFramesPerSecond > AnimateOnlyKeepKeyFrame > AnimateTimeIntervalOfFrame
Quality	Request.Video	设置相对质量	String	否	无	[1, 100) webp 图像质量设定生效，gif 没有质量参数

Request 节点 TimeInterval 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Start	Request.TimeInterval	开始时间	String	否	0	[0 视频时长] 单位为秒 支持 float 格式，执行精度精确到毫秒
Duration	Request.TimeInterval	持续时间	String	否	视频时长	[0 视频时长] 单位为秒 支持 float 格式，执行精度精确到毫秒

Container 类型 Snapshot 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Mode	Request.Snapshot	截图	String	是	Interval	值范围：{Interval, Average}

		模式				Interval 表示间隔模式 Average 表示平均模式 Interval 模式：Start，TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片 Average 模式：Start，Count 参数生效。表示从 Start 开始到视频结束，按平均间隔截取共 Count 张图片
Start	Request.Snapshot	开始时间	String	是	0	[0 视频时长] 单位为秒 支持 float 格式，执行精度精确到毫秒
TimeInterval	Request.Snapshot	截图频率	String	否	无	(0 3600] 单位为秒 支持 float 格式，执行精度精确到毫秒
Count	Request.Snapshot	截图数量	String	是	无	(0 10000]
Width	Request.Snapshot	宽	String	否	视频原始宽度	值范围：[128，4096] 单位：px 若只设置 Width 时，按照视频原始比例计算 Height
Height	Request.Snapshot	高	String	否	视频原始高度	值范围：[128，4096] 单位：px 若只设置 Height 时，按照视频原始比例计算Width

Container 类型 Watermark 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Type	Request.Watermark	水印类型	String	是	无	Text：文字水印、 Image：图片水印
Pos	Request.Watermark	基准位	String	是	无	TopRight、TopLeft、

版权所有：腾讯云计算（北京）有限责任公司

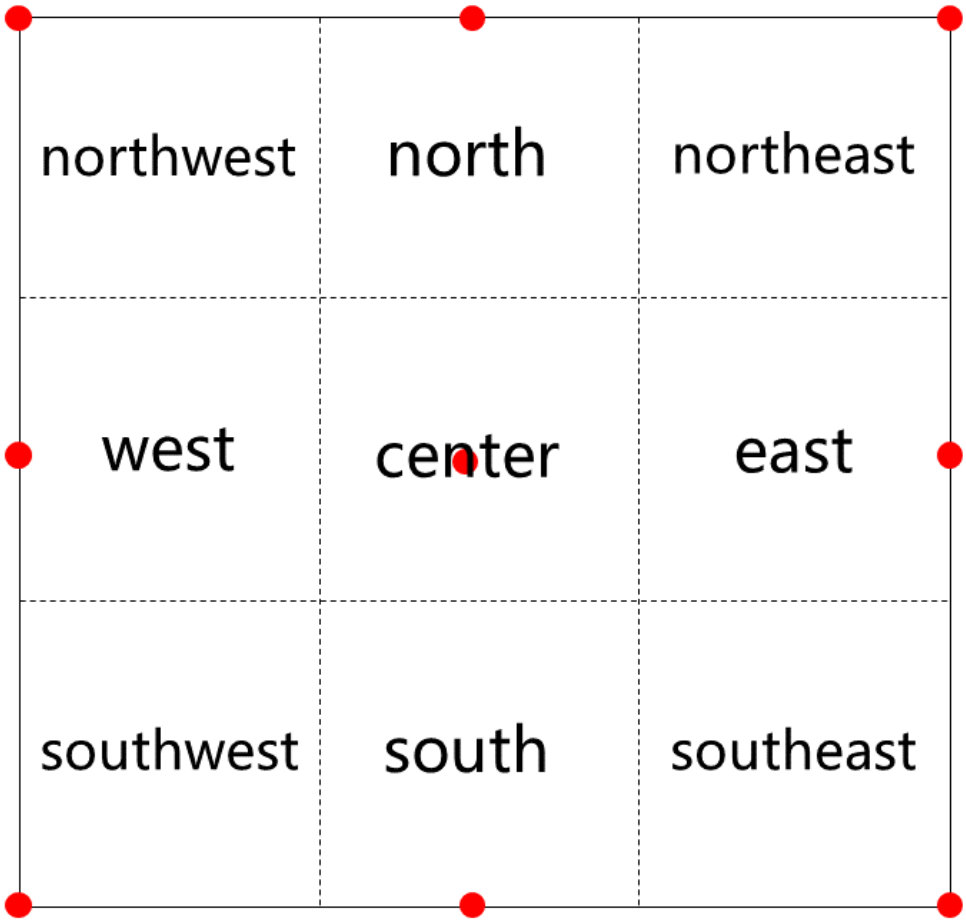
		置				BottomRight、 BottomLeft
LocMode	Request.Watermark	偏移方式	String	是	无	Relativity：按比例、 Absolute：固定位置
Dx	Request.Watermark	水平偏移	String	是	无	当 locMode 为 Relativity 时， 单位为%，取值范围[0 100] 当 locMode 为 Absolute 时， 单位为px，取值范围[0 4096]
Dy	Request.Watermark	垂直偏移	String	是	无	当 locMode 为 Relativity 时， 单位为%，取值范围[0 100] 当 locMode 为 Absolute 时， 单位为px，取值范围[0 4096]
StartTime	Request.Watermark	水印开始时间	String	否	0	[0 视频时长] 单位为秒 支持 float 格式，执行精度精确到毫秒
EndTime	Request.Watermark	水印结束时间	String	否	视频结束时间	[0 视频时长] 单位为秒 支持 float 格式，执行精度精确到毫秒
Image	Request.Watermark	图片水印节点	Container	否	无	无
Text	Request.Watermark	文本水印节点	Container	否	无	无

Container 类型 Image 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Url	Request.Watermark.Image	水印图地址	String	是	无	同 bucket 的水印图片地址

Mode	Request.Watermark.Image	尺寸模式	String	是	无	Original：原有尺寸 Proportion：按比例 Fixed：固定大小
Width	Request.Watermark.Image	宽	String	否	无	当 Mode 为 Original，水印图宽 当 Mode 为 Proportion，单位为%，取值范围：[1 100] 当 Mode 为 Fixed，单位为px，取值范围：[1，4096] 若只设置 Width 时，按照视频原始比例计算 Height
Height	Request.Watermark.Image	高	String	否	无	当 Mode 为 Original，水印图高 当 Mode 为 Proportion，单位为%，取值范围：[1 100] 当 Mode 为 Fixed，单位为px，取值范围：[1，4096] 若只设置 Height 时，按照视频原始比例计算 Width
Transparency	Request.Watermark.Image	透明度	String	是	无	值范围：[1 100]，单位%

水印位置说明：



Container 类型 Text 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
FontSize	Request.Watermark.Text	字体大小	String	是	无	值范围：[5 100]，单位为px
FontType	Request.Watermark.Text	字体类型	String	是	无	参考下表
FontColor	Request.Watermark.Text	字体颜色	String	是	无	格式：0xRRGGBB
Transparency	Request.Watermark.Text	透明度	String	是	无	值范围：[1 100]，单位为%
Text	Request.Watermark.Text	水印内容	String	是	无	长度不超过64个字符，仅支持中文、英文、数字、_、-和*

Text 的 FontType 具体数据描述如下：

字体名称	支持的语言	描述
simfang.ttf	中/英	仿宋
simhei.ttf	中/英	黑体
simkai.ttf	中/英	楷体
simsun.ttc	中/英	宋体
STHeiti-Light.ttc	中/英	华文黑体
STHeiti-Medium.ttc	中/英	华文黑体中
youyuan.TTF	中/英	幼圆
ariblk.ttf	英	无
arial.ttf	英	无
ahronbd.ttf	英	无
Helvetica.dfont	英	无
HelveticaNeue.dfont	英	无

Audio 音频视频支持的格式如下表：

Container	Audio Codecs	Video Codecs
flv/mp4/ts/hls	AAC、MP3	H.264
aac	aac	不支持
mp3	mp3	不支持

Audio 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Codec	Request.Audio	编解码格式	String	否	aac	取值 aac、mp3
Samplerate	Request.Audio	采样率	String	否	44100	单位：Hz 可选 11025、22050、32000、44100、48000、96000

版权所有：腾讯云计算（北京）有限责任公司

						不同的封装，mp3 支持不同的采样率，如下表所示
Bitrate	Request.Audio	原始音频码率	String	否	无	单位：Kbps 值范围：[8，1000]
Channels	Request.Audio	声道数	String	否	无	当 Codec 设置为 aac，支持1、2、4、5、6、8 当 Codec 设置为 mp3，支持1、2
Remove	Request.Audio	是否删除音频流	String	否	false	取值 true、false

Y表示支持这种采样率，N表示不支持。

封装格式/音频采样率	11025	22050	32000	44100	48000	96000
flv	Y	Y	N	Y	N	N
mp4	N	Y	Y	Y	Y	N
avi/hls/ts/mp3	Y	Y	Y	Y	Y	N

TransConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
AdjDarMethod	Request.TransConfig	分辨率调整方式	String	否	none	取值 scale、crop、pad、none 当输出视频的宽高比与原视频不等时，需要此参数进行执行调整方式
IsCheckReso	Request.TransConfig	是否检查分辨率	String	否	false	true、false 当为 false 时，按照配置参数转码
ResoAdjMethod	Request.TransConfig	分辨	String	否	0	取值0、1；0 表示使用原视频分辨率；1表示

版权所有：腾讯云计算（北京）有限责任公司

		率调整方式				返回转码失败 当 IsCheckReso 为 true 时生效
IsCheckVideoBitrate	Request.TransConfig	是否检查视频码率	String	否	false	true、false 当为 false 时，按照配置参数转码
VideoBitrateAdjMethod	Request.TransConfig	视频码率调整方式	String	否	0	取值0、1；0 表示使用原视频码率；1表示返回转码失败 当 IsCheckVideoBitrate 为 true 时生效
IsCheckAudioBitrate	Request.TransConfig	是否检查音频码率	String	否	false	true、false 当为 false 时，按照配置参数转码
AudioBitrateAdjMethod	Request.TransConfig	音频码率调整方式	String	否	0	取值0、1；0 表示使用原音频码率；1表示返回转码失败 当 IsCheckAudioBitrate 为 true 时生效

Video 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制

版权所有：腾讯云计算（北京）有限责任公司

Codec	Request.Video	编解码格式	String	否	H.264	H.264
Width	Request.Video	宽	String	否	视频原始宽度	值范围：[128，4096] 单位：px 若只设置 Width 时，按照视频原始比例计算 Height
Height	Request.Video	高	String	否	视频原始高度	值范围：[128，4096] 单位：px 若只设置 Height 时，按照视频原始比例计算 Width
Fps	Request.Video	帧率	String	否	无	值范围：(0，60] 单位：fps
Remove	Request.Video	是否删除视频流	String	否	false	true、false
Profile	Request.Video	编码级别	String	否	high	支持 baseline、main、high baseline：适合移动设备 main：适合标准分辨率设备 high：适合高分辨率设备 仅 H.264 支持此参数
Bitrate	Request.Video	视频输出文件的码率	String	否	无	值范围：[10，50000] 单位：Kbps
Crf	Request.Video	码率-质量控制因子	String	否	无	值范围：(0，51] 如果设置了 Crf，则 Bitrate 的设置失效
Gop	Request.Video	关键帧间最大帧数	String	否	无	值范围：[0，100000]
Preset	Request.Video	视频算法器预置	String	否	medium	仅 H.264 支持该参数 取值 veryfast、fast、medium、slow、slower
Bufsize	Request.Video	缓冲区大小	String	否	无	值范围：[1000，128000] 单位：Kb
Maxrate	Request.Video	视频码率峰值	String	否	无	值范围：[10，50000] 单位：Kbps
HlsTsTime	Request.Video	hls分片时	String	否	5	(0 视频时长]

版权所有：腾讯云计算（北京）有限责任公司

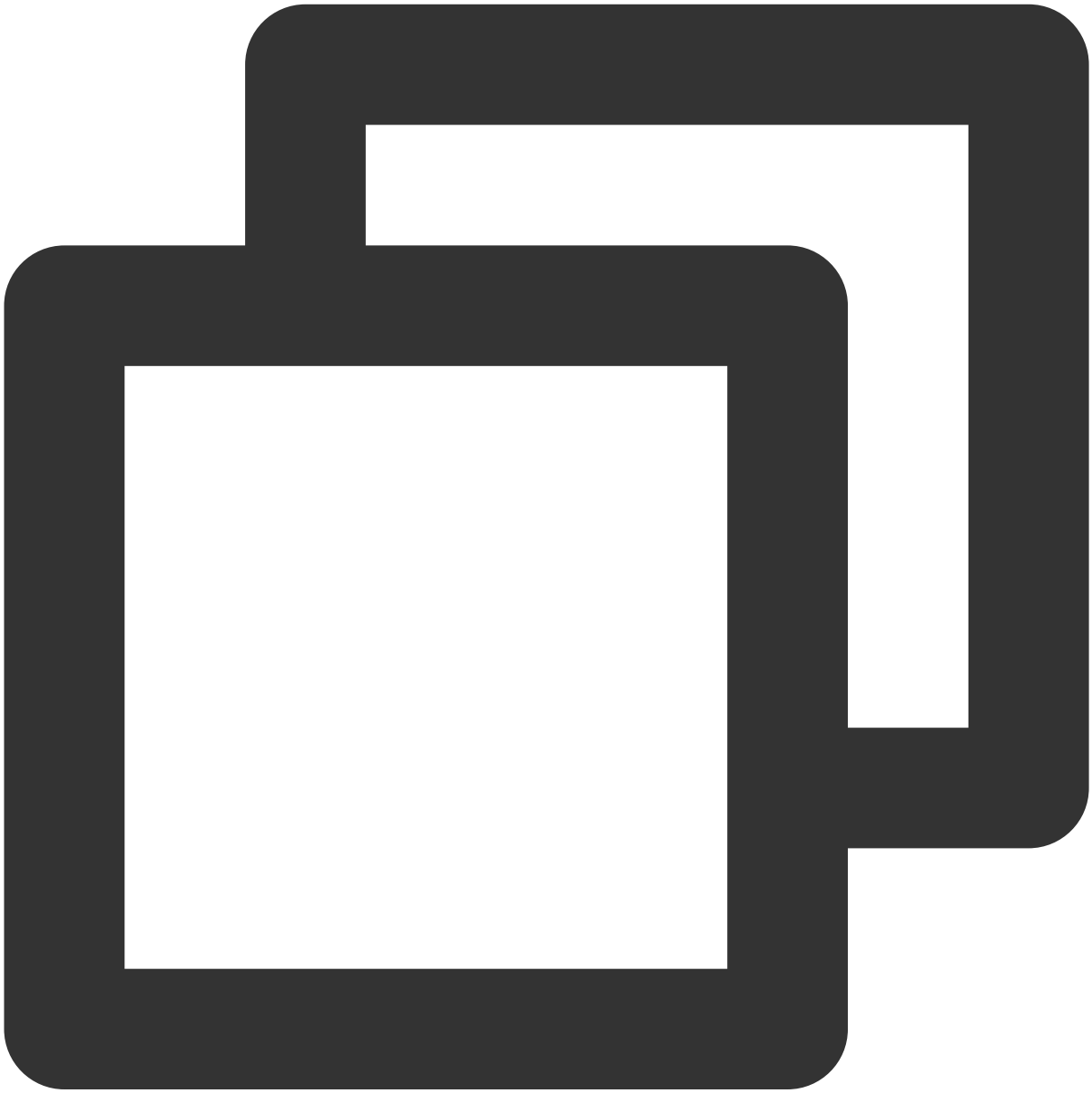
		间				单位为秒
Pixfmt	Request.Video	视频颜色格式	String	否	无	支持 yuv420p、yuv422p、yuv444p、yuvj420p、yuvj422p、yuvj444p

返回结果说明

成功：返回 MediaTemplateResponse 对象响应信息，内含创建的模板详情。

失败：发生错误（如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
//1.创建模板请求对象
MediaTemplateRequest request = new MediaTemplateRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setTag("Animation");
request.setName("TestTemplate40");
request.getContainer().setFormat("gif");
request.getVideo().setCodec("gif");
request.getVideo().setWidth("1280");
request.getVideo().setFps("15");
request.getVideo().setAnimateOnlyKeepKeyFrame("true");
```

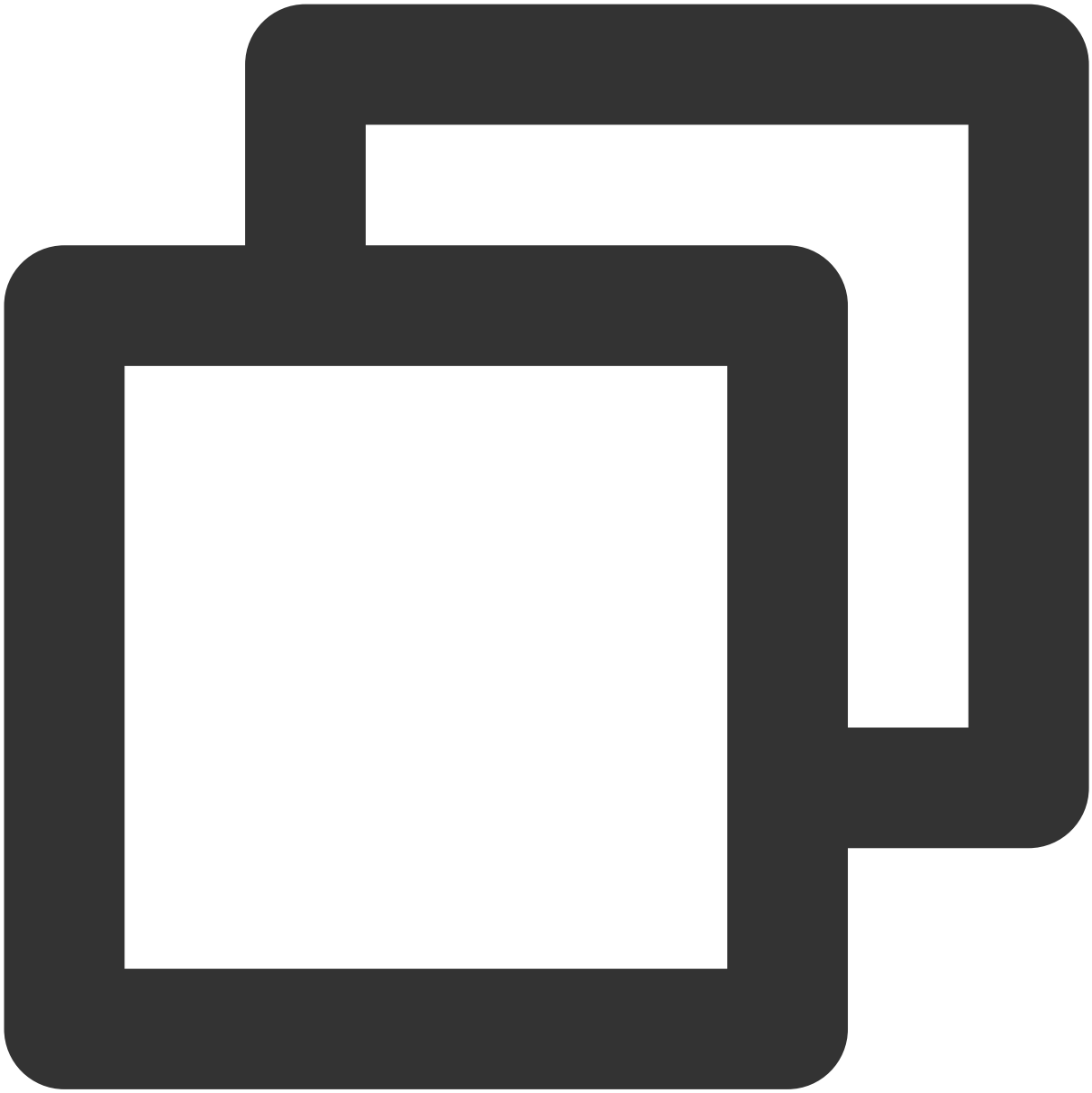
```
request.getTimeInterval().setStart("0");
request.getTimeInterval().setDuration("60");
//3.调用接口,获取模板响应对象
MediaTemplateResponse response = client.createMediaTemplate(request);
```

删除模板

功能说明

删除一个模板

方法原型



```
public Boolean deleteMediaTemplate(MediaTemplateRequest request);
```

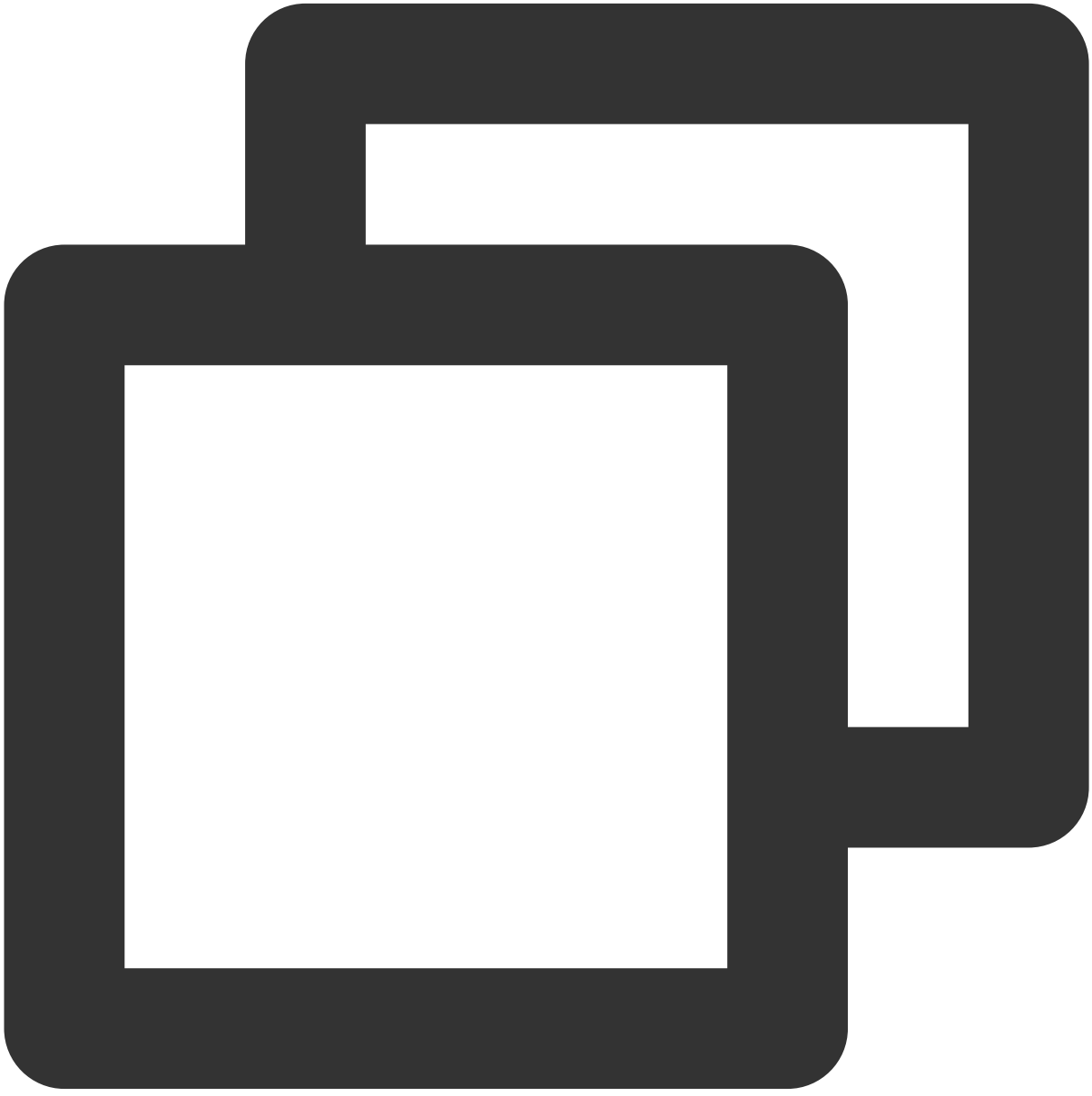
参数说明

参数名称	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
templateId	要取消的模板 id	String	是

返回结果说明

成功：返回一个布尔类型，删除成功则返回 true。
失败：发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
MediaTemplateRequest request = new MediaTemplateRequest();
request.setBucketName("examplebucket-1250000000");
request.setTemplateId("t19c4a60ae1a694621a01f0c7130c*****");
```

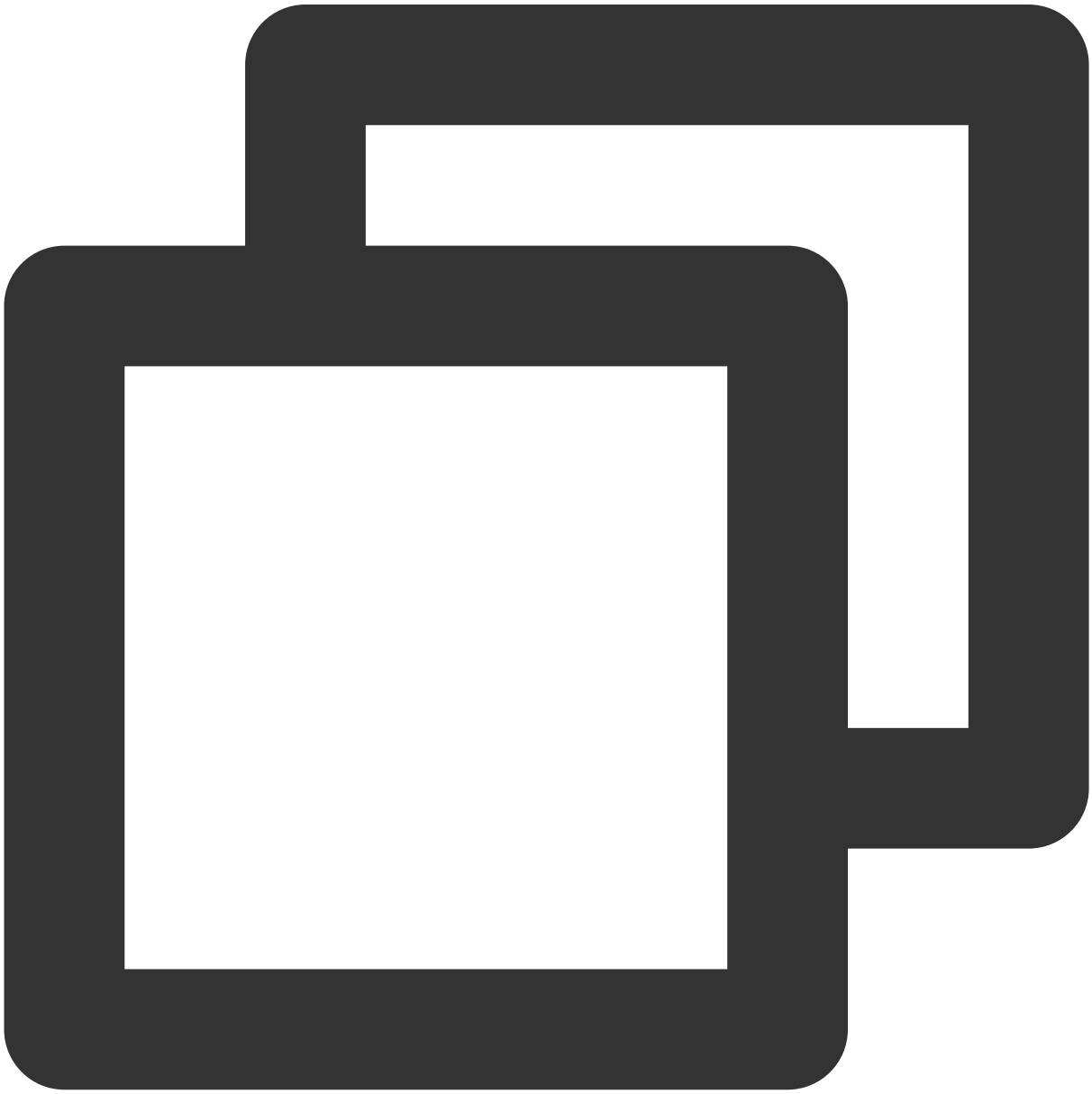
```
Boolean response = client.deleteMediaTemplate(request);
```

查询模板列表

功能说明

查询模板列表

方法原型



```
public MediaJobResponse describeMediaJob(MediaJobsRequest req);
```

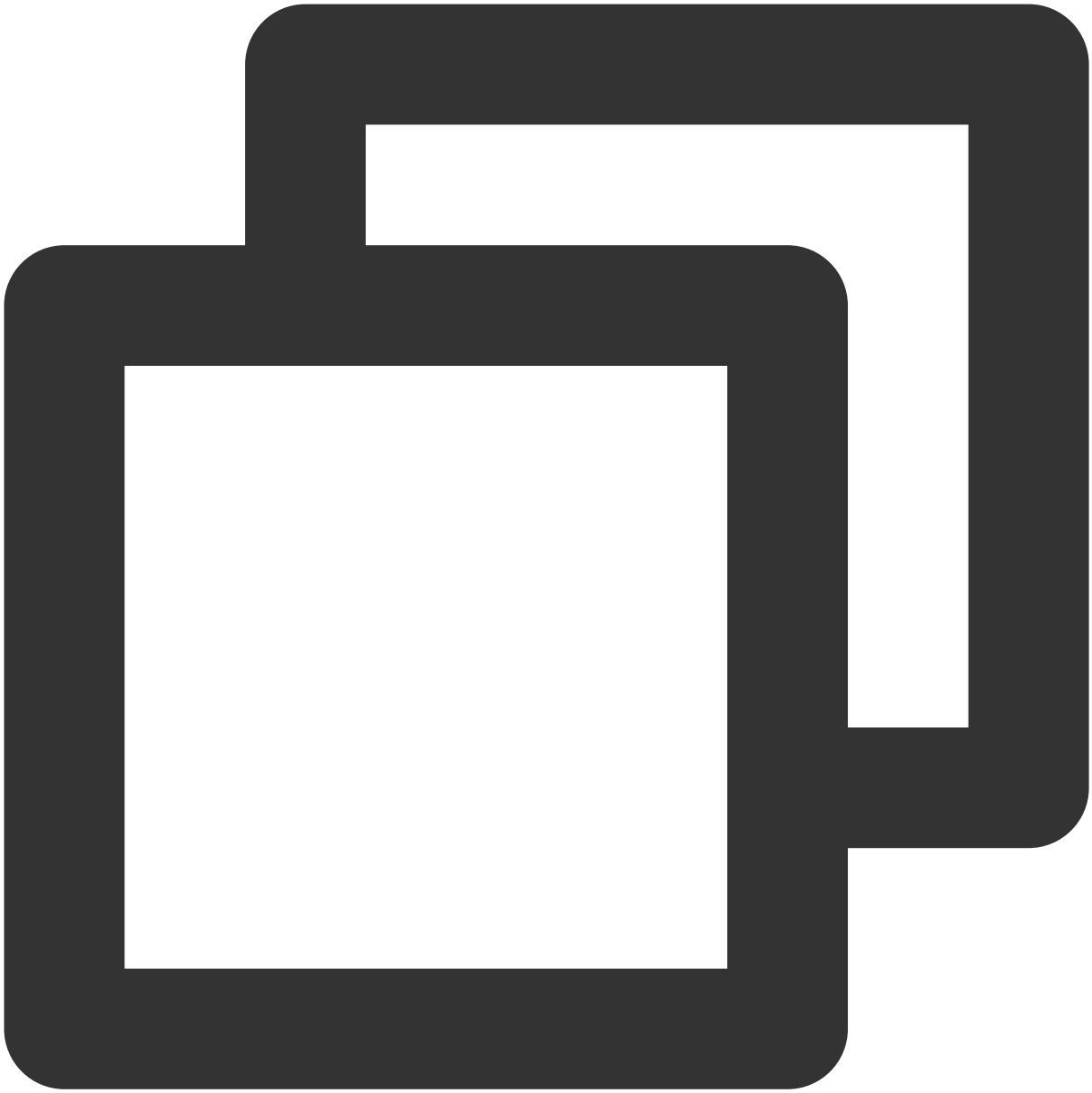
参数说明

参数名称	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
tag	模板 Tag：Animation 动图，Snapshot 截图，Transcode 转码，Watermark 水印，SmartCover 智能封面	String	是
category	Official，Custom，默认值：Custom	String	否
ids	模板 ID，以 `,` 符号分割字符串	String	否
name	模板名称前缀	String	否
pageNumber	第几页	Integer	否
pageSize	每页个数	Integer	否

返回结果说明

成功：返回模板响应包装类，其中包含模板详情集合 templateList。
失败：发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



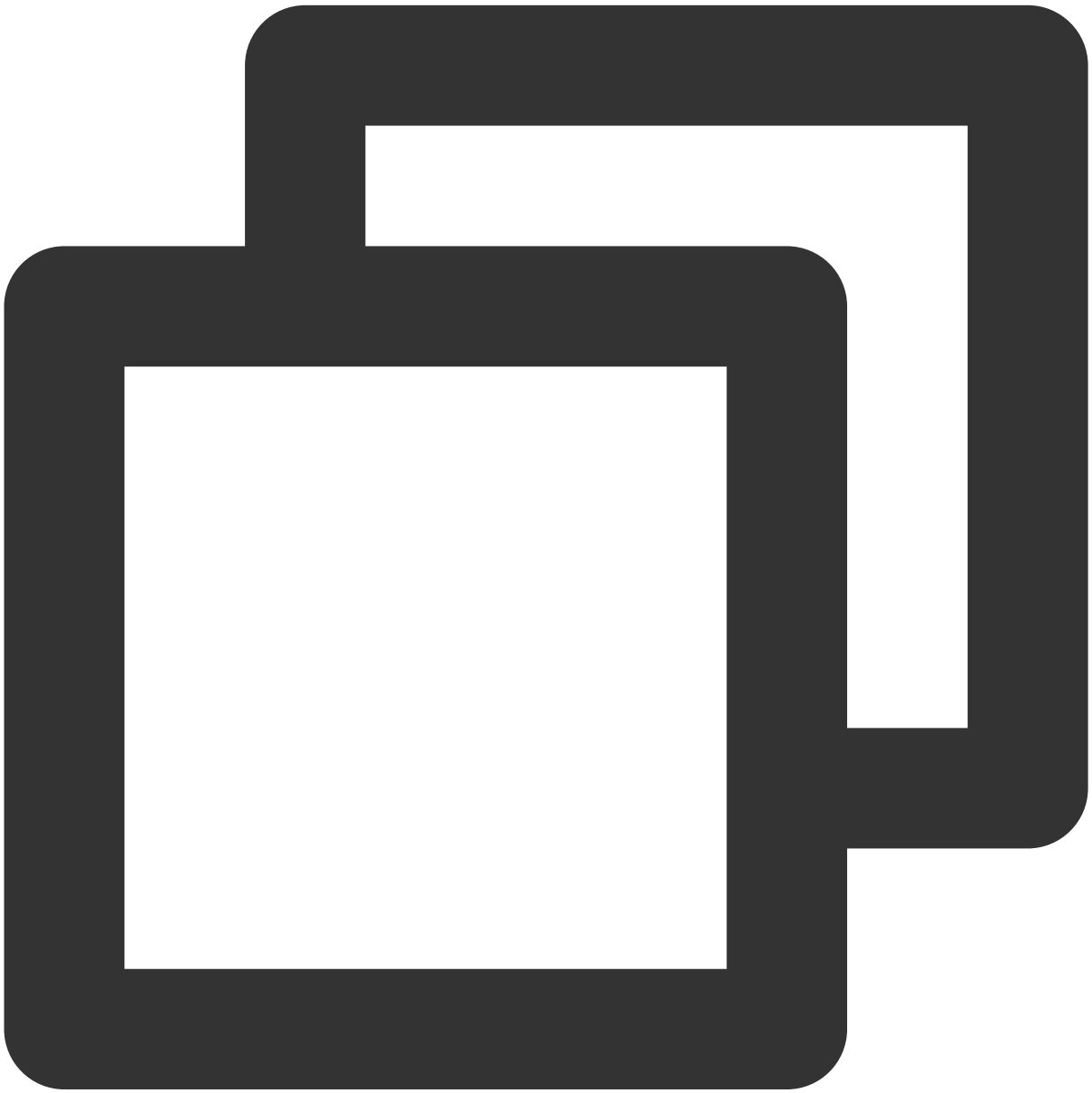
```
//1.创建任务请求对象
MediaTemplateRequest request = new MediaTemplateRequest();
request.setBucketName("examplebucket-1250000000");
MediaListTemplateResponse response = client.describeMediaTemplates(request);
List<MediaTemplateObject> templateList = response.getTemplateList();
```

修改模板

功能说明

修改模板

方法原型



```
public Boolean updateMediaTemplate(MediaTemplateRequest request);
```

参数说明

节点名称（关键字）	描述	类型	必选

bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
templateId	修改的模板 ID	String	是

说明：

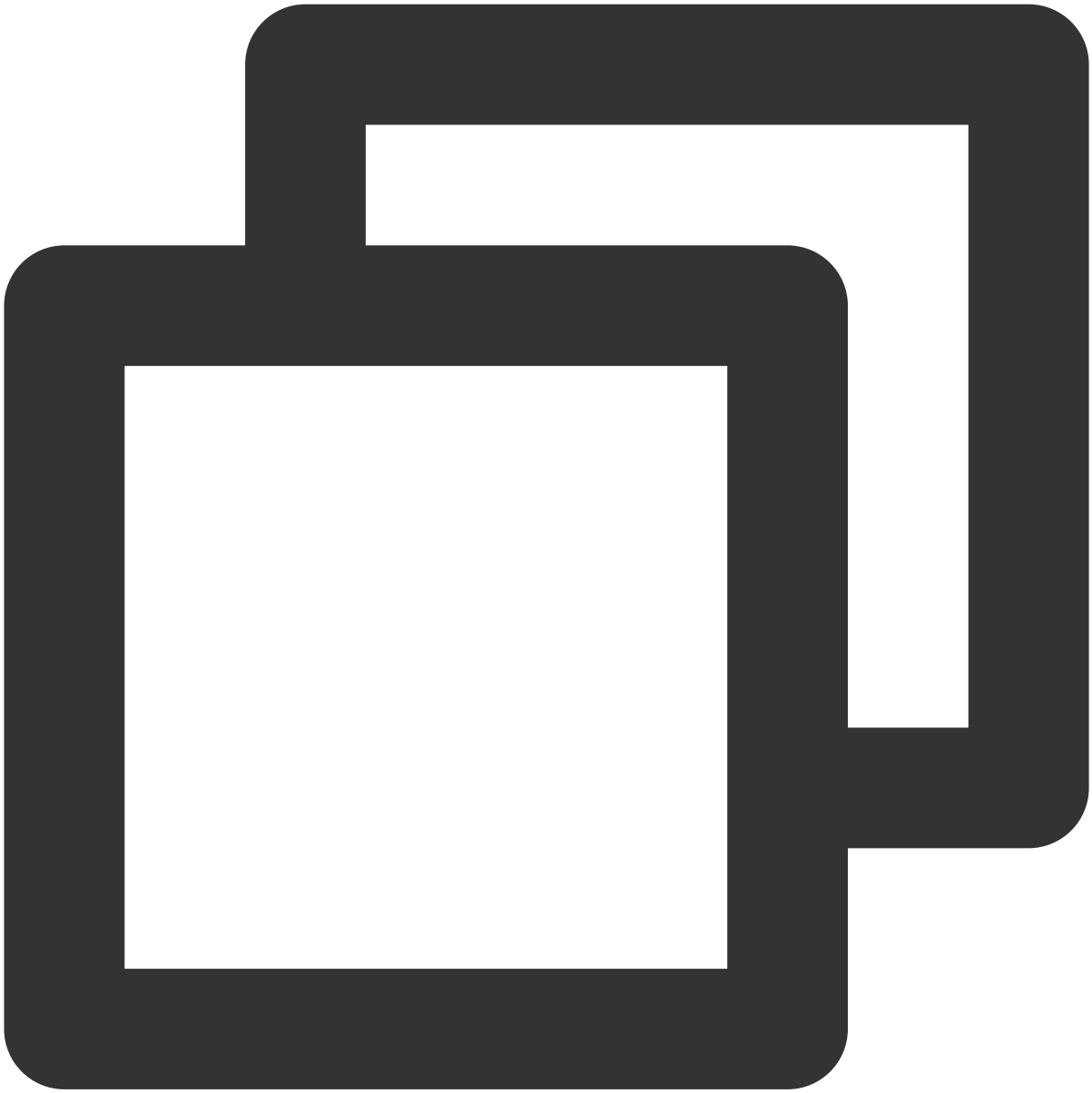
其余参数请参见 创建模板接口，与其参数一致。

返回结果说明

成功： 修改成功则返回 true。

失败： 发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
MediaTemplateRequest request = new MediaTemplateRequest();
request.setBucketName("examplebucket-1250000000");
request.setTemplateId("t19c4a60ae1a694621a01f0c7130*****");
request.setTag("Animation");
request.setName("updateName");
request.getContainer().setFormat("gif");
Boolean aBoolean = client.updateMediaTemplate(request);
```

工作流操作

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象媒体处理工作流的相关 API 概览以及 SDK 示例代码，此处以动图任务举例。

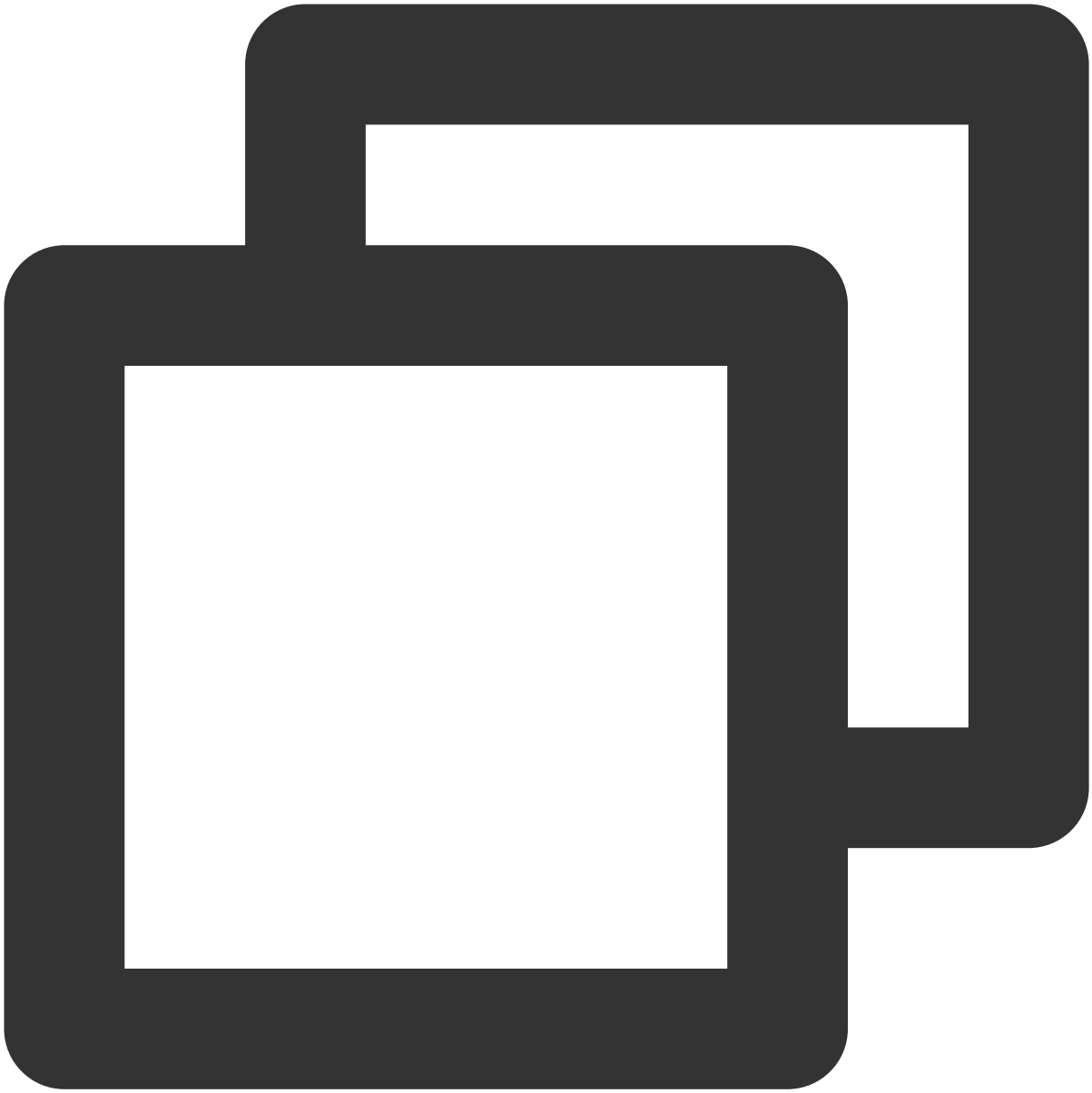
API	操作名	操作描述
DeleteWorkflow	删除工作流	删除一个工作流
DescribeWorkflow	查询工作流	查询一个工作流
DescribeWorkflowExecution	查询工作流实例详情	查询一个工作流实例详情
DescribeWorkflowExecutions	查询工作流详情列表	获取工作流实例列表
TriggerWorkflowList	触发工作流	工作流执行

删除工作流

功能说明

删除一个工作流。

方法原型



```
public Boolean deleteWorkflow(MediaWorkflowListRequest request);
```

参数说明

Request中的具体数据描述如下：

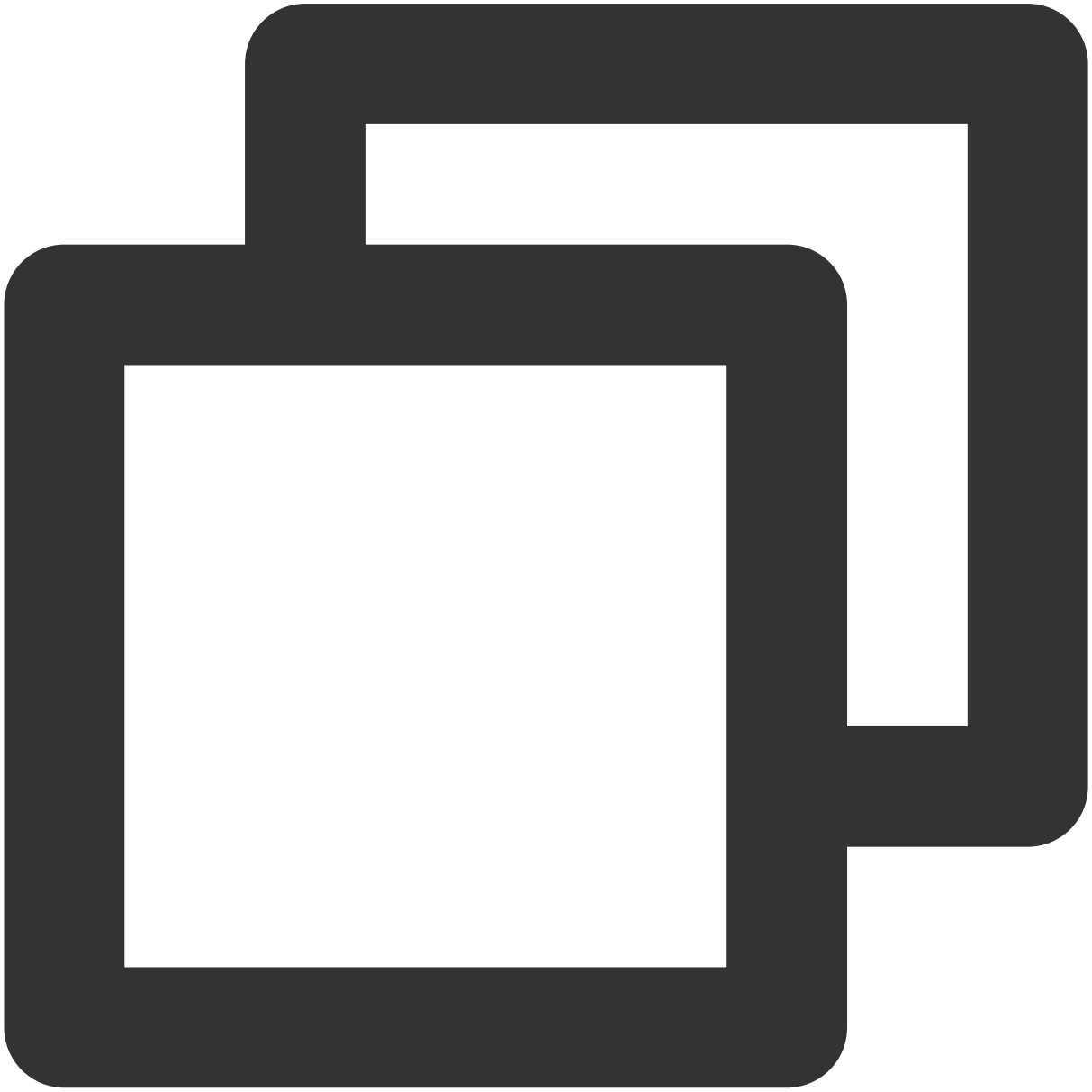
节点名称（关键字）	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是

workflowId	工作流 ID	String	是

返回结果说明

成功：成功则返回 true。
失败：发生错误（如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



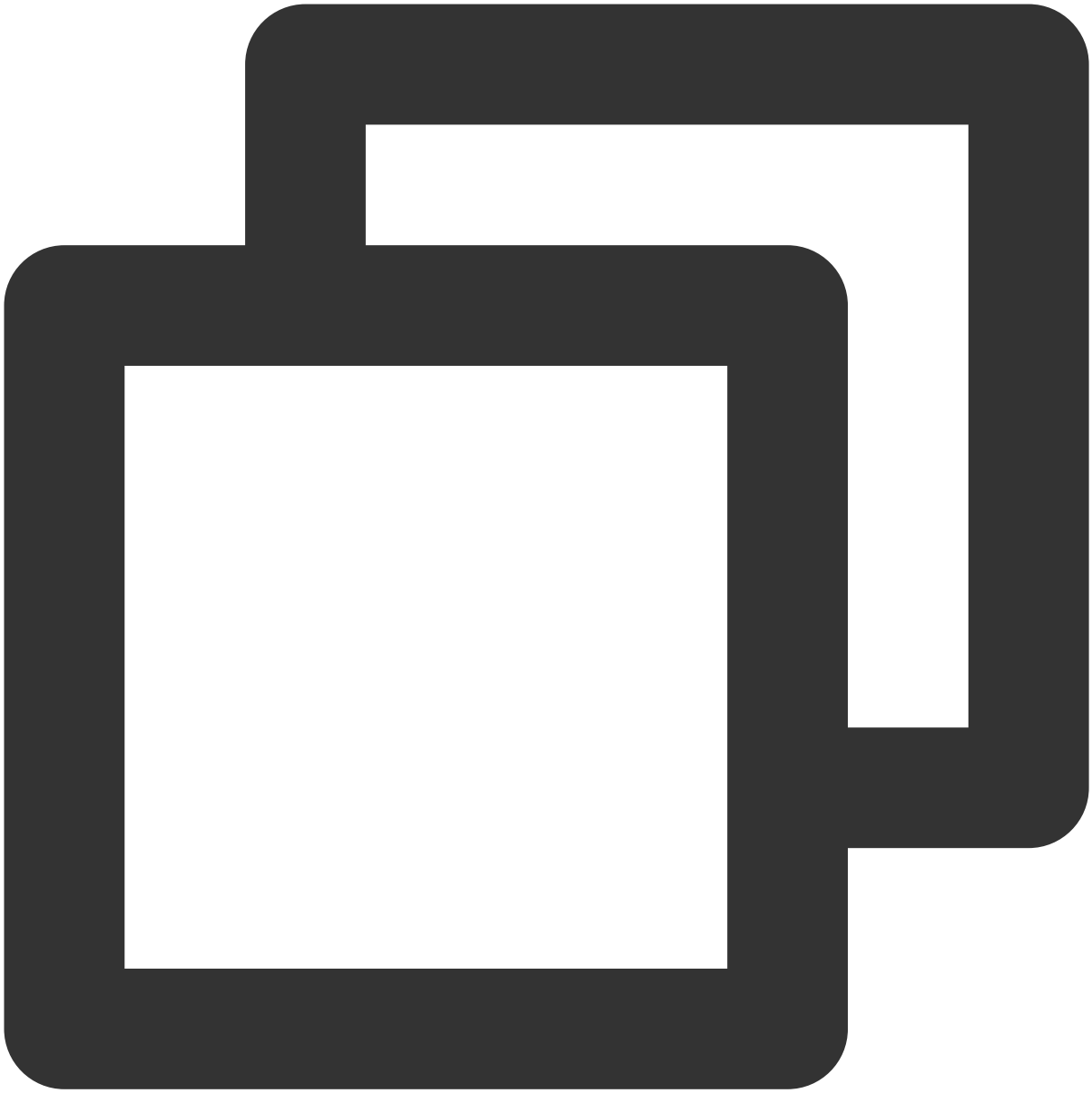
```
//1.创建工作流请求对象
MediaWorkflowListRequest request = new MediaWorkflowListRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setWorkflowId("aaaa");
Boolean response = client.deleteWorkflow(request);
```

查询工作流

功能说明

用于搜索工作流。

方法原型



```
public MediaWorkflowListResponse describeWorkflow(MediaWorkflowListRequest request)
```

参数说明

参数名称	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
ids	工作流 ID，以 `,` 符号分割字符串	String	否

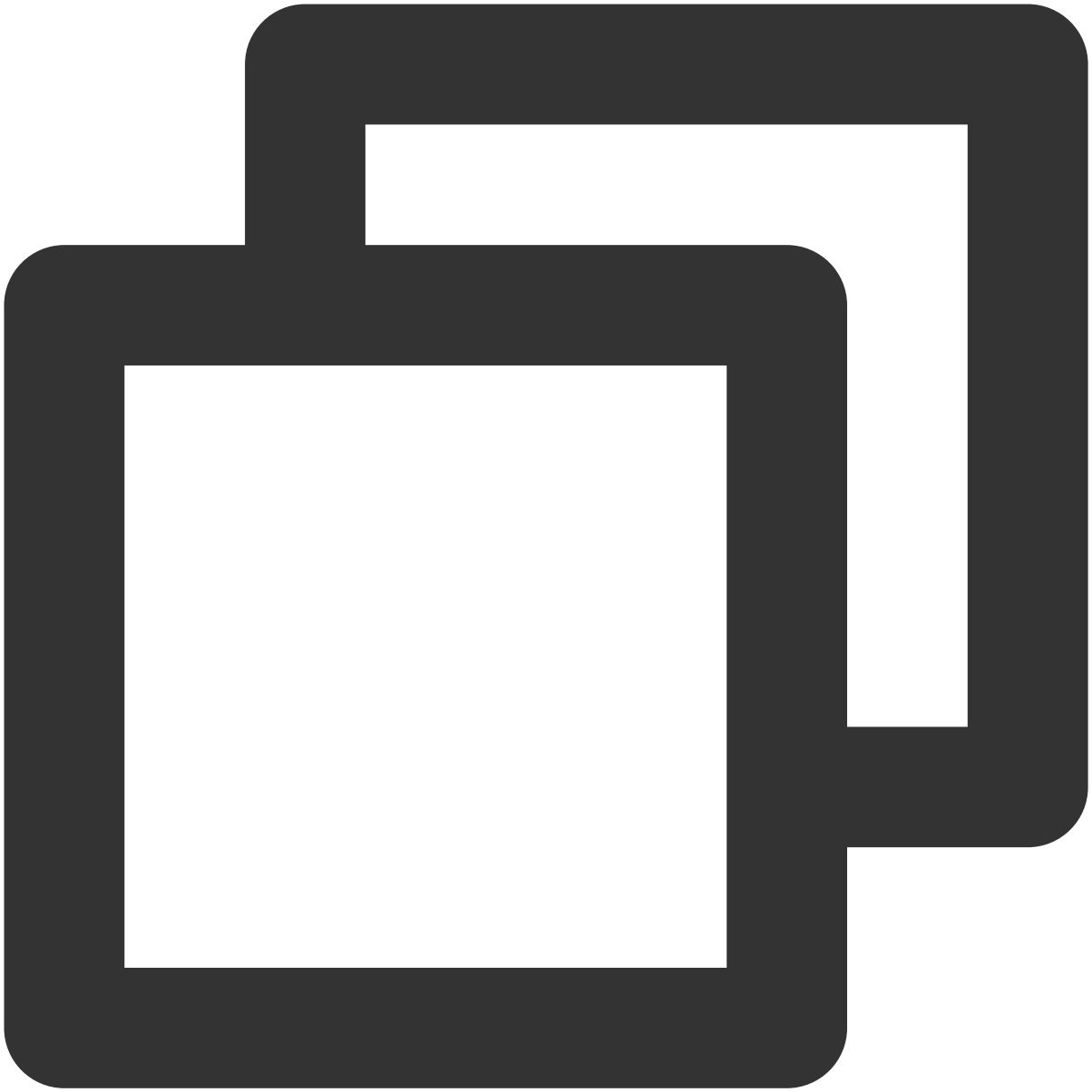
name	工作流名称	String	否
pageNumber	第几页	String	否
pageSize	每页个数	String	否

返回结果说明

成功： 返回工作流集合响应对象，其中包含一个工作流对象集合。

失败： 发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



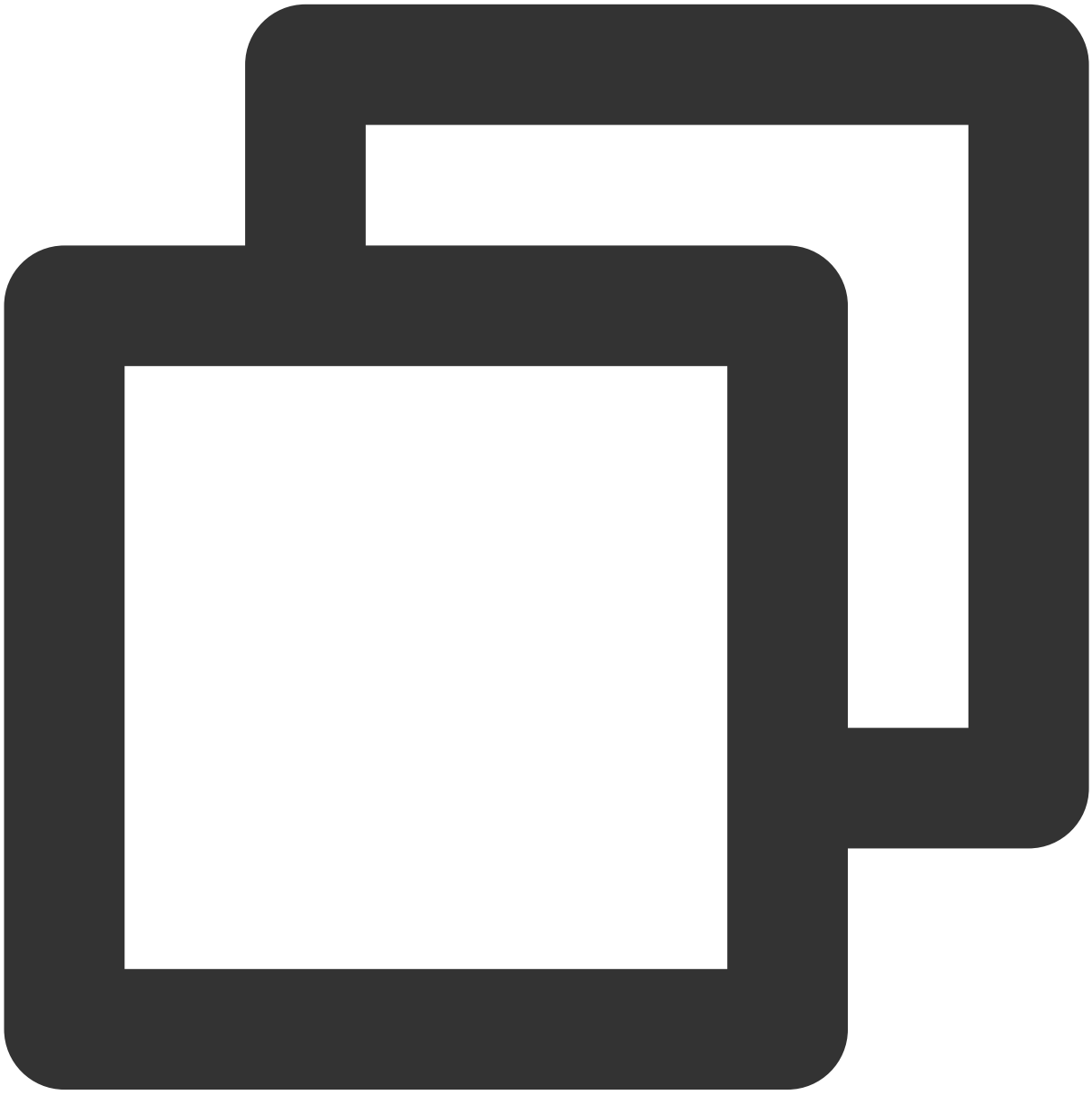
```
//1.创建工作流请求对象
MediaWorkflowListRequest request = new MediaWorkflowListRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
MediaWorkflowListResponse response = client.describeWorkflow(request);
List<MediaWorkflowObject> mediaWorkflowList = response.getMediaWorkflowList();
```

查询工作流实例详情

功能说明

查询工作流实例详情。

方法原型



```
public MediaWorkflowExecutionResponse describeWorkflowExecution(MediaWorkflowListRe
```

参数说明

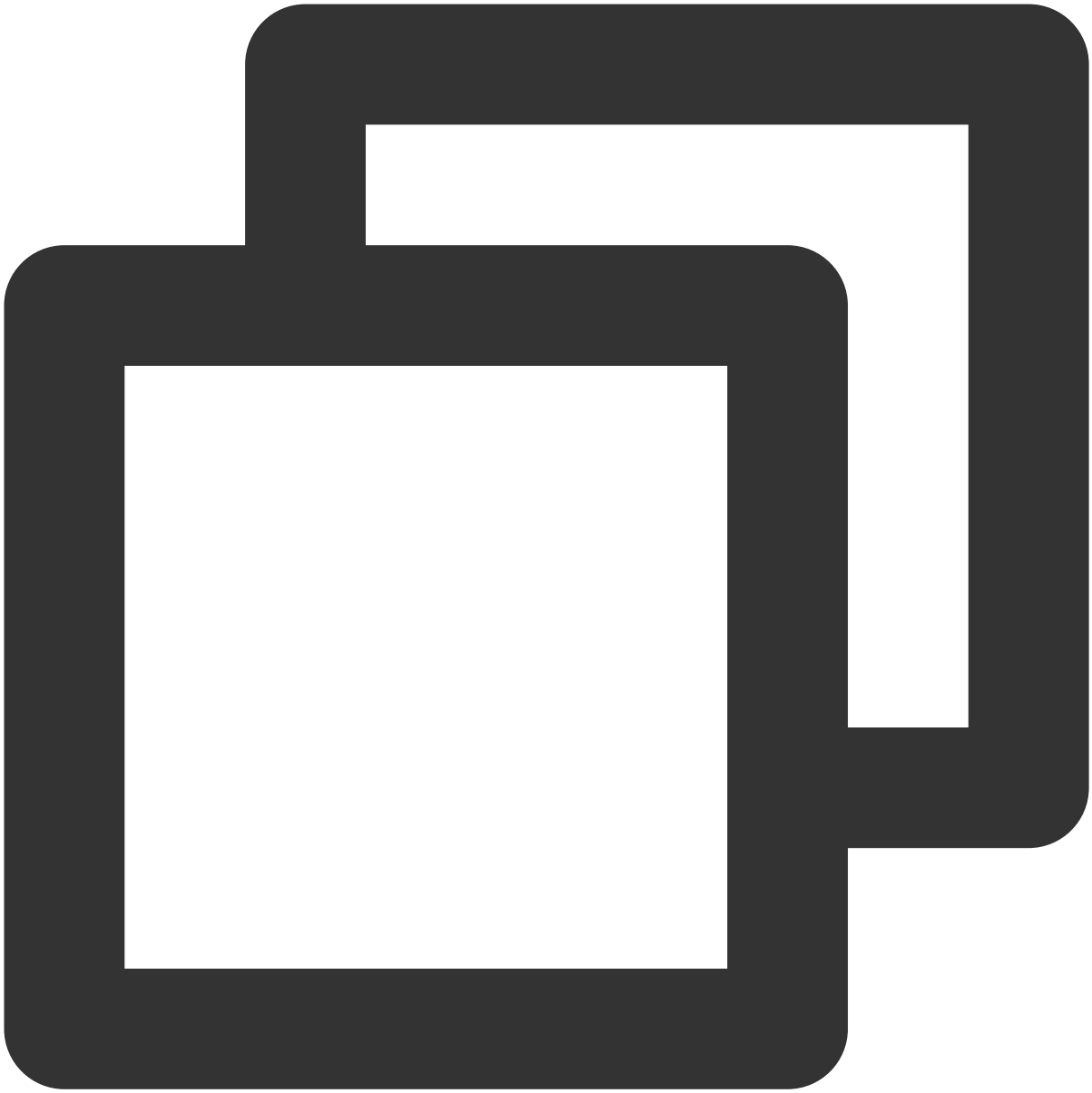
参数名称	描述	类型	必选

bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
runId	工作流实例 ID	String	是

返回结果说明

成功： 返回工作流实例响应包装类，类中包含一个工作流实例详情对象。
失败： 发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



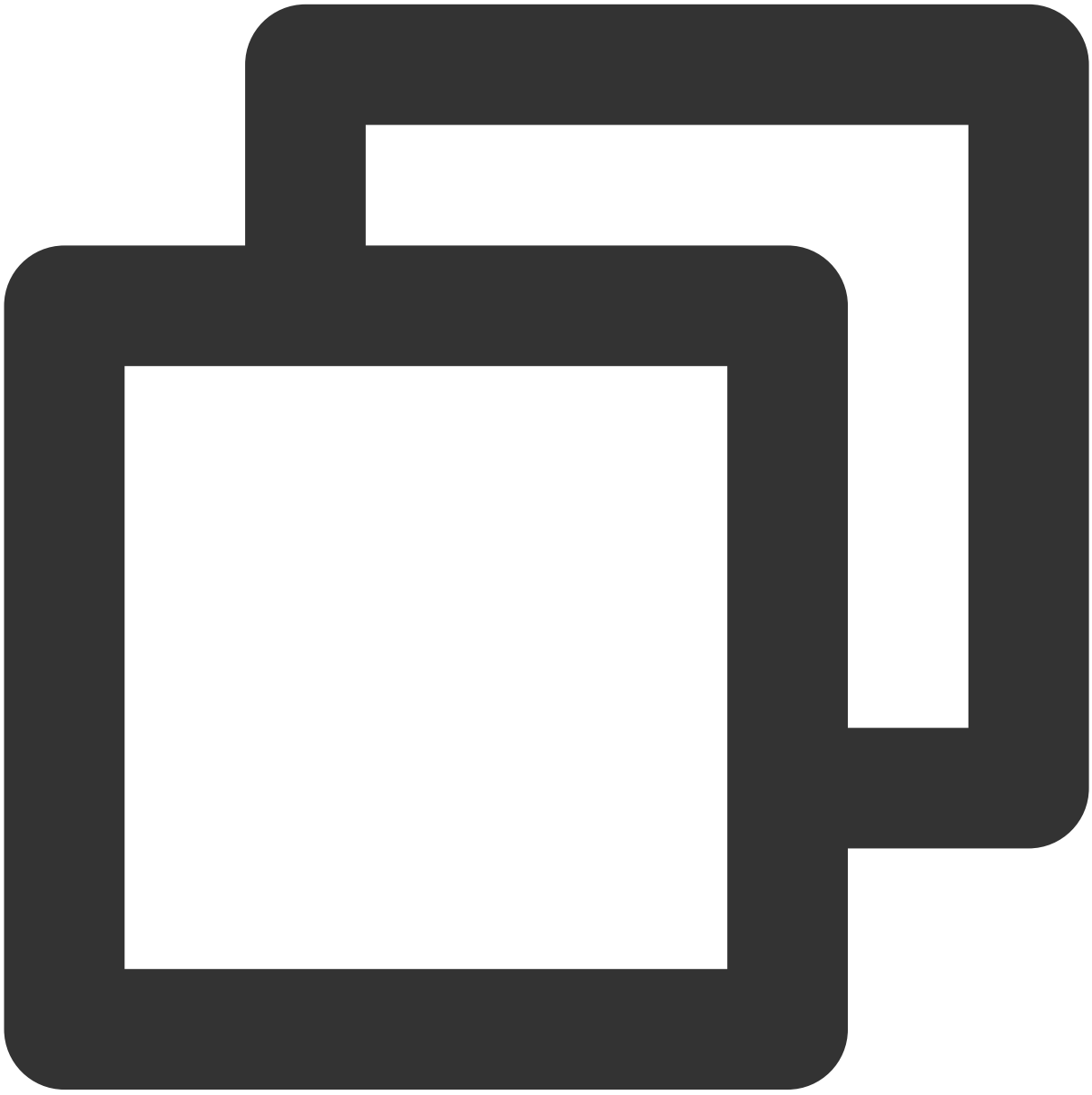
```
//1.创建工作流请求对象
MediaWorkflowListRequest request = new MediaWorkflowListRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setRunId("i34bfd8d7eae711ea89fe525400c******");
MediaWorkflowExecutionResponse response = client.describeWorkflowExecution(request)
```

查询工作流详情列表

功能说明

查询工作流详情列表。

方法原型



```
public MediaWorkflowExecutionsResponse describeWorkflowExecutions(MediaWorkflowList
```

参数说明

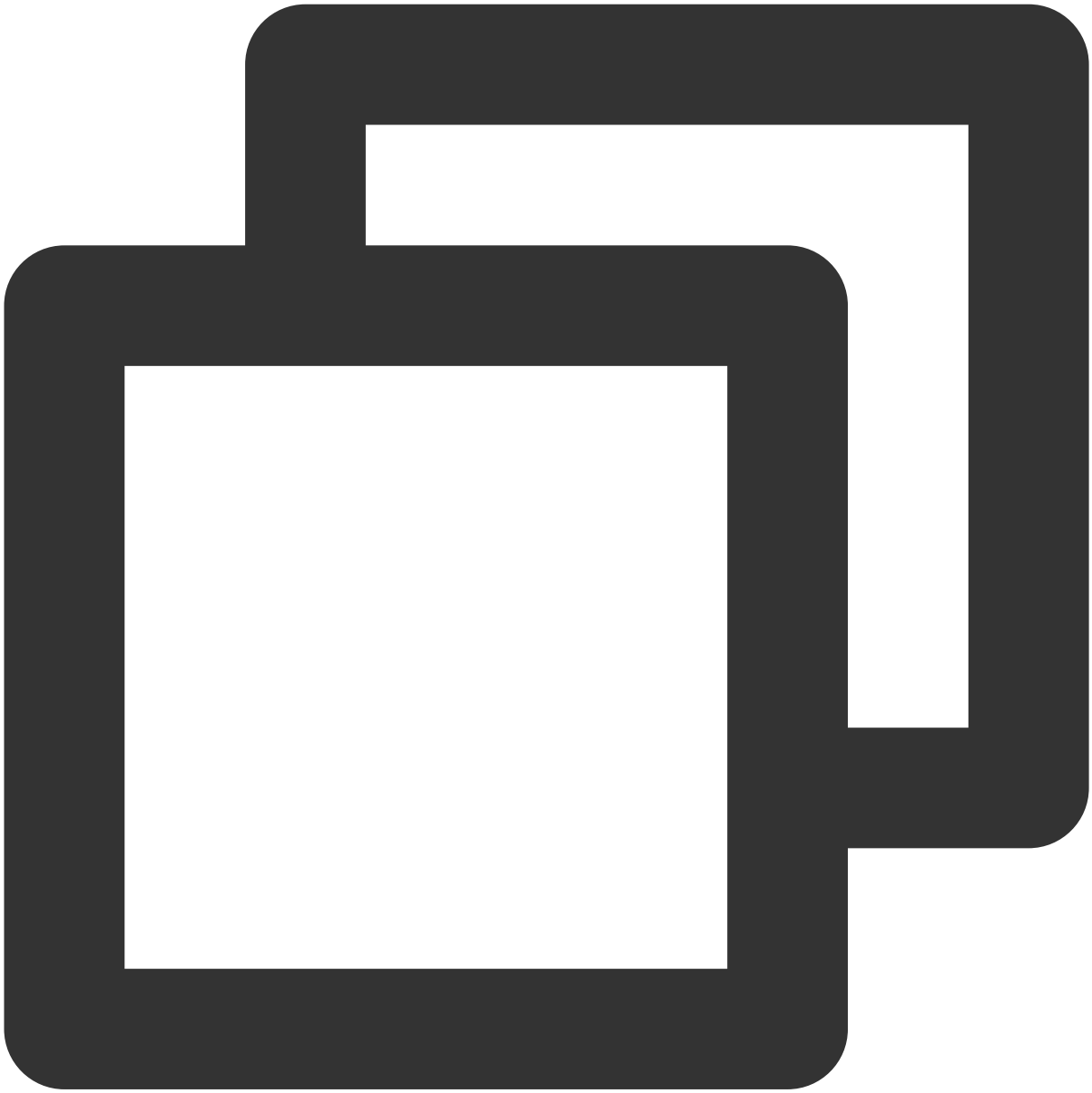
节点名称	描述	类型	必选

bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
workflowId	工作流 ID	String	是
name	文件名称	String	否
orderByTime	Desc 或者 Asc。默认为 Desc	String	否
size	拉取的最大任务数。默认为10。最大为100	String	否
states	工作流实例状态，以 `,` 分割支持多状态 All，Success，Failed，Running，Cancel。默认为 All	String	否
startCreationTime	拉取创建时间大于该时间。格式为：`%Y-%m-%dT%H:%m:%S%z`	String	否
endCreationTime	拉取创建时间小于该时间。格式为：`%Y-%m-%dT%H:%m:%S%z`	String	否
nextToken	请求的上下文，用于翻页。下一页输入 token	String	否

返回结果说明

成功：返回工作流实例集合响应对象，其中包含一个工作流对象实例集合。
失败：发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



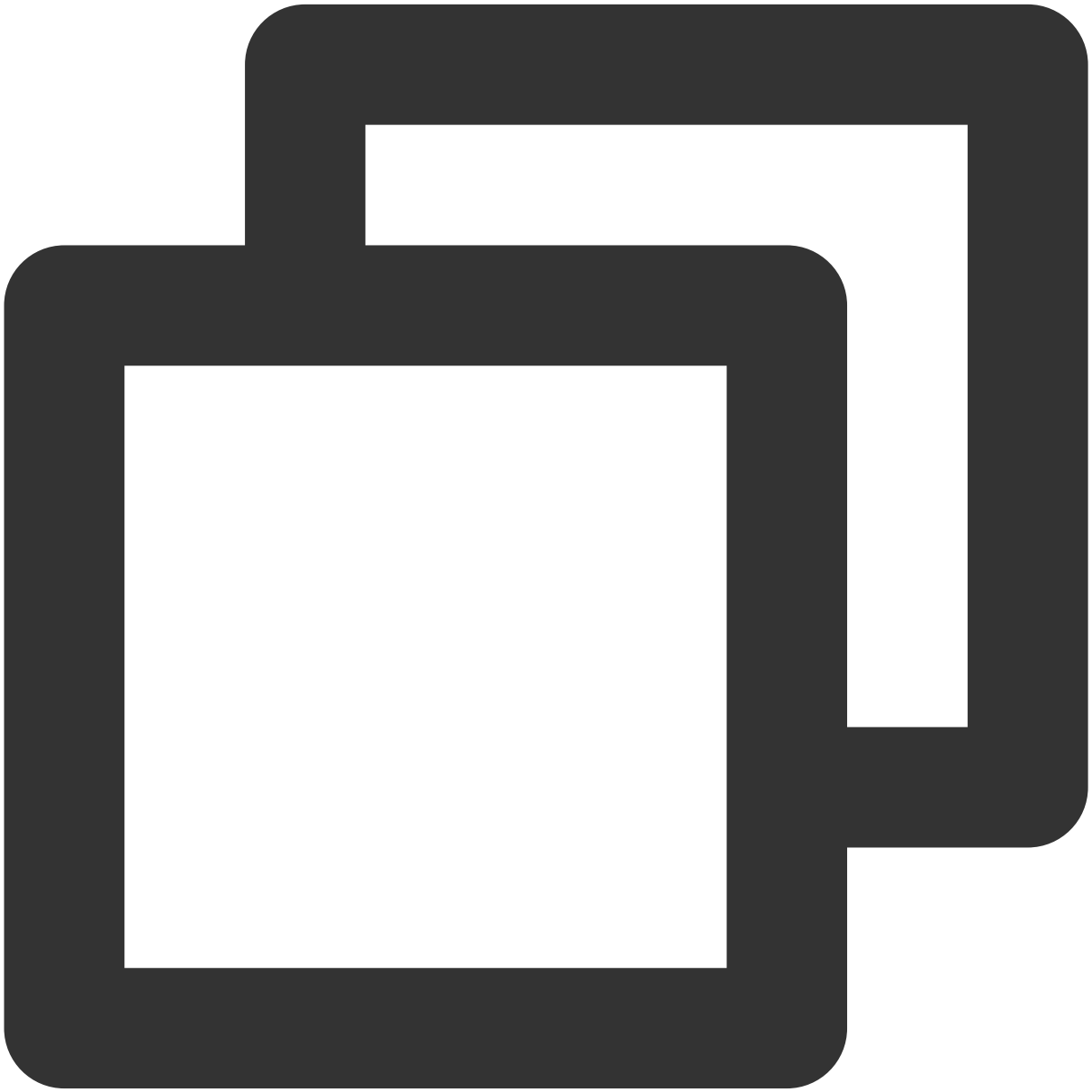
```
//1.创建工作流请求对象
MediaWorkflowListRequest request = new MediaWorkflowListRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setWorkflowId("w4e6963a18e2446ed8bc8f09410e******");
MediaWorkflowExecutionsResponse response = client.describeWorkflowExecutions(reques
List<MediaWorkflowExecutionObject> workflowExecutionList = response.getWorkflowExec
```

手动触发工作流

功能说明

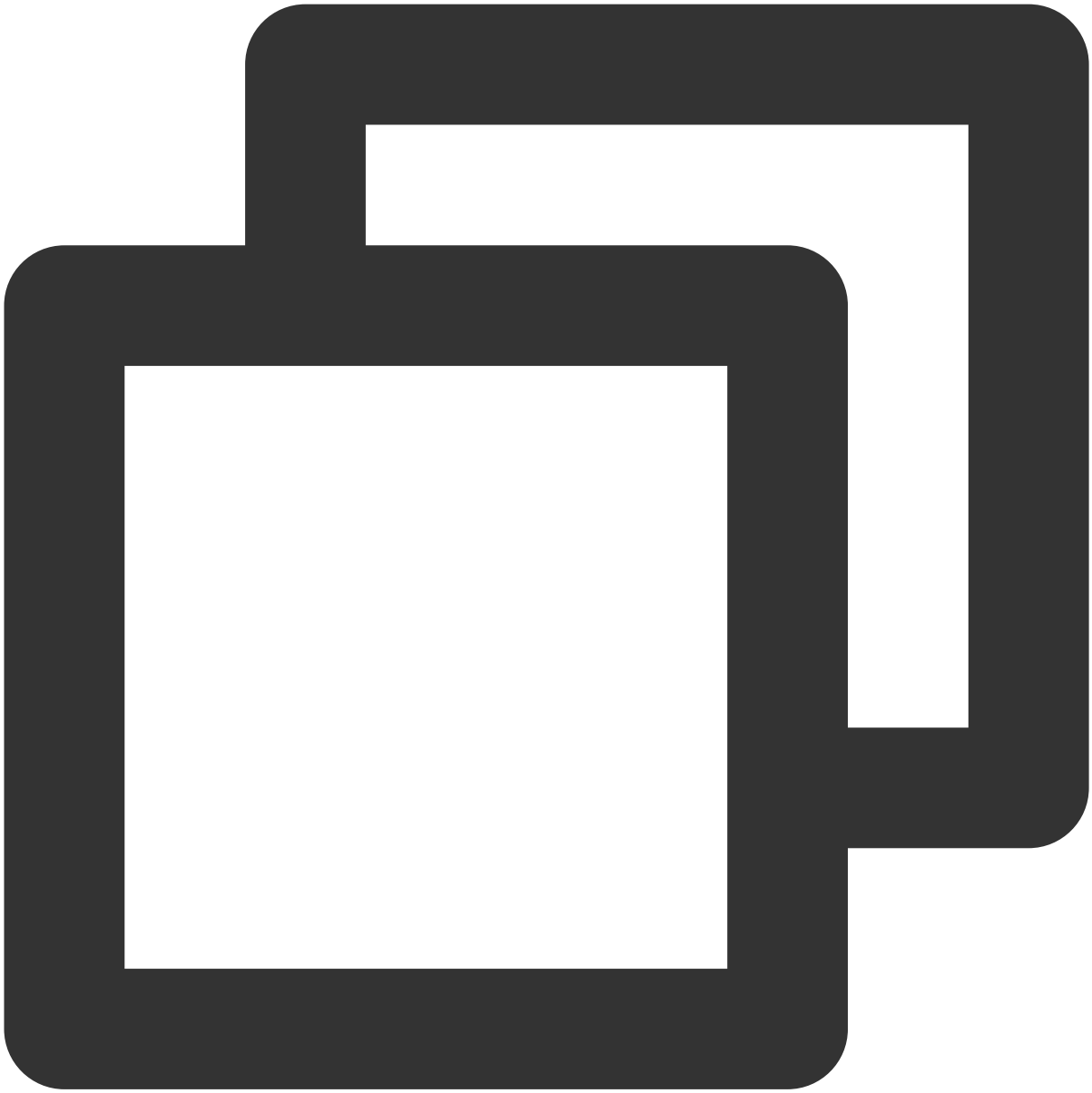
用于手动触发工作流。

方法原型



```
public MediaWorkflowListResponse triggerWorkflowList(MediaWorkflowListRequest reque
```

请求示例



```
//1.创建工作流请求对象
MediaWorkflowListRequest request = new MediaWorkflowListRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("DemoBucket-123456789");
request.setWorkflowId("we32f75950afe4a4682463d8158d*****");
request.setObject("1.mp4");
MediaWorkflowListResponse response = client.triggerWorkflowList(request);
```

参数说明

节点名称	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
object	需要进行工作流处理的对象名称	String	是
workflowId	需要触发的工作流 ID	String	是
name	存量触发任务名称，支持中文、英文、数字、—和_，长度限制128字符，默认为空	String	否

返回结果说明

成功：返回 MediaWorkflowListResponse 实例。

失败：发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

视频截帧

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于数据万象媒体截图的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GenerateSnapshot	获取媒体文件某个时间的截图	获取媒体文件某个时间的截图

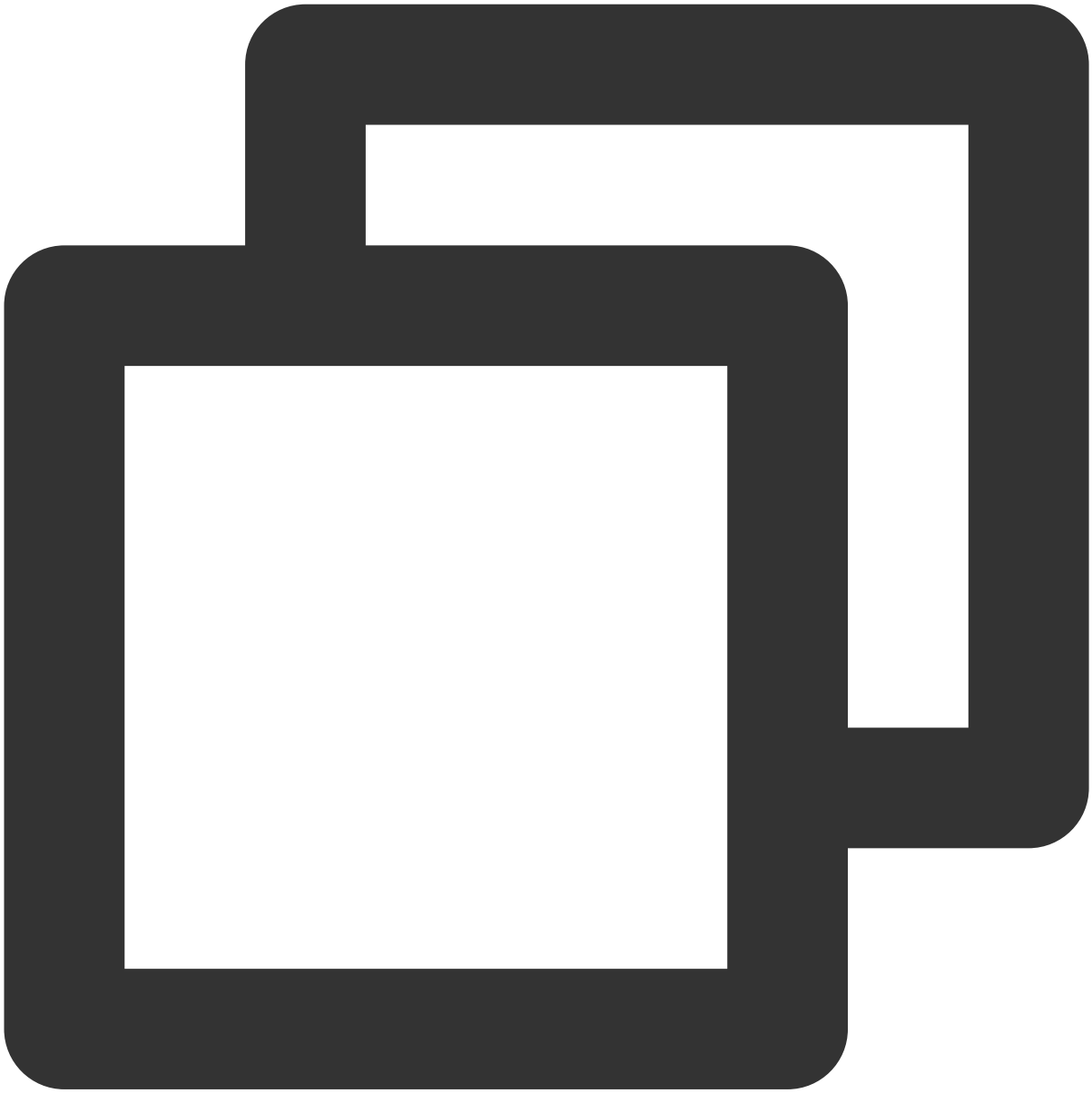
基本操作

获取媒体文件某个时间的截图

功能说明

获取媒体文件某个时间的截图。

方法原型



```
public SnapshotResponse generateSnapshot(SnapshotRequest request);
```

参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是

Input	媒体文件的位置信息	Container	是
Time	截取哪个时间点的内容，单位为秒	Float	是
Output	截图保存的位置信息	Container	是
Width	截图的宽。默认为0	Int	否
Height	截图的高。默认为0。 Width 和 Height 都为0时，表示使用视频的宽高。 如果单个为0，则以另外一个值按视频宽高比例自动适应	Int	否
Format	截图的格式，支持 jpg 和 png，默认 jpg	String	否
Mode	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 默认值为 exactframe	String	否
Rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转 off：不旋转 默认值为 auto	String	否

Input 的内容：

参数名称	父节点	描述	类型	是否必选
Object	Request.Input	文件的名称	String	是

Output 的内容：

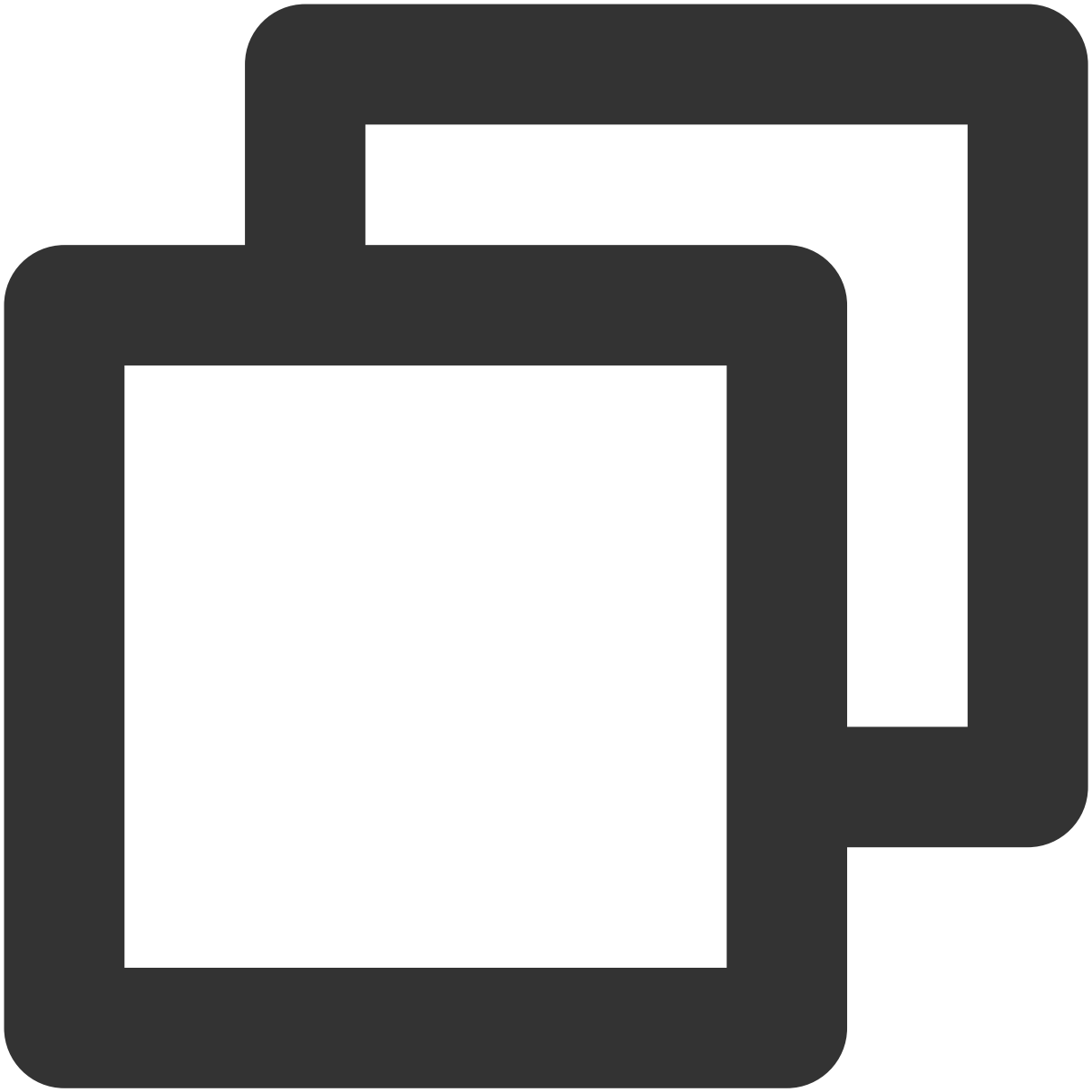
参数名称	父节点	描述	类型	是否必选
Region	Request.Output	存储桶所在的地域	String	是
Bucket	Request.Output	文件所属的 COS 存储桶	String	是
Object	Request.Output	文件的名称	String	是

返回结果说明

成功：返回截图实体信息。

失败：发生错误（如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
//1.创建截图请求对象
SnapshotRequest request = new SnapshotRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.getInput().setObject("1.mp4");
request.getOutput().setBucket("examplebucket-1250000000");
request.getOutput().setRegion("ap-chongqing");
request.getOutput().setObject("test/1.jpg");
request.setTime("15");
//3.调用接口,获取截图响应对象
SnapshotResponse response = client.generateSnapshot(request);
```

版权所有：腾讯云计算（北京）有限责任公司

文档预览
生成同步预览链接

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象文档处理任务的相关 API 概览以及 SDK 示例代码。

API	操作描述
同步预览转图片	生成同步预览转图片链接
同步预览转 HTML	生成同步预览转 HTML 链接

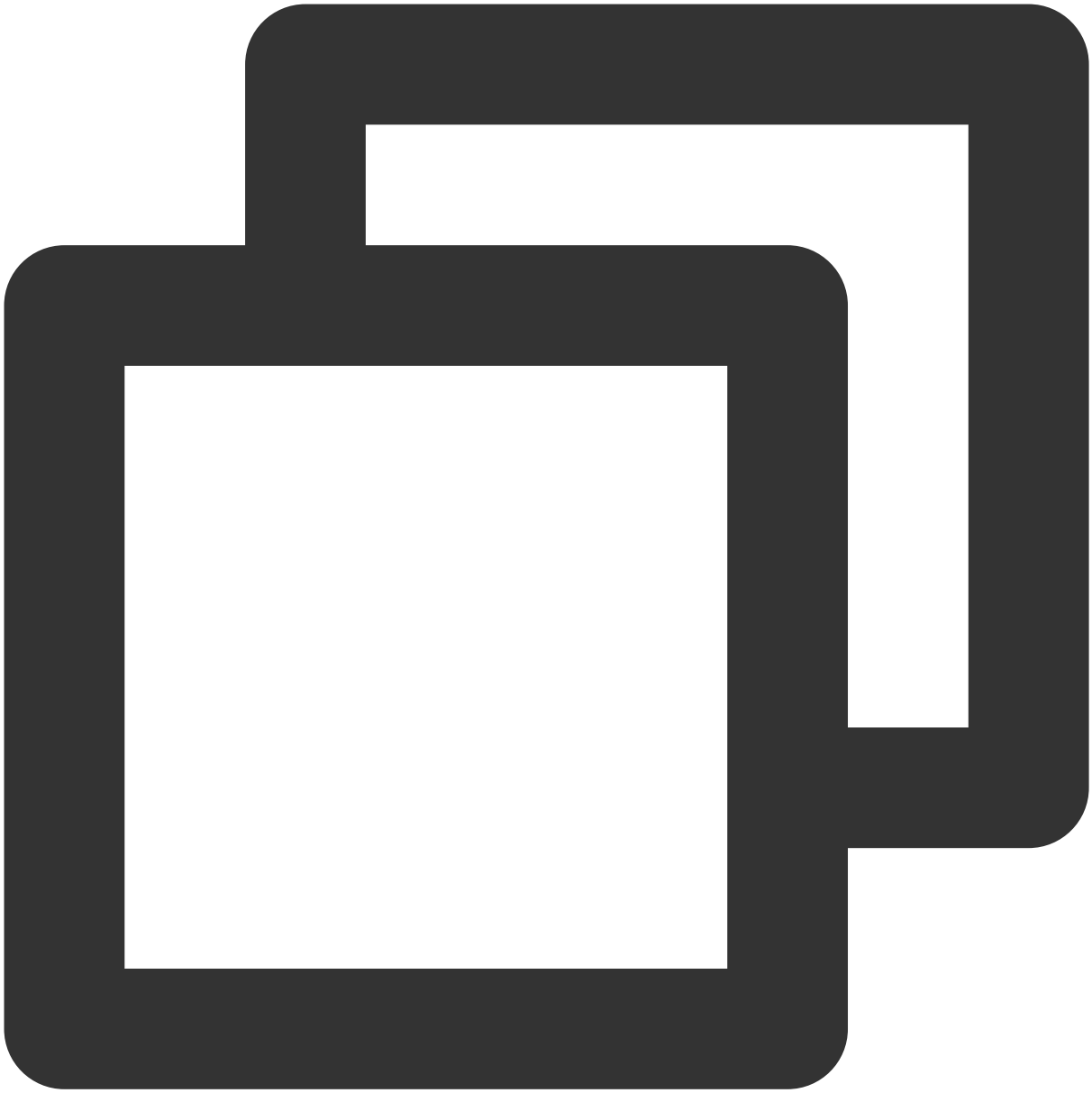
基本操作

同步预览转图片/HTML

功能说明

生成同步预览转图片或 HTML 链接，通过同一接口的不同参数实现。

方法原型



```
String GenerateDocPreviewUrl(DocHtmlRequest docJobRequest);
```

参数说明

DocHtmlRequest 中的具体数据描述如下：

节点名称（关键字）	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是

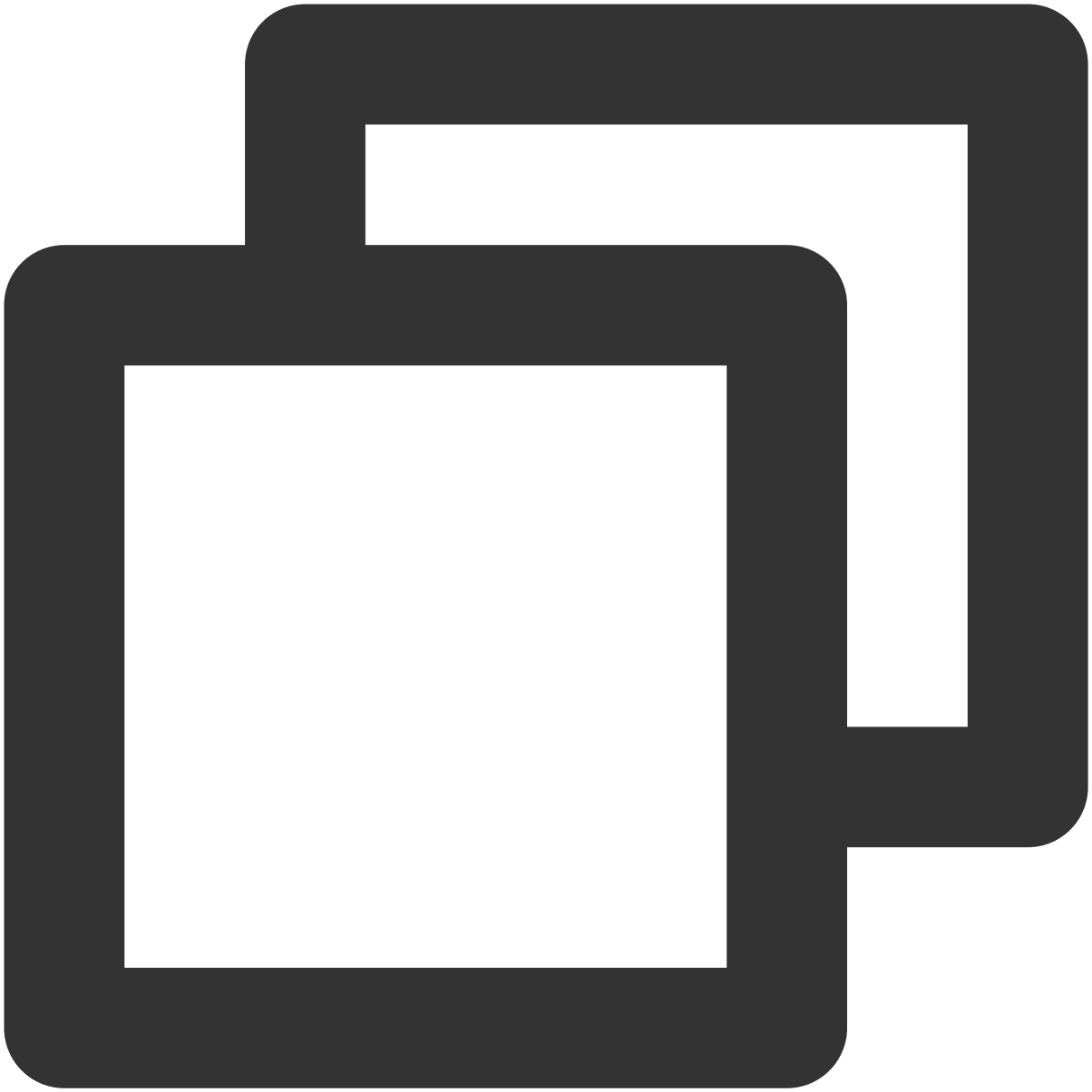
objectKey	COS 存储桶中的图片文件名称，COS 存储桶由Host指定，例如在北京的 examplebucket-1250000000存储桶中的目录 test 下的文件 img.jpg，则 Host 填写 examplebucket-1250000000.cos.ap-beijing.myqcloud.com，ObjectKey 填写 test/img.jpg	String	是
srcType	源数据的后缀类型，当前文档转换根据 COS 对象的后缀名来确定源数据类型。当 COS 对象没有后缀名时，可以设置该值	String	否
page	需转换的文档页码，默认从1开始计数；表格文件中 page 表示转换的第 X 个 sheet 的第 X 张图	Int	否
dstType	转换输出目标文件类型： png，转成 png 格式的图片文件 jpg，转成 jpg 格式的图片文件 pdf，转成 pdf 格式文件。 无法选择页码，page 参数不生效 html，转成 html 在线文档 如果传入的格式未能识别，默认使用 jpg 格式，SDK 默认值为 html	String	否
password	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String	否
comment	是否隐藏批注和应用修订，默认为0 0：隐藏批注，应用修订 1：显示批注和修订	Int	否
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String	否
quality	生成预览图的图片质量，取值范围为 [1, 100]，默认值100。 例如取值为100，代表生成图片质量为100%	Int	否
scale	预览图片的缩放参数，取值范围为 [10, 200]， 默认值100。 例如取值为200，代表图片缩放比例为200%即放大两倍	Int	否
imageDpi	按指定 dpi 渲染图片，该参数与 scale 共同作用，取值范围 96-600 ，默认值为 96 。转码后的图片单边宽度需小于65500像素	Int	否

返回结果说明

成功：返回预览链接。

失败：发生错误（如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
//1.创建请求对象
DocHtmlRequest request = new DocHtmlRequest();
//2.添加请求参数，参数详情请见 API 接口文档
request.setBucketName("markjrzhang-1251704708");
//如果需要转为图片 dstType 为 DocHtmlRequest.DocType.jpg
request.setDstType(DocHtmlRequest.DocType.html);
request.setObjectKey("1.pptx");
request.setPage("1");
//3.调用接口，获取任务响应对象
String previewUrl = client.GenerateDocPreviewUrl(request);
System.out.println(previewUrl);
```

版权所有：腾讯云计算（北京）有限责任公司

内容审核
图片审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核的 API 概览和 SDK 示例代码。

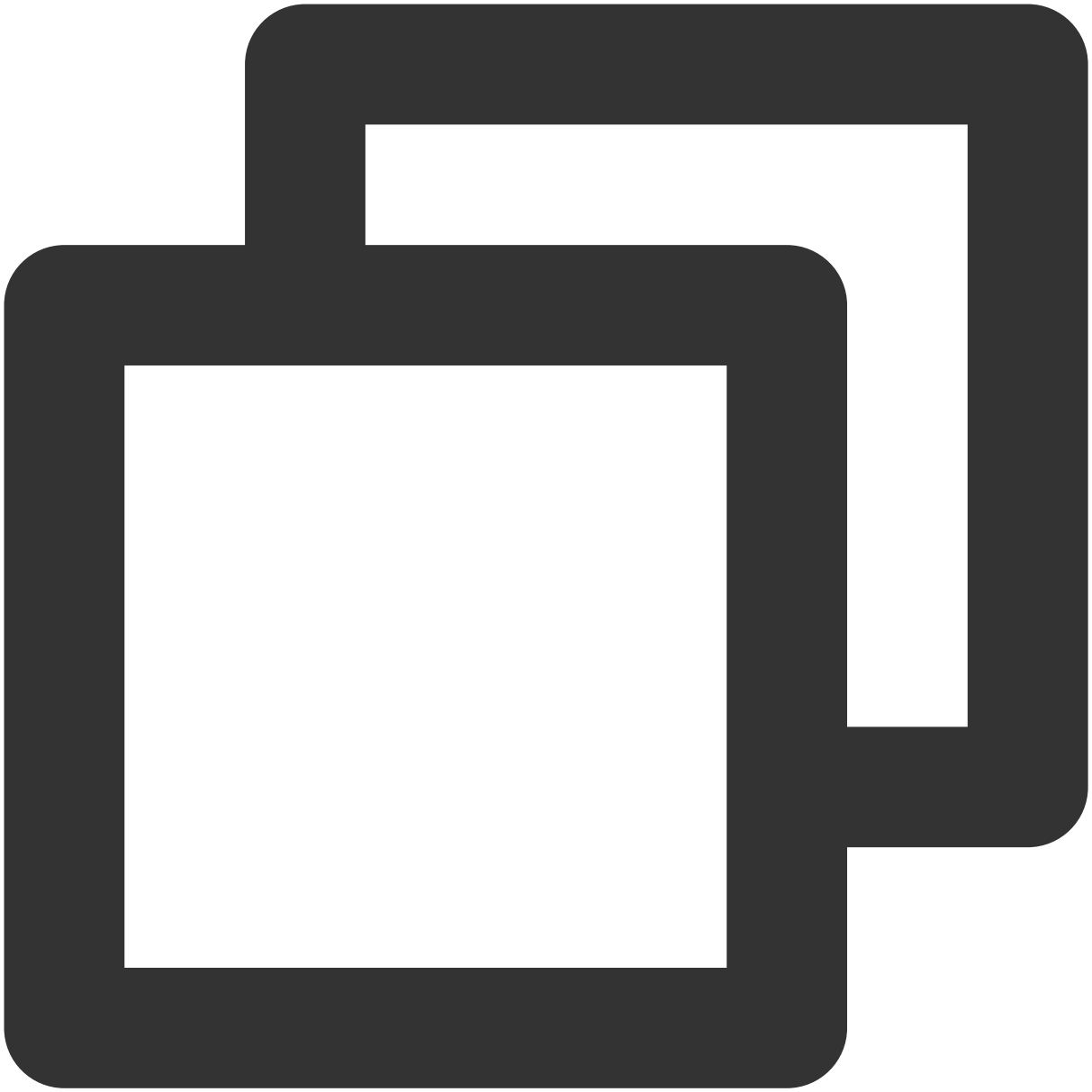
API	操作描述
图片同步审核	对对象存储（Cloud Object Storage，COS）存量数据进行涉黄、违法违规以及广告引导类图片的扫描
图片批量审核	对多个图片进行批量审核

图片同步审核

功能说明

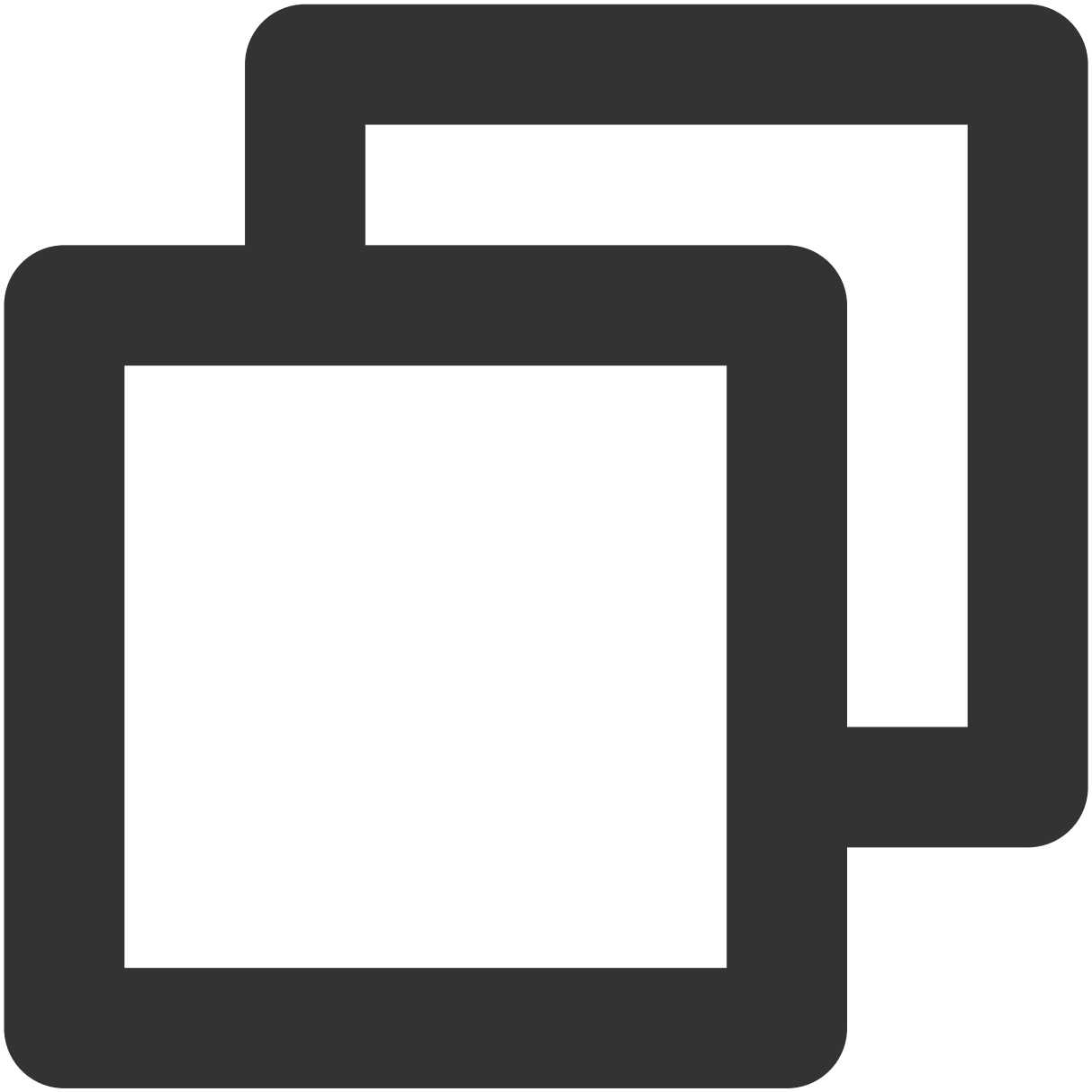
图片审核的存量扫描功能通过借助数据万象的持久化处理接口，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描。

方法原型



```
ImageAuditingResponse imageAuditing(ImageAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
ImageAuditingRequest request = new ImageAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
//2.1设置请求bucket
request.setBucketName("examplebucket-1250000000");
//2.2设置审核类型
request.setDetectType("porn,ads");
//2.3设置bucket中的图片位置
request.setObjectKey("1.png");
//3.调用接口,获取任务响应对象
ImageAuditingResponse response = client.imageAuditing(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	描述	类型	是否必选
ObjectKey	COS 存储桶中的图片文件名称，COS 存储桶由 `Host` 指定，例如在北京的 examplebucket-1250000000存储桶中的目录 test 下的文件 img.jpg，则 `Host` 填写 examplebucket-1250000000.cos.ap-beijing.myqcloud.com，`ObjectKey` 填写 test/img.jpg。	String	否
bizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置审核策略。您可以在控制台上获取到 bizType。bizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。bizType 不填写时，将自动使用默认的审核策略。	String	否
detectType	（后续不再维护）审核的场景类型，有效值：Porn（涉黄）、Ads（广告）等，可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 bizType 参数。	String	否
detectUrl	您可以通过填写 `detectUrl` 审核任意公网可访问的图片链接不填写 `detectUrl` 时，后台会默认审核 `ObjectKey` 填写了 `detectUrl` 时，后台会审核 `detectUrl` 链接，无需再填写 `ObjectKey`\`\`detectUrl` 示例：http://www.example.com/abc.jpg。	String	否
interval	审核 GIF 动图时，可使用该参数进行截帧配置，代表截帧的间隔。例如值设为5，则表示从第1帧开始截取，每隔5帧截取一帧，默认值5。	Int	否
maxFrames	针对 GIF 动图审核的最大截帧数量，需大于0。例如值设为5，则表示最大截取5帧，默认值为5。	Int	否
largeImageDetect	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。注：压缩最大支持32MB的图片，且会收取图片压缩费用。对于 GIF 等动态图过大时，压缩时间较长，可能会导致审核超时失败。	Int	否
dataId	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否

版权所有：腾讯云计算（北京）有限责任公司

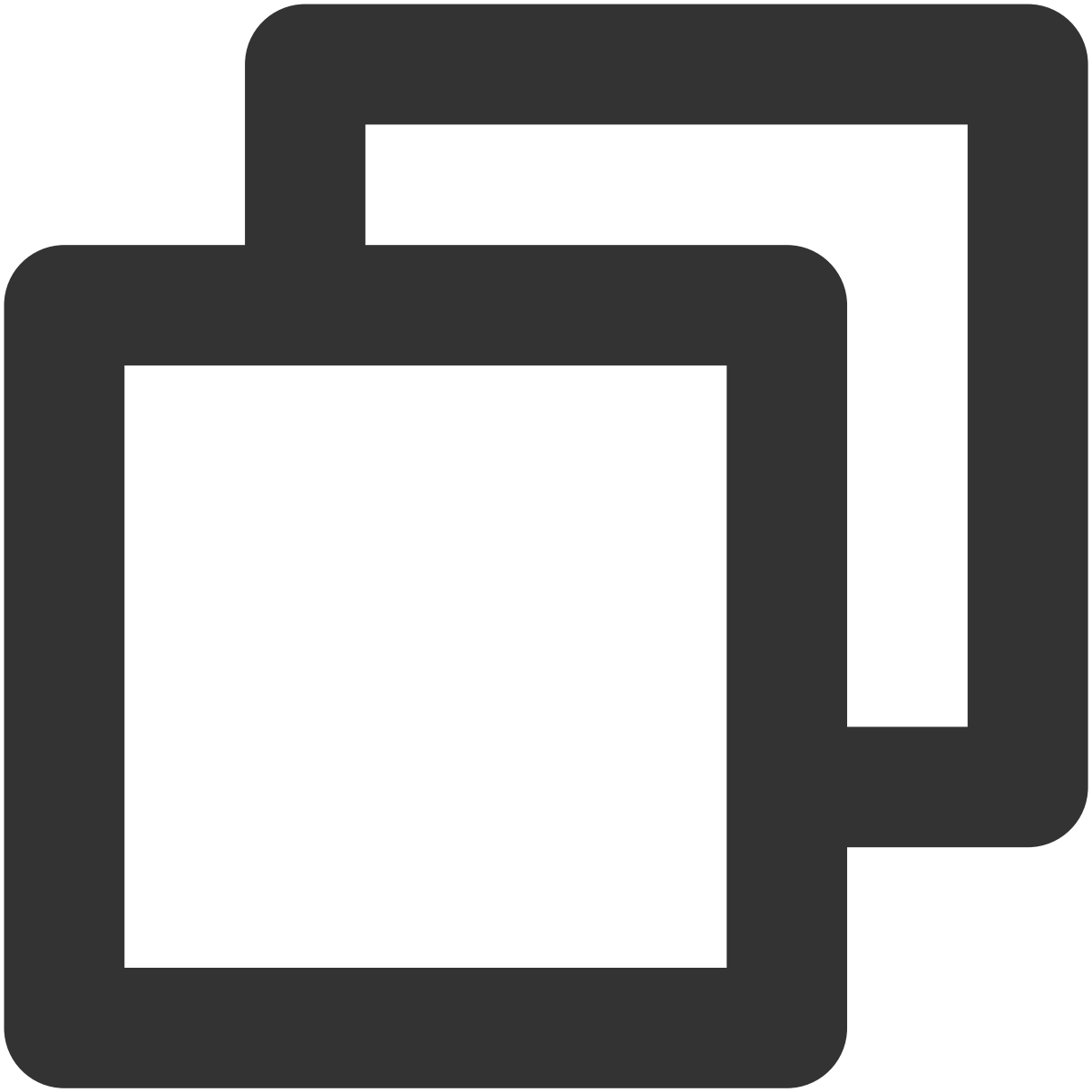
async	是否异步进行审核，取值 0：同步返回结果，1：异步进行审核，默认为0。	Int	否
callback	审核结果（Detail版本）以回调形式发送至您的回调地址，异步审核时生效，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否

返回结果说明

成功：成功则返回 ImageAuditingResponse 实例，内含审核结果内容。

失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

ImageAuditingResponse对象转为json后内容展示




```
{
    "requestId":"NjJkOTI3YTJfNGNhY2ZhMDlfOWJhOV9hZ*****",
    "jobId":"si461d9f9608de11ed9b3d525400b*****",
    "object":"1.png",
    "compressionResult":"0",
    "result":"0",
    "label":"Normal",
    "category":"",
    "subLabel":"",
    "score":"42",
    "text":"",
```

```
    "state":"Success",
    "pornInfo":{
        "code":"0",
        "msg":"OK",
        "hitFlag":"0",
        "score":"0",
        "label":"",
        "ocrResults":{
            "text":"",
            "keywords":""
        },
        "category":""
    }
}
```

响应包体具体数据内容如下：

参数名称	类型	描述
DataId	String	图片标识，审核结果会返回原始内容，长度限制为512字节。
JobId	String	图片审核任务的 ID。
State	String	审核任务的状态，值为 Success（审核成功）。
Object	String	存储在 COS 桶中的图片名称，创建任务使用 ObjectKey 时返回。
Url	String	图片文件的链接地址，创建任务使用 detect-url 时返回。
CompressionResult	Int	图片是否被压缩处理，值为 0（未压缩），1（正常压缩）。
Result	Int	该字段表示本次判定的审核结果，您可以根据该结果，进行后续的操作；建议您按照业务所需，对不同的审核结果进行相应处理。有效值：**0**（审核正常），**1**（判定为违规敏感文件），**2**（疑似敏感，建议人工复核）。
Label	String	该字段用于返回检测结果中所对应的**优先级最高的恶意标签**，表示模型推荐的审核结果，建议您按照业务所需，对不同违规类型与建议值进行处理。返回值：**Normal** 表示正常，**Porn** 表示色情，**Ads** 表示广告，以及其他不安全或不适宜的类型。
Category	String	该字段为 `Label` 的子集，表示审核命中的具体审核类别。例如 Sexy，表示色情标签中的性感类别。
SubLabel	String	该图命中的二级标签结果。
Score	Int	该字段表示审核结果命中审核信息的置信度，取值范围：0（**置信度最低**）-100（**置信度最高**），越高代表该内容越有可能属于当前返回审核

		信息例如：色情 99，则表明该内容非常有可能属于色情内容。
Text	String	该图里的文字内容（OCR），当审核策略开启文本内容检测时返回。
PornInfo	Container	审核场景为**涉黄**的审核结果信息。
AdsInfo	Container	审核场景为**广告引导**的审核结果信息。

审核信息（PornInfo、AdsInfo）中包含如下内容：

参数名称	类型	描述
Code	Int	错误码，0为正确，其他数字对应相应错误。详情请参见 错误码。
Msg	String	具体错误信息，如正常则为 OK。
HitFlag	Int	用于返回该审核场景的审核结果，返回值：0：正常。1：确认为当前场景的违规内容。2：疑似为当前场景的违规内容。
Score	Int	该字段表示审核结果命中审核信息的置信度，取值范围：0（**置信度最低**）-100（**置信度最高**），越高代表该内容越有可能属于当前返回审核信息。其中0 - 60分表示图片正常，61 - 90分表示图片疑似敏感，91 - 100分表示图片确定敏感。例如：色情 99，表明该内容非常有可能属于色情内容。
Label	String	该字段表示该截图的综合结果标签（可能为 SubLabel，可能为人物名字等）。
Category	String	该字段为 `Label` 的子集，表示审核命中的具体审核类别。例如 Sexy，表示色情标签中的性感类别。
SubLabel	String	该字段表示审核命中的具体子标签，例如：Porn 下的 SexBehavior 子标签。 注意：该字段可能返回空，表示未命中具体的子标签。
OcrResults	Container Array	该字段表示 OCR 文本识别的详细检测结果，包括文本识别结果、命中的关键词等信息，有相关违规内容时返回。
LibResults	Container Array	该字段用于返回基于风险库识别的结果。 注意：未命中风险库中样本时，此字段不返回。

Container 节点 LibResults 的内容：

参数名称	类型	描述
ImageId	String	该字段表示命中的风险库中的图片样本 ID。
Score	String	该字段用于返回当前标签下的置信度，取值范围：0（置信度最低）-100（置信度最高），越高代表当前的图片越有可能命中库中的样本。例如：色情 99，表明该数据非常有可能命中库中的色情样本。

版权所有：腾讯云计算（北京）有限责任公司

Container 节点 OcrResults 的内容：

参数名称	类型	描述
Text	String	图片 OCR 文本识别出的具体文本内容。
Keywords	Array	在当前审核场景下命中的关键词。
Location	Container	该参数用于返回 OCR 检测框在图片中的位置（左上角 xy 坐标、长宽、旋转角度），以方便快速定位识别文字的相关信息。

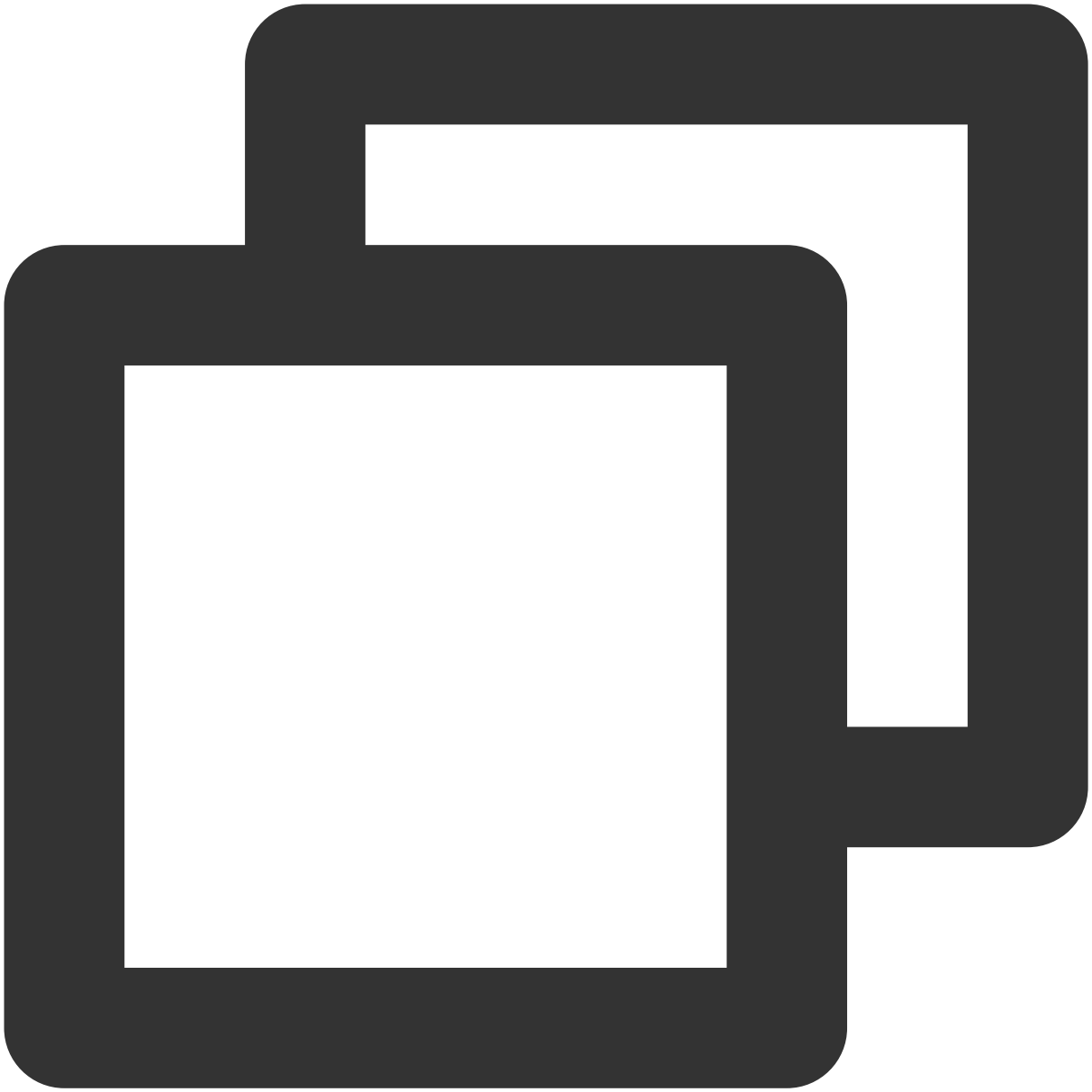
名称	类型	描述
X	Float	该参数用于返回检测框**左上角位置的横坐标**（x）所在的像素位置，结合剩余参数可唯一确定检测框的大小和位置。
Y	Float	该参数用于返回检测框**左上角位置的纵坐标**（y）所在的像素位置，结合剩余参数可唯一确定检测框的大小和位置。
Width	Float	该参数用于返回**检测框的宽度**（由左上角出发在 x 轴向右延伸的长度），结合剩余参数可唯一确定检测框的大小和位置。
Height	Float	该参数用于返回**检测框的高度**（由左上角出发在 y 轴向下延伸的长度），结合剩余参数可唯一确定检测框的大小和位置。
Rotate	Float	该参数用于返回**检测框的旋转角度**，该参数结合 X 和 Y 两个坐标参数可唯一确定检测框的具体位置；取值：**0-360**（**角度制**），方向为**逆时针旋转**。

图片批量审核

功能说明

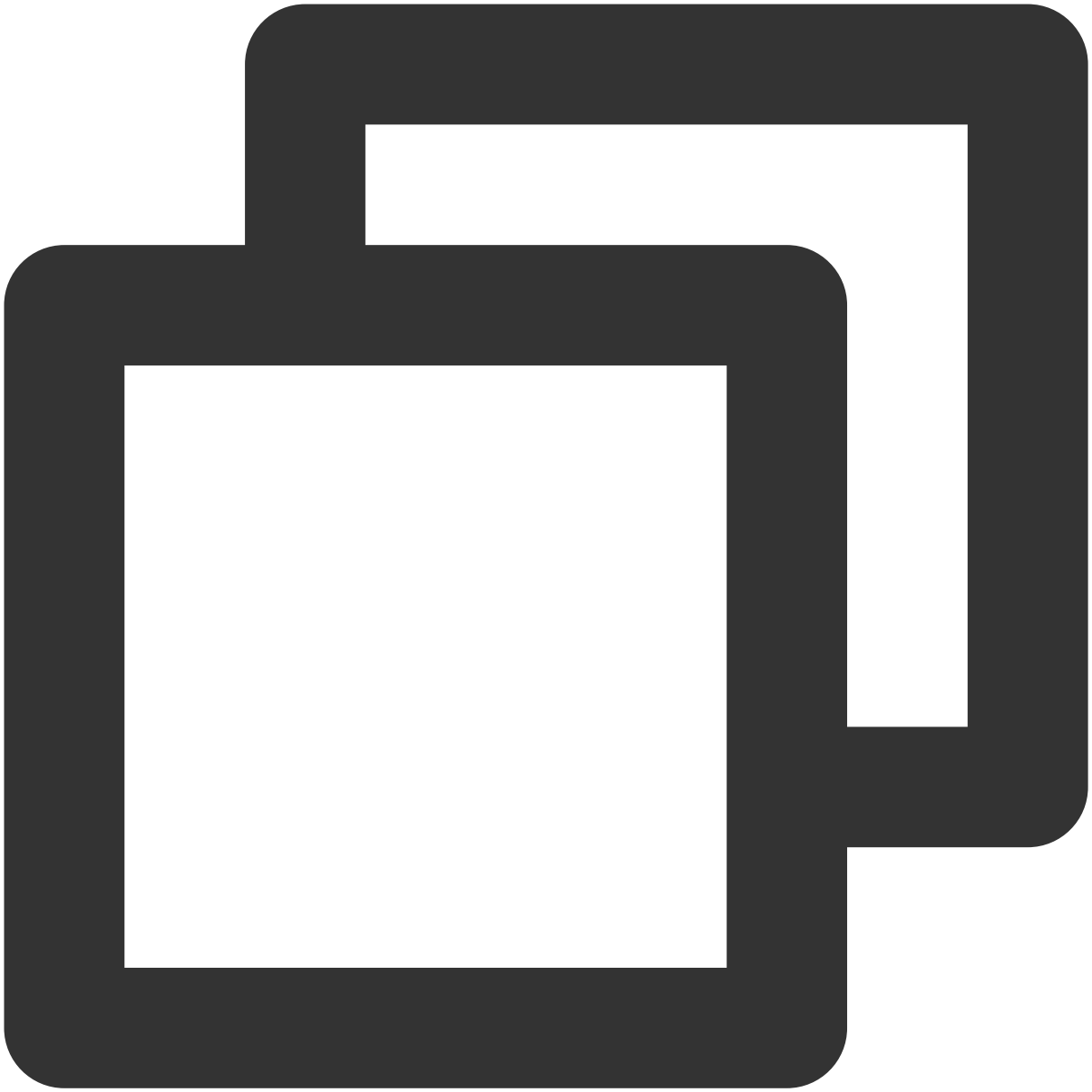
图片批量审核接口为同步请求方式，您可以通过本接口对多个图片文件进行内容审核。该接口属于 POST 请求。

方法原型



```
BatchImageAuditingResponse batchImageAuditing(BatchImageAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
BatchImageAuditingRequest request = new BatchImageAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
//2.1设置请求bucket
request.setBucketName("examplebucket-1250000000");
//2.2添加请求内容
List<BatchImageAuditingInputObject> inputList = request.getInputList();
BatchImageAuditingInputObject input = new BatchImageAuditingInputObject();
input.setObject("1.jpg");
input.setDataId("DataId");
inputList.add(input);
```

```
input = new BatchImageAuditingInputObject();
input.setUrl("https://examplebucket-1250000000.cos.ap-chongqing.myqcloud.com/1.png"
input.setDataId("DataId");
inputList.add(input);

//2.2设置审核类型
request.getConf().setDetectType("all");
//3.调用接口,获取任务响应对象
BatchImageAuditingResponse response = client.batchImageAuditing(request);
List<BatchImageJobDetail> jobList = response.getJobList();
```

参数说明

request 的数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	图片批量审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容，如有多个图片，请传入多个 Input 结构。	Container Array	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下，使用其中一种：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	存储在 COS 存储桶中的图片文件名称，例如在目录 test 中的文件 image.jpg，则文件名称为 test/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Url	Request.Input	图片文件的链接地址，例如 http://a-1250000.cos.ap-shanghai.myqcloud.com/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Interval	Request.Input	截帧频率，GIF 图检测专用，默认值为5，表示从第一帧（包含）开始每隔5帧截取一帧。	Int	否

版权所有：腾讯云计算（北京）有限责任公司

MaxFrames	Request.Input	最大截帧数量，GIF 图检测专用，默认值为5，表示只截取 GIF 的5帧图片进行审核，必须大于0。	Int	否
DataId	Request.Input	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否
LargeImageDetect	Request.Input	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。 注：压缩最大支持32M的图片，且会收取压缩费用。	Int	否
UserInfo	Request.Input	用户业务字段。	Container	否

Container 节点 UserInfo 的内容：

节点名称（关键字）	描述	类型	是否必选
TokenId	一般用于表示账号信息，长度不超过128字节。	String	否
Nickname	一般用于表示昵称信息，长度不超过128字节。	String	否
DeviceId	一般用于表示设备信息，长度不超过128字节。	String	否
AppId	一般用于表示 App 的唯一标识，长度不超过128字节。	String	否
Room	一般用于表示房间号信息，长度不超过128字节。	String	否
IP	一般用于表示 IP 地址信息，长度不超过128字节。	String	否
Type	一般用于表示业务类型，长度不超过128字节。	String	否
ReceiveTokenId	一般用于表示接收消息的用户账号，长度不超过128字节。	String	否
Gender	一般用于表示性别信息，长度不超过128字节。	String	否
Level	一般用于表示等级信息，长度不超过128字节。	String	否
Role	一般用于表示角色信息，长度不超过128字节。	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
BizType	Request.Conf	表示审核策略的唯一标识，您可以通过控制台上的	String	否

版权所有：腾讯云计算（北京）有限责任公司

		审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。		
DetectType	Request.Conf	（后续不再维护）审核的场景类型，有效值：Porn（涉黄）、Ads（广告）等，可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	String	否
Async	Request.Conf	是否异步进行审核，0：同步返回结果，1：异步进行审核。默认值为 0。	Integer	否
Callback	Request.Conf	审核结果（Detail版本）以回调形式发送至您的回调地址，异步审核时生效，支持以 `http://` 或者 `https://` 开头的地址，例如：`http://www.callback.com` 。	String	否

注意：

通过 Object 进行审核为内网操作，不会产生额外的外网流量。

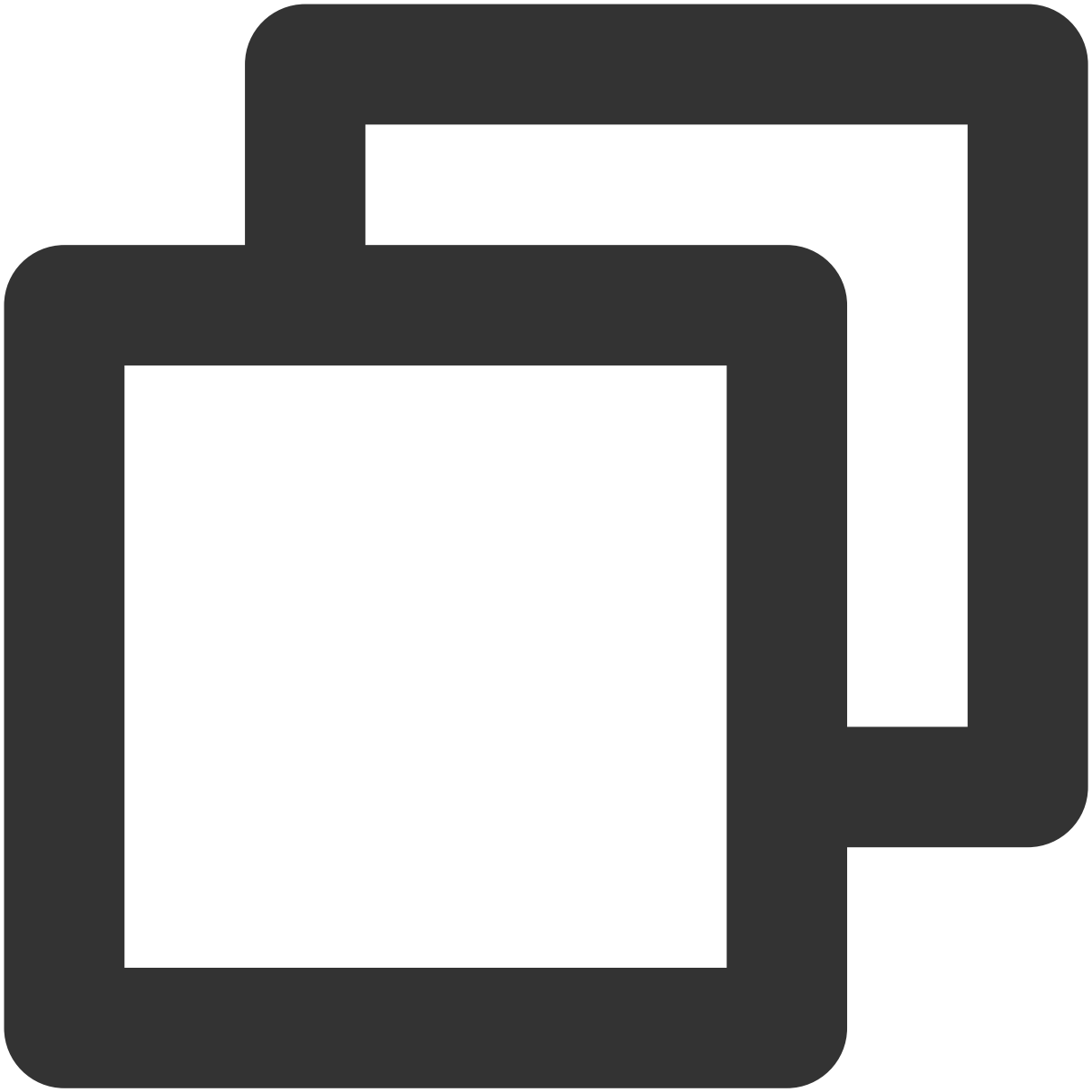
通过 Url 进行审核，会产生图片所在源站对应的外网流量。

返回结果说明

成功：成功则返回 BatchImageAuditingResponse 实例，内含审核结果内容。

失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

ImageAuditingResponse对象转为json后内容展示




```
{
  "requestId":"NjJkOTJjMjVfMTIwNjUzMDlfNDg5OF8*****",
  "jobList":[
    {
      "dataId":"DataId",
      "jobId":"sif6423b8008e011ed9b3d525400*****",
      "category":"",
      "label":"Normal",
      "result":"0",
      "object":"1.jpg",
      "score":"0",
```

```
        "subLabel":"",
        "text":"",
        "code":"",
        "message":"",
        "url":"",
        "compressionResult":"0",
        "pornInfo":{
          "code":"0",
          "msg":"OK",
          "hitFlag":"0",
          "score":"45",
          "label":"",
          "keywords":"",
          "count":"",
          "subLabel":"",
          "ocrResults":{
            "text":"",
            "keywords":""
          },
          "category":""
        },
        "adsInfo":{
          "code":"0",
          "msg":"OK",
          "hitFlag":"0",
          "score":"0",
          "label":"",
          "keywords":"",
          "count":"",
          "subLabel":"",
          "ocrResults":{
            "text":"",
            "keywords":""
          },
          "category":""
        },
        "userInfo":{
          "tokenId":"",
          "nickname":"",
          "deviceId":"",
          "appId":"",
          "room":"",
          "ip":"",
          "type":"",
          "receiveTokenId":"",
          "gender":"",
          "level":"",
```

```
        "role":""
      },
      "ocrResults":{
        "text":"",
        "keywords":""
      }
    },
    {
      "dataId":"DataId",
      "jobId":"sif640a3c108e011ed9b3d525400*****",
      "category":"",
      "label":"Normal",
      "result":"0",
      "object":"",
      "score":"42",
      "subLabel":"",
      "text":"",
      "code":"",
      "message":"",
      "url":"https://demo-1234567890.cos.ap-chongqing.myqcloud.com/1.png",
      "compressionResult":"0",
      "pornInfo":{
        "code":"0",
        "msg":"OK",
        "hitFlag":"0",
        "score":"0",
        "label":"",
        "keywords":"",
        "count":"",
        "subLabel":"",
        "category":"",
        "ocrResults":{
          "text":"",
          "keywords":""
        }
      },
      "adsInfo":{
        "code":"0",
        "msg":"OK",
        "hitFlag":"0",
        "score":"0",
        "label":"",
        "keywords":"",
        "count":"",
        "subLabel":"",
        "category":"",
        "ocrResults":{
```

```
            "text":"",
            "keywords":""
          }
        },
        "userInfo":{
          "tokenId":"",
          "nickname":"",
          "deviceId":"",
          "appId":"",
          "room":"",
          "ip":"",
          "type":"",
          "receiveTokenId":"",
          "gender":"",
          "level":"",
          "role":""
        },
        "ocrResults":{
          "text":"",
          "keywords":""
        }
      }
    ]
}
```

响应包体具体数据内容如下：

Response 的内容：

参数名称	类型	描述
JobsDetail	Array	图片审核的结果
RequestId	String	每次请求发送时，服务端将会自动为请求生成一个 ID，遇到问题时，该 ID 能更快地协助定位问题。

JobsDetail 节点内容如下：

参数名称	类型	描述
Code	String	错误码，只有失败时返回。详情请查看 错误码列表。
Message	String	错误描述，只有失败时返回。
DataId	String	图片标识，审核结果会返回原始内容，长度限制为512字节。
JobId	String	本次审核任务的 ID。
State	String	审核任务的状态，值为Success（审核成功）、Failed（审核失败）其中

		一个。
Object	String	存储在 COS 存储桶中的图片文件名称，创建任务使用Object时返回。
Url	String	图片文件的链接地址，创建任务使用Url时返回。
CompressionResult	Integer	图片是否被压缩处理，值为 0（未压缩），1（正常压缩）。
Label	String	该字段用于返回检测结果中所对应的**优先级最高的恶意标签**，表示模型推荐的审核结果，建议您按照业务所需，对不同违规类型与建议值进行处理。 返回值：**Normal**：正常，**Porn**：色情，**Ads**：广告等。
Result	Integer	该字段表示本次判定的审核结果，您可以根据该结果，进行后续的操作；建议您按照业务所需，对不同的审核结果进行相应处理。有效值：**0**（审核正常），**1**（判定为违规敏感文件），**2**（疑似敏感，建议人工复核）。
Score	Integer	该字段表示审核结果命中审核信息的置信度，取值范围：0（**置信度最低**）-100（**置信度最高**），越高代表该内容越有可能属于当前返回审核信息。例如：色情 99，表明该内容非常有可能属于色情内容。
Category	String	该图命中的审核类别结果。
SubLabel	String	该图命中的二级标签结果。
Text	String	该图里的文字内容（OCR），当审核策略开启文本内容检测时返回。
PornInfo	Container	审核场景为**涉黄**的审核结果信息。
AdsInfo	Container	审核场景为**广告引导**的审核结果信息。
UserInfo	Container	请求中设置的 UserInfo 原样返回。
ListInfo	Container	账号黑白名单结果。

审核信息（PornInfo、AdsInfo 等）中的内容如下：

参数名称	类型	描述
Code	Integer	单个审核场景的错误码，0为成功，其他为失败。详情请参见 错误码。
Msg	String	具体错误信息，如正常则为 OK。
HitFlag	Integer	用于返回该审核场景的审核结果，返回值：0：正常。1：确认为当前场景的违规内容。2：疑似为当前场景的违规内容。
Score	Integer	该字段表示审核结果命中审核信息的置信度，取值范围：0（**置信度最低**）-100（**置信度最高**），越高代表该内容越有可能属于当前返回审核信息例如：色情

		99，则表明该内容非常有可能属于色情内容。
Label	String	该图的结果标签（为综合标签，可能为 SubLabel，可能为人物名字等）。
Category	String	该字段为 `Label` 的子集，表示审核命中的具体审核类别。例如 Sexy，表示色情标签中的性感类别。
SubLabel	String	该图的二级标签结果。
OcrResults	Container Array	该字段表示 OCR 文本识别的详细检测结果，包括文本坐标信息、文本识别结果等信息，有相关违规内容时返回。
LibResults	Container Array	该字段用于返回基于风险库识别的结果。注意：未命中风险库中样本时，此字段不返回。

Container 节点 LibResults 的内容：

节点名称（关键字）	类型	描述
ImageId	String	该字段表示命中的风险库中的图片样本ID。
Score	Integer	该字段用于返回当前标签下的置信度，取值范围：0（置信度最低）-100（置信度最高），越高代表当前的图片越有可能命中库中的样本。例如：色情 99，则表明该数据非常有可能命中库中的色情样本。

OcrResults 的内容如下：

节点名称（关键字）	类型	描述
Text	String	有敏感信息的 Ocr 文本内容。
Keywords	String Array	该段内容中的敏感文字。

UserInfo 的内容（与请求中的 UserInfo 一致）：

节点名称（关键字）	描述	类型	是否必选
TokenId	一般用于表示账号信息，长度不超过128字节。	String	否
Nickname	一般用于表示昵称信息，长度不超过128字节。	String	否
DeviceId	一般用于表示设备信息，长度不超过128字节。	String	否
AppId	一般用于表示App的唯一标识，长度不超过128字节。	String	否
Room	一般用于表示房间号信息，长度不超过128字节。	String	否

版权所有：腾讯云计算（北京）有限责任公司

IP	一般用于表示IP地址信息，长度不超过128字节。	String	否
Type	一般用于表示业务类型，长度不超过128字节。	String	否
ReceiveTokenId	一般用于表示接收消息的用户账号，长度不超过128字节。	String	否
Gender	一般用于表示性别信息，长度不超过128字节。	String	否
Level	一般用于表示等级信息，长度不超过128字节。	String	否
Role	一般用于表示角色信息，长度不超过128字节。	String	否

ListInfo 的内容：

节点名称（关键字）	类型	描述
ListResults	Container Array	命中的所有名单结果。

ListResults 的内容：

节点名称（关键字）	类型	描述
ListType	String	命中的名单类型，取值为0（白名单）和1（黑名单）。
ListName	String	命中的名单名称。
Entity	String	命中了名单中的哪条内容。

版权所有：腾讯云计算（北京）有限责任公司

视频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核的 API 概览和 SDK 示例代码。

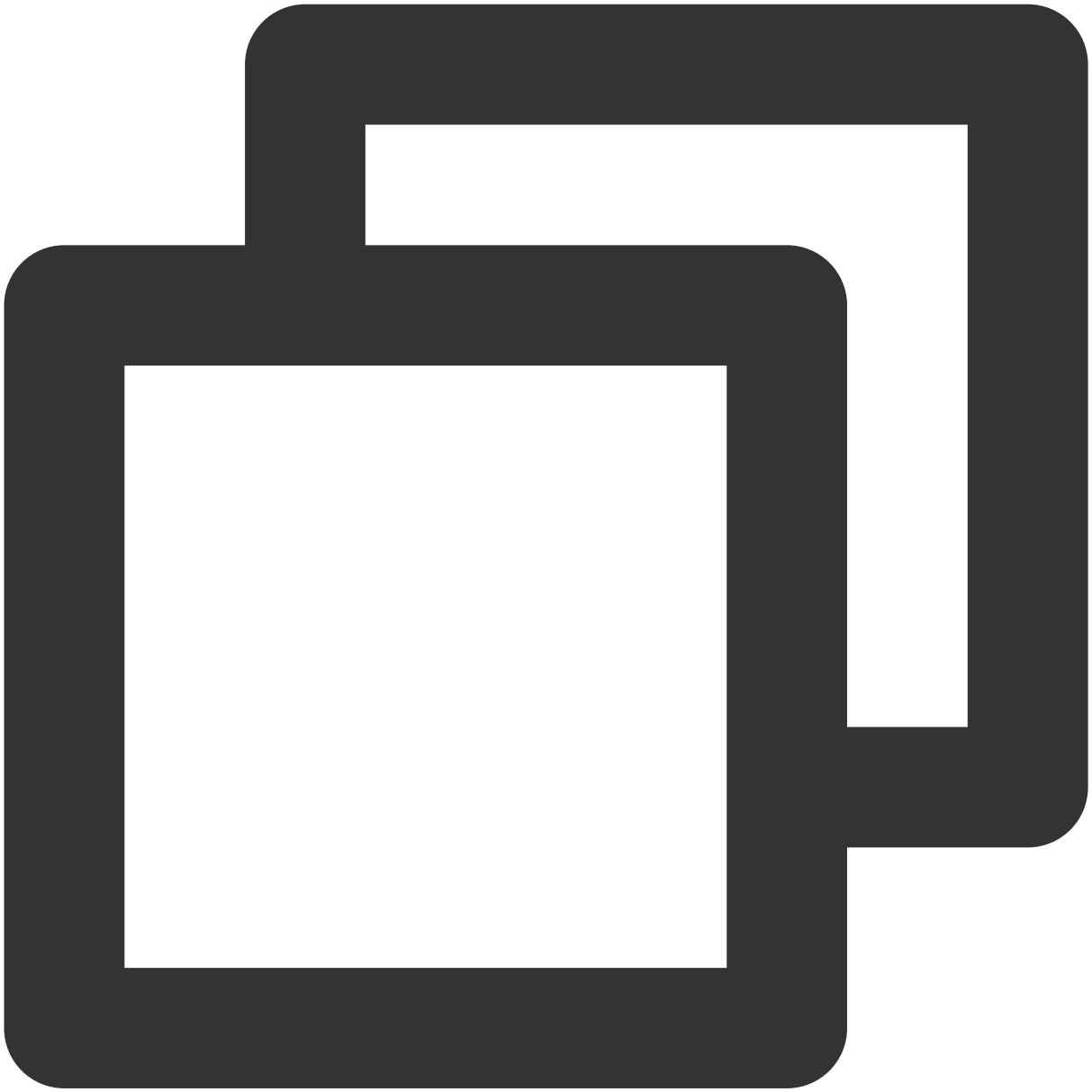
API	操作描述
提交视频审核任务	用于提交一个视频审核任务
查询视频审核任务结果	用于查询指定的视频审核任务

提交视频审核任务

功能说明

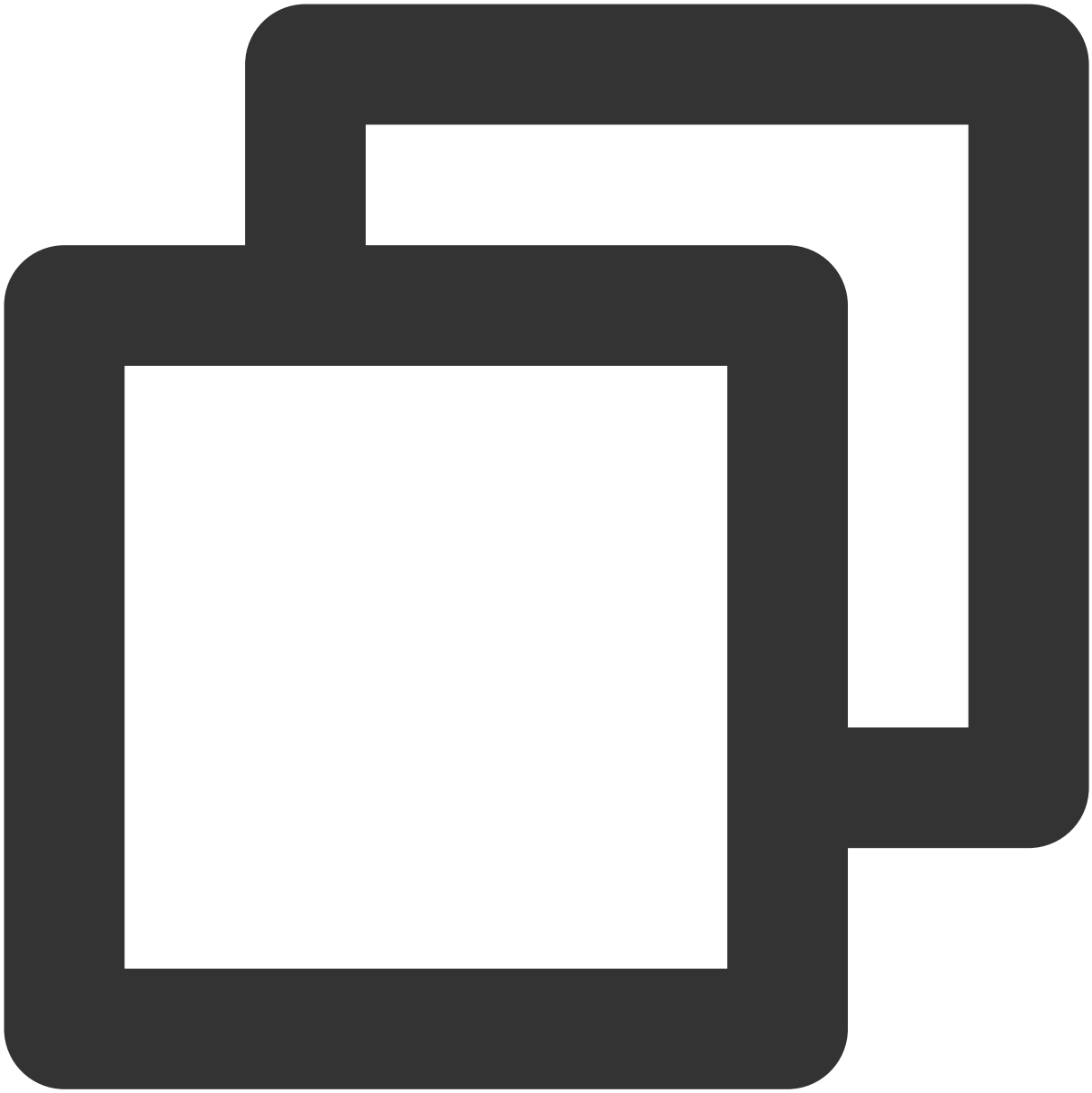
用于提交一个视频审核任务。

方法原型



```
VideoAuditingResponse createVideoAuditingJob(VideoAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
VideoAuditingRequest request = new VideoAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.getInput().setObject("1.mp4");
request.getConf().setDetectType("Porn,Ads");
request.getConf().getSnapshot().setCount("10");
request.getConf().getSnapshot().setMode("Interval");
request.getConf().getSnapshot().setTimeInterval("10");
//3.调用接口,获取任务响应对象
VideoAuditingResponse response = client.createVideoAuditingJob(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的视频。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	当前 COS 存储桶中的视频文件名称，例如在目录 test 中的文件 video.mp4，则文件名称为 test/video.mp4。	String	是
Url	Request.Input	视频文件的链接地址，例如 http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/test.mp4。Object 和 Url 只能选择其中一种。	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	是
Snapshot	Request.Conf	视频画面的审核通过视频截帧能力截取出一定量的截图，通过对截图逐一审核而实现的，该参数用于指定视频截帧的配置。	Container	是
Callback	Request.Conf	回调地址，以 `http://` 或者 `https://` 开头的地址。	String	否
CallbackVersion	Request.Conf	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String	否
BizType	Request.Conf	审核策略，不带审核策略时使用默认策略。	String	否
DetectContent	Request.Conf	用于指定是否审核视频声音，当值为0时：表	Integer	否

版权所有：腾讯云计算（北京）有限责任公司

		示只审核视频画面截图；值为1时：表示同时审核视频画面截图和视频声音。默认值为0。		

Container 类型 Snapshot 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Mode	Request.Conf.Snapshot	截帧模式。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。 Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片。 Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。 Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧。	String	否
Count	Request.Conf.Snapshot	视频截帧数量，范围为(0, 10000]。	String	否
TimeInterval	Request.Conf.Snapshot	视频截帧频率，范围为(0, 60]，单位为秒，支持 float 格式，执行精度精确到毫秒。	Float	否

返回结果说明

成功：成功则返回 VideoAuditingResponse 媒体审核任务结果对象。

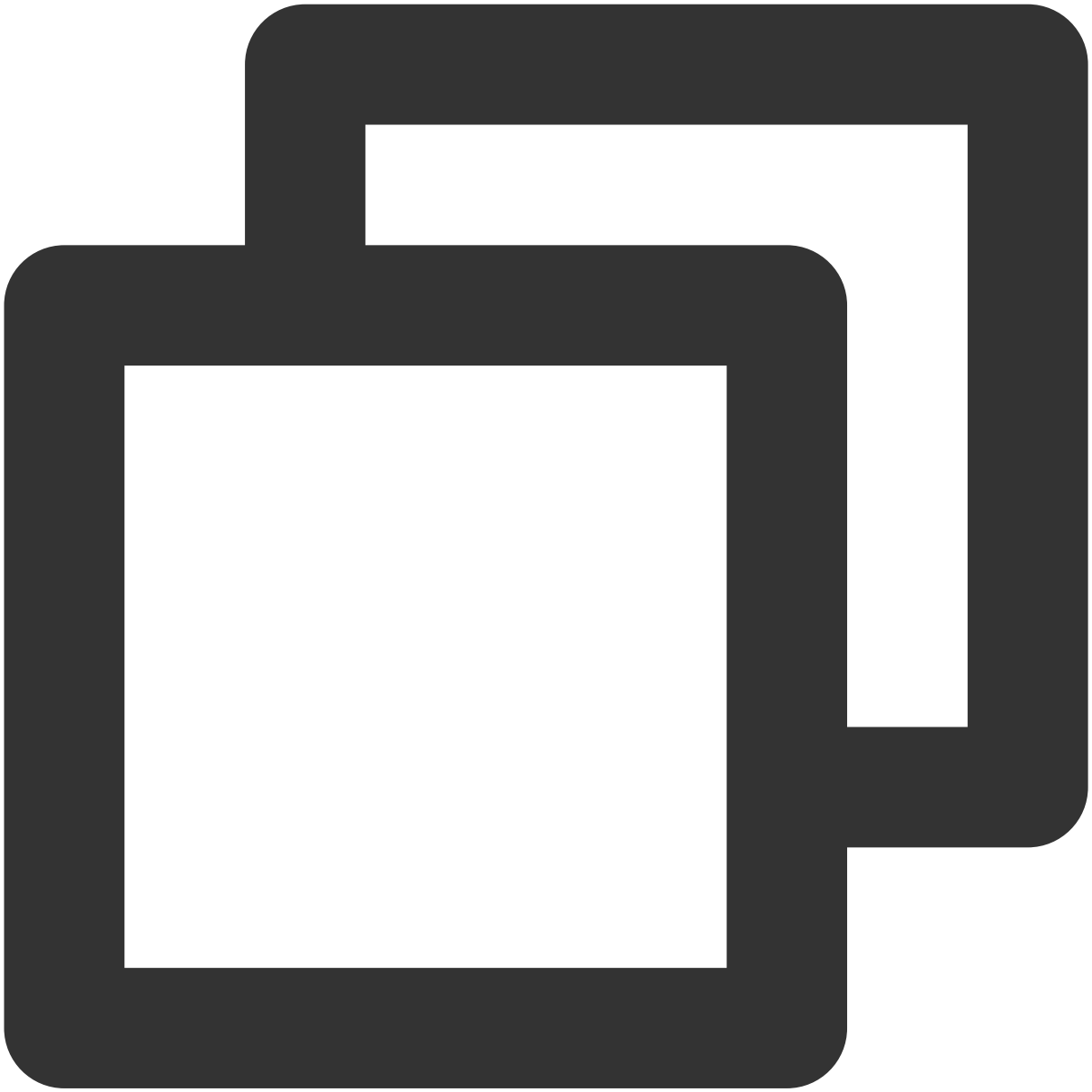
失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

查询视频审核任务结果

功能说明

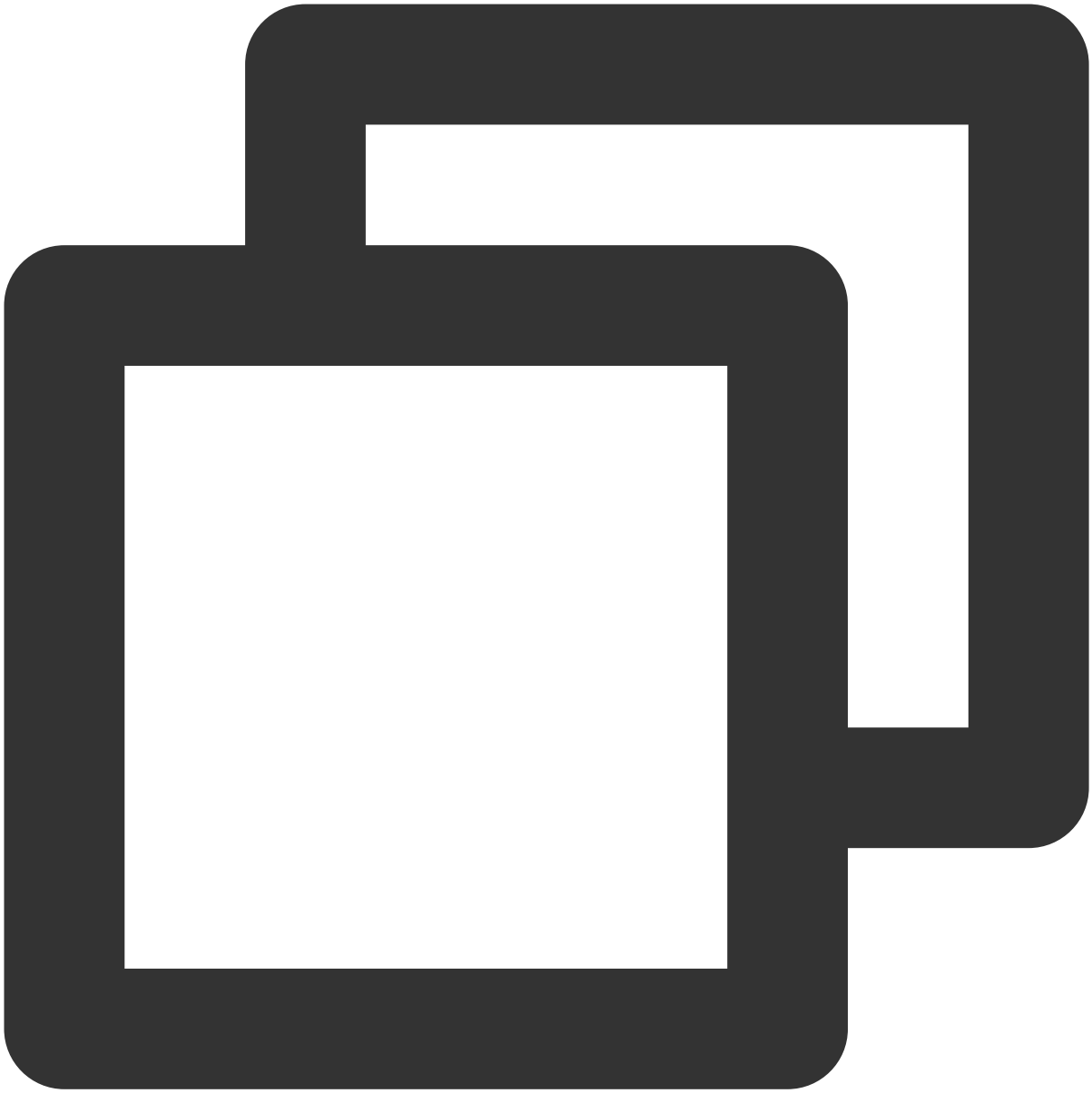
用于查询视频审核任务执行情况和结果。

方法原型



```
VideoAuditingResponse describeAuditingJob(VideoAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
VideoAuditingRequest request = new VideoAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("av81591b4bbd2211eb80235254008*****");
//3.调用接口,获取任务响应对象
VideoAuditingResponse response = client.describeAuditingJob(request);
```

参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
jobId	需要查询的任务 ID	String	是

返回结果说明

成功： 返回 VideoAuditingResponse 审核任务结果对象。

失败： 发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

音频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于音频审核的 API 概览和 SDK 示例代码。

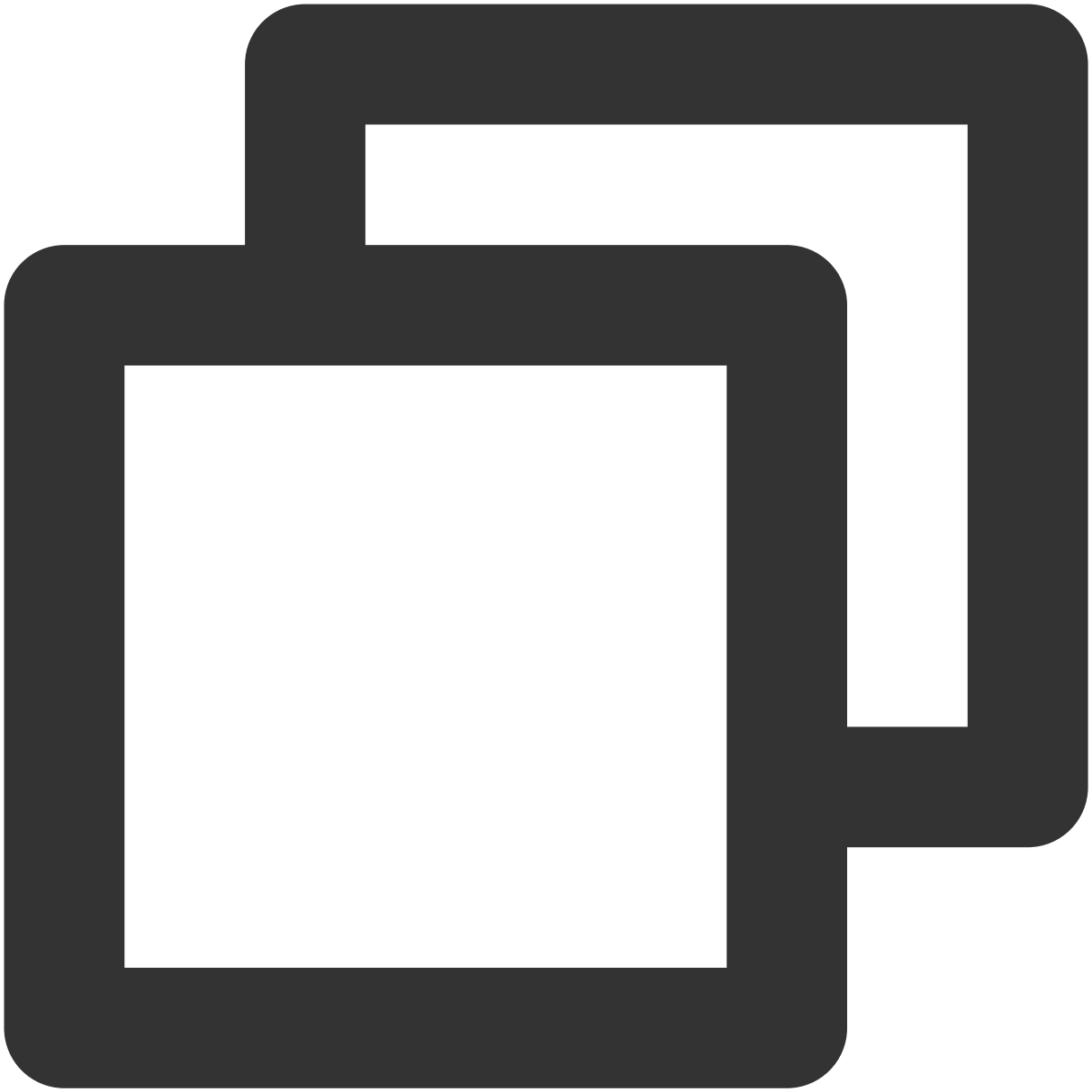
API	操作描述
提交音频审核任务	用于提交一个音频审核任务
查询音频审核任务结果	用于查询指定的音频审核任务

提交音频审核任务

功能说明

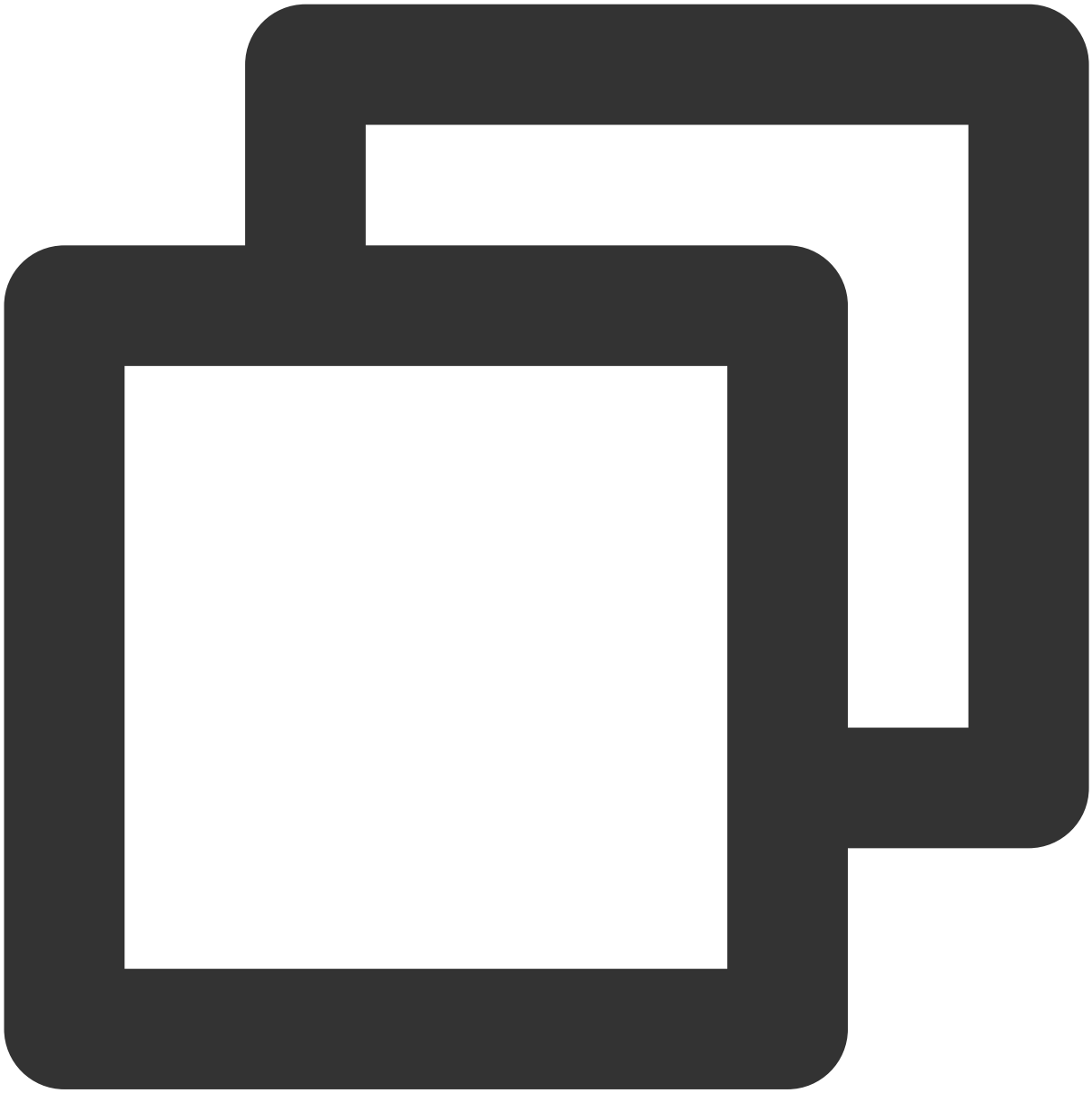
用于提交一个音频审核任务。

方法原型



```
AudioAuditingResponse createAudioAuditingJob(AudioAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
AudioAuditingRequest request = new AudioAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.getInput().setObject("pron.mp3");
request.getConf().setDetectType("Porn,Ads");
request.getConf().setCallback("http://cloud.tencent.com/");
//3.调用接口,获取任务响应对象
AudioAuditingResponse response = client.createAudioAuditingJobs(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	存储在 COS 存储桶中的音频文件名称，例如在目录 test 中的文件audio.mp3，则文件名称为 test/audio.mp3。Object 和 Url 只能选择其中一种。	String	否
Url	Request.Input	音频文件的链接地址，例如 http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/audio.mp3。Object 和 Url 只能选择其中一种。	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见设置公共审核策略。	String	否
DetectType	Request.Conf	（后续不再维护）审核的场景类型，有效值：Porn（涉黄）、Ads（广告）等，可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。如您有更多场景的审核需要，请使用 BizType 参数。	String	否
Callback	Request.Conf	审核结果以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如：`http://www.callback.com` 。	String	否
CallbackVersion	Request.Conf	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	string	否

返回结果说明

版权所有：腾讯云计算（北京）有限责任公司

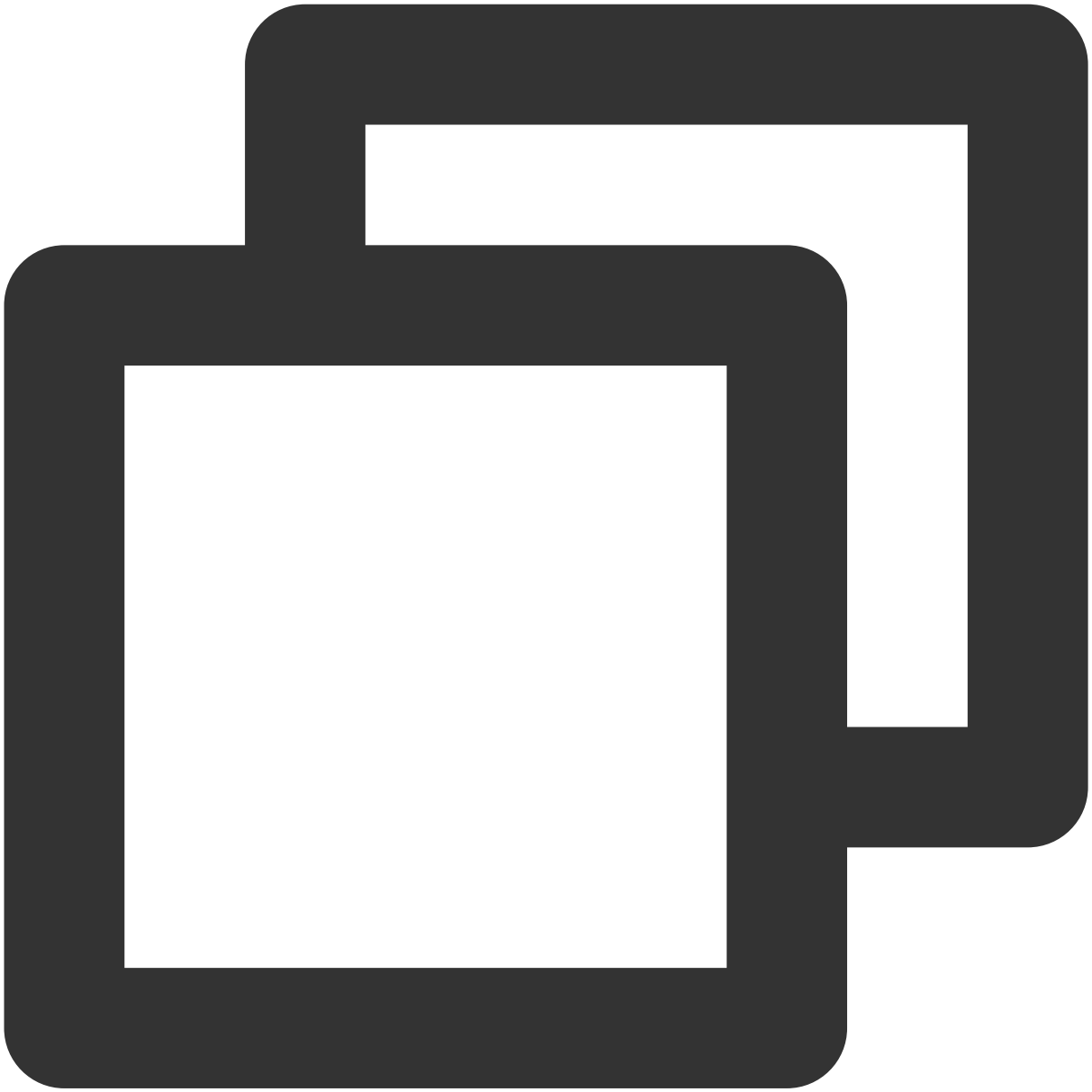
成功：成功则返回 AudioAuditingResponse 音频审核任务结果对象。
失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

查询音频审核任务

功能说明

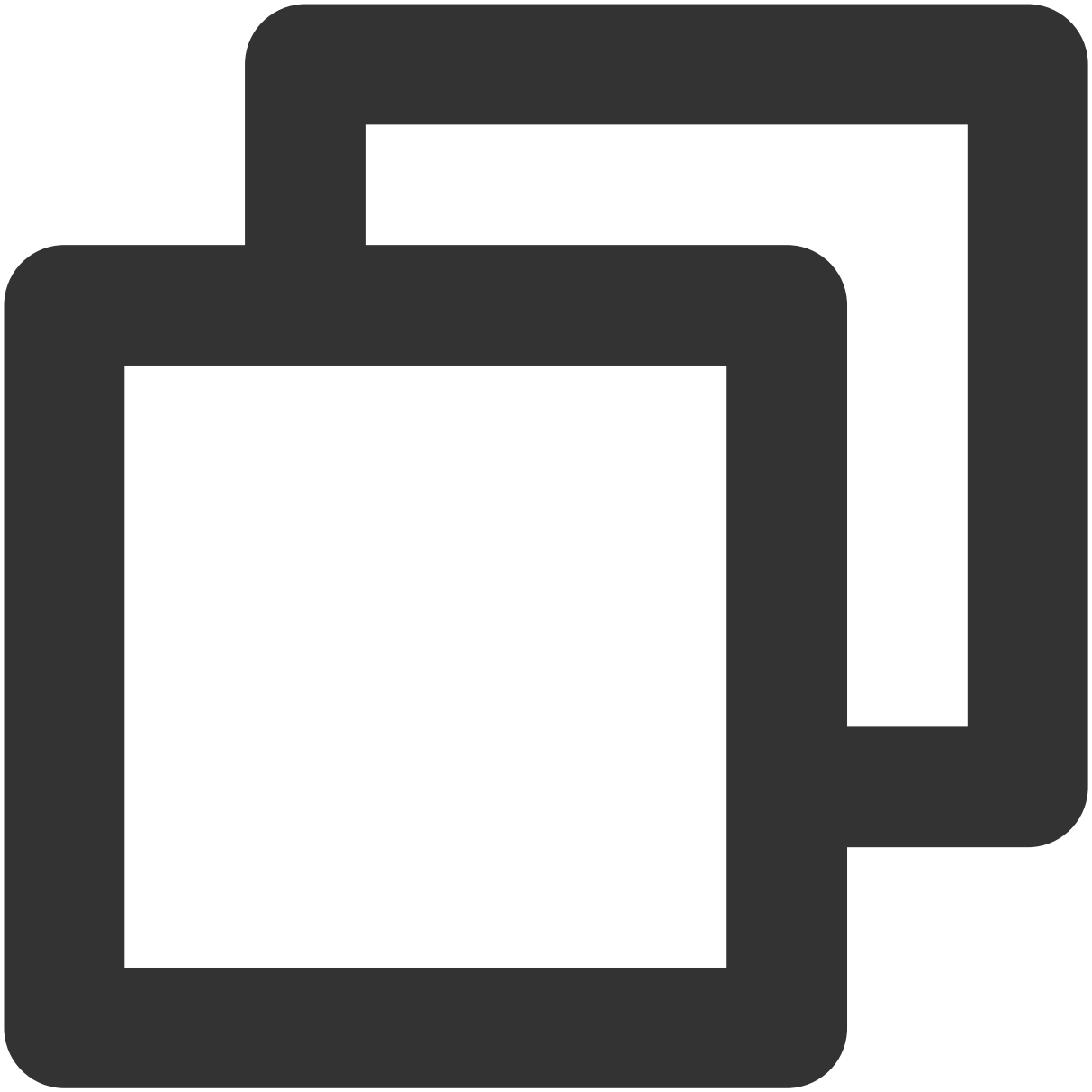
用于查询音频审核任务执行情况和结果。

方法原型



```
AudioAuditingResponse describeAuditingJob(AudioAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
AudioAuditingRequest request = new AudioAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("sacbf7269cbd2e11eba5325254009*****");
//3.调用接口,获取任务响应对象
AudioAuditingResponse response = client.describeAudioAuditingJob(request);
```

参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
jobId	需要查询的任务 ID	String	是

返回结果说明

成功： 返回 AudioAuditingResponse 审核任务结果对象，字段含义请参见 查询音频审核任务结果。
失败： 发生错误（例如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

文档审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文档审核的 API 概览和 SDK 示例代码。

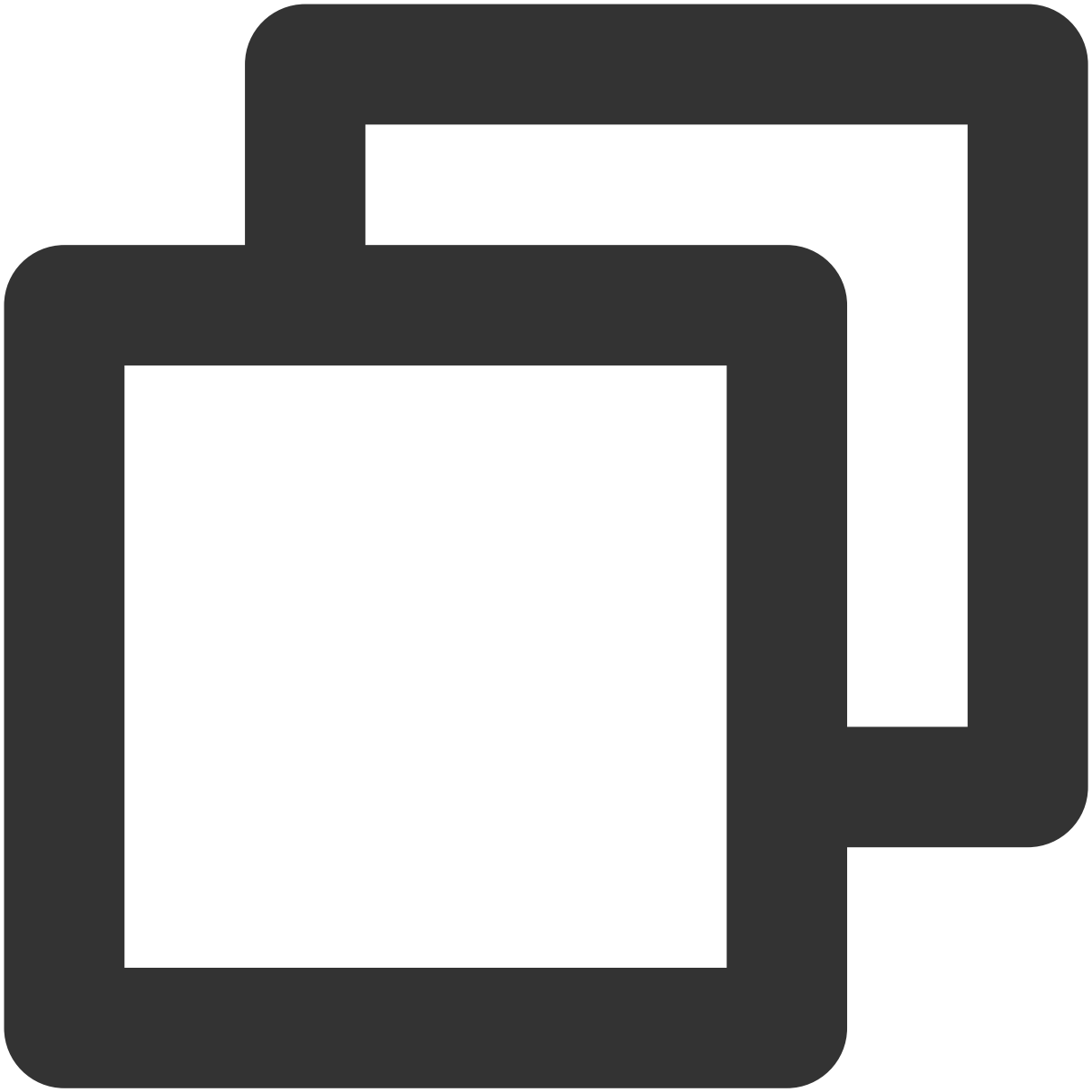
API	操作描述
提交文档审核任务	用于提交一个文档审核任务
查询文档审核任务结果	用于查询指定的文档审核任务

提交文档审核任务

功能说明

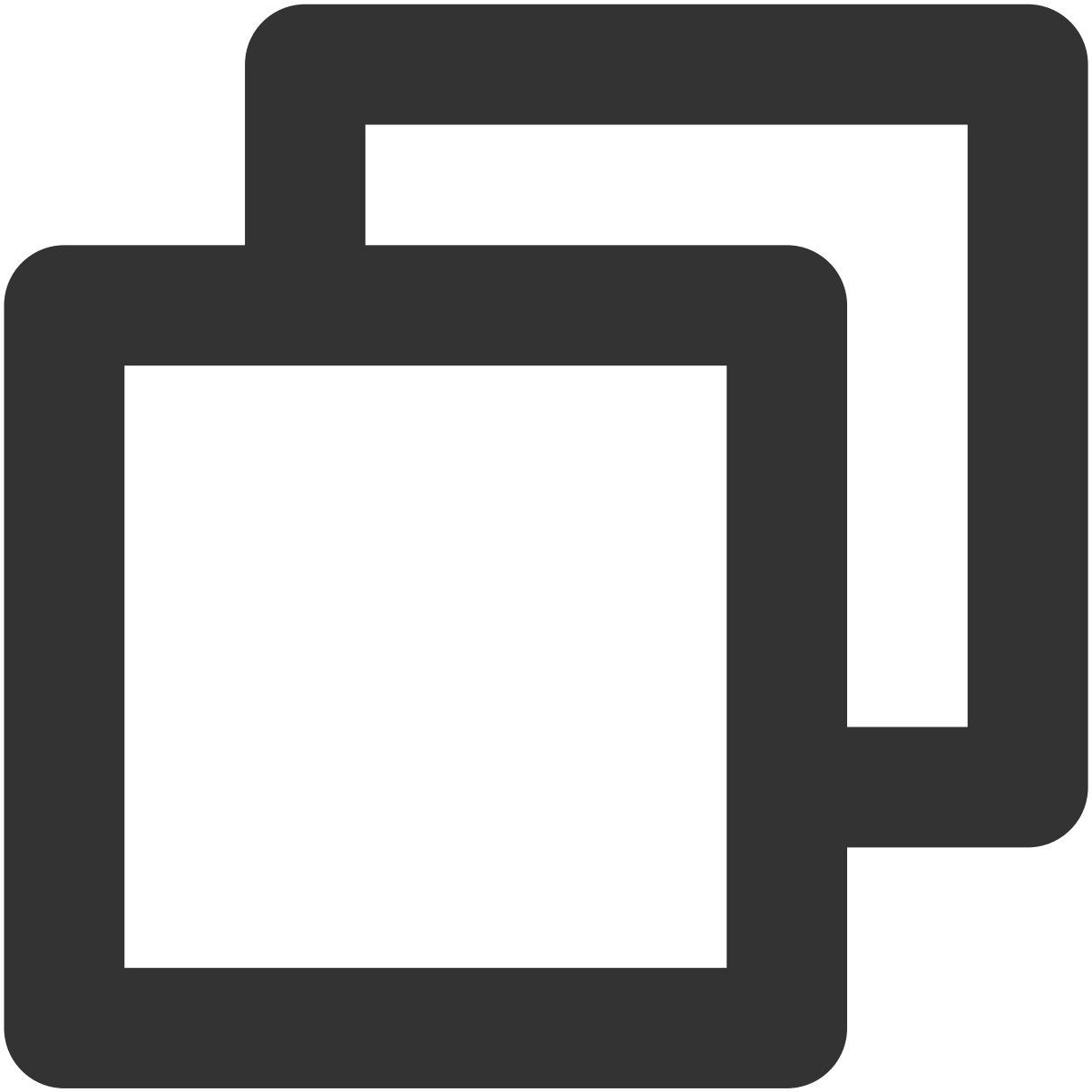
用于提交一个文档审核任务。

方法原型



```
DocumentAuditingResponse createAuditingDocumentJobs(DocumentAuditingRequest request
```

请求示例



```
//1.创建任务请求对象
DocumentAuditingRequest request = new DocumentAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
//2.1.1设置对象地址
// request.getInput().setObject("1.txt");
//2.1.2或直接设置文档地址
request.getInput().setUrl("https://markjrzhang-1251704708.cos.ap-chongqing.myqcloud
//2.2设置审核类型参数
request.getConf().setDetectType("all");
//3.调用接口,获取任务响应对象
```

```
DocumentAuditingResponse response = client.createAuditingDocumentJobs(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容	Container	是
Conf	Request	审核规则配置	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Url	Request.Input	文档文件的链接地址，例如 http://www.example.com/doctest.doc	String	是
Type	Request.Input	指定文档文件的类型，如未指定则默认以文件的后缀为类型。 如果文件没有后缀，该字段必须指定，否则会审核失败。例如：doc、docx、ppt、pptx 等	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置审核策略	String	否
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads	String	否
Callback	Request.Conf	审核结果可以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com`	String	否

返回结果说明

成功：成功则返回 DocumentAuditingResponse 文档审核任务结果对象。

版权所有：腾讯云计算（北京）有限责任公司

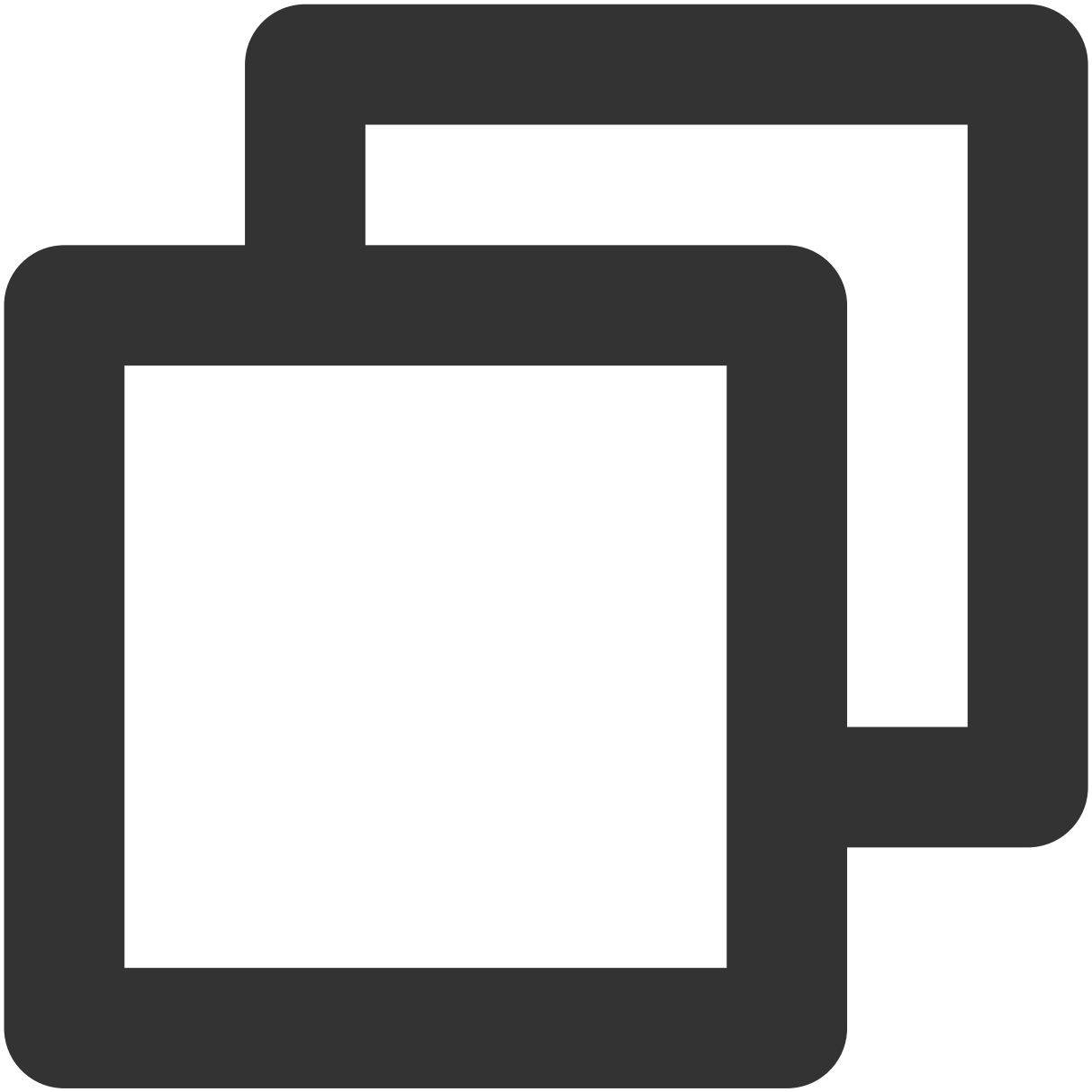
失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

查询文档审核任务

功能说明

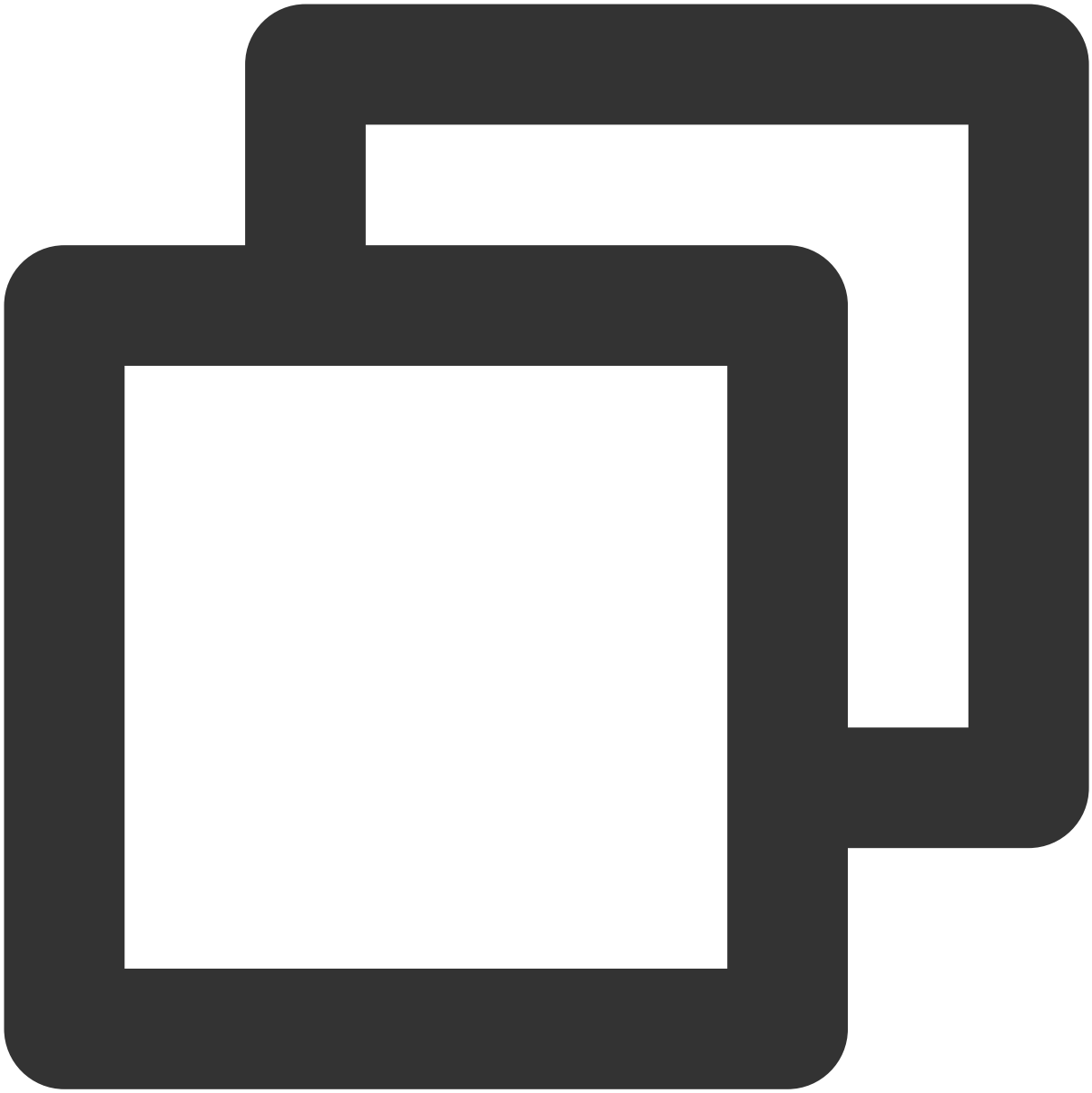
用于查询文档审核任务执行情况和结果。

方法原型



```
DocumentAuditingResponse describeAuditingJob(DocumentAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
DocumentAuditingRequest request = new DocumentAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("sdd5d1cc630fdc11ecb3fa5254009******");
//3.调用接口,获取任务响应对象
DocumentAuditingResponse response = client.describeAuditingDocumentJob(request);
```

参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
jobId	需要查询的任务 ID	String	是

返回结果说明

成功： 返回 DocumentAuditingResponse 审核任务结果对象。

失败： 发生错误（例如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

文本审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文本审核的 API 概览和 SDK 示例代码。

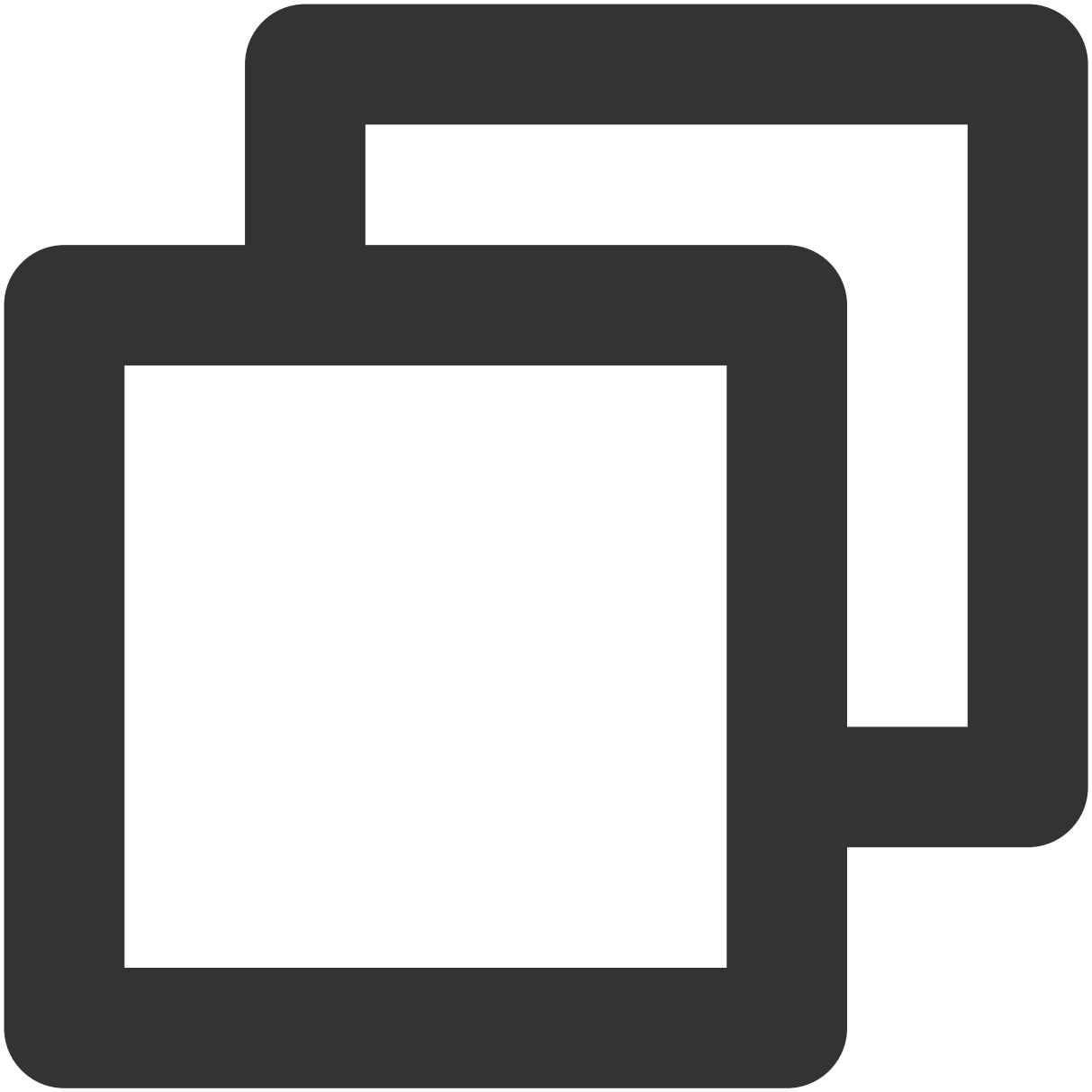
API	操作描述
提交文本审核任务	用于提交一个文本审核任务
查询文本审核任务结果	用于查询指定的文本审核任务

提交文本审核任务

功能说明

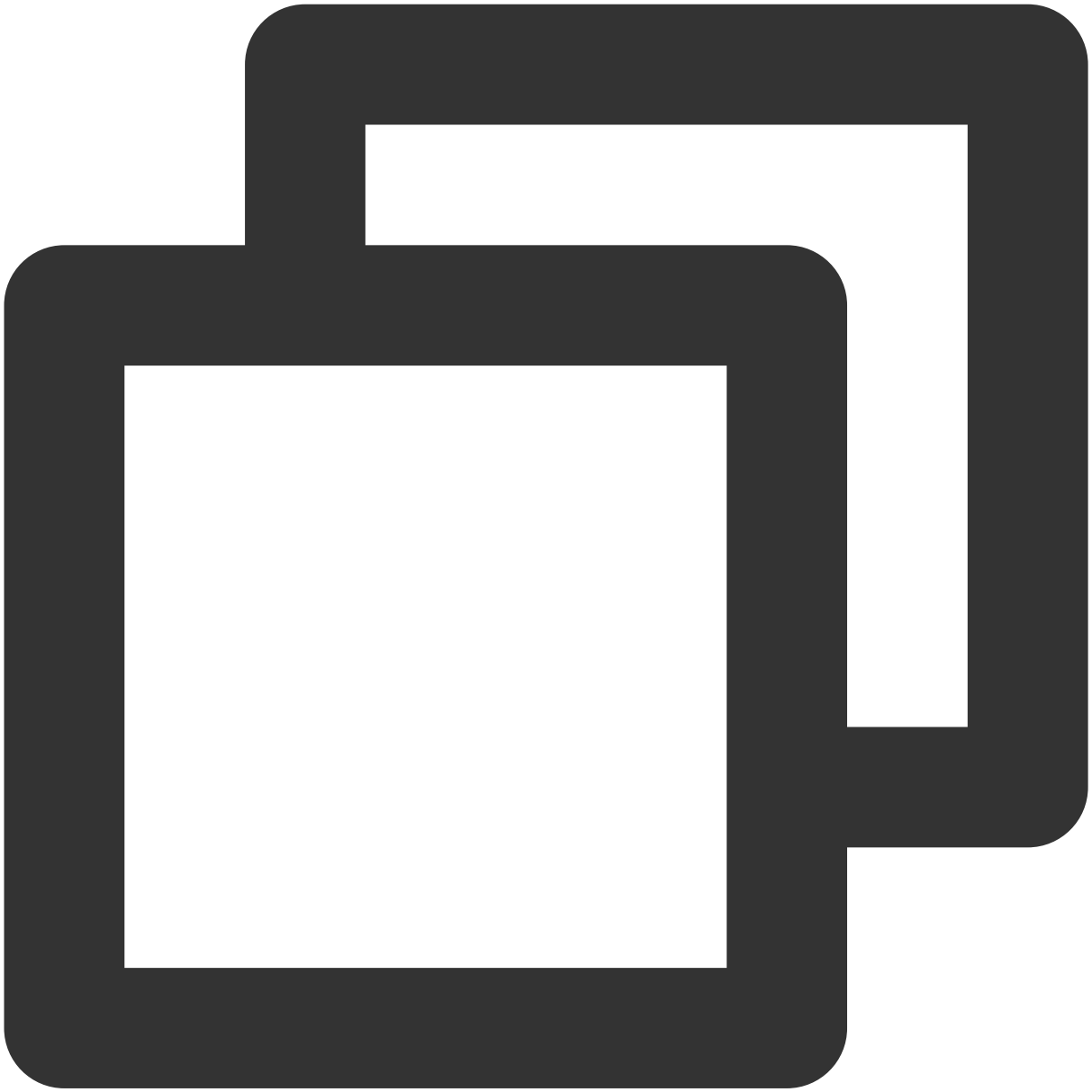
用于提交一个文本审核任务。

方法原型



```
TextAuditingResponse createAuditingTextJobs(TextAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
TextAuditingRequest request = new TextAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-12500000008");
//2.1.1设置对象地址
//request.getInput().setObject("1.txt");
//2.1.2或直接设置请求内容,文本内容的Base64编码
request.getInput().setContent("Base64Str");
//2.2设置审核类型参数
request.getConf().setDetectType("all");
//2.3设置审核模板（可选）
```

```
//request.getConf().setBizType("aa3e9d84a6a079556b0109a935c*****");
//3.调用接口,获取任务响应对象
TextAuditingResponse response = client.createAuditingTextJobs(request);
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	文本审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	当前 COS 存储桶中的文本文件名称，例如在目录 test 中的文件 img.jpg，则文件名称为 test/img.jpg，文本文件仅支持UTF8编码和 GBK 编码的内容，且文件大小不得超过1MB。	String	否
Content	Request.Input	当传入的内容为纯文本信息，需要先经过 base64 编码，文本编码前的原文长度不能超过10000个 utf8 编码字符。若超出长度限制，接口将会报错。	String	否

注意：

不可同时输入 Object 和 Content。

当选择 Object 时，审核结果为异步返回，可通过 查询文本审核任务结果 API 接口获取返回结果。

当选择 Content 时，审核结果为同步返回，可通过 响应体 查看返回结果。

目前仅支持中文、英文、阿拉伯数字的检测与审核。

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选

BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告）、Illegal（违法）、Abuse（谩骂），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	否
Callback	Request.Conf	审核结果可以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否

返回结果说明

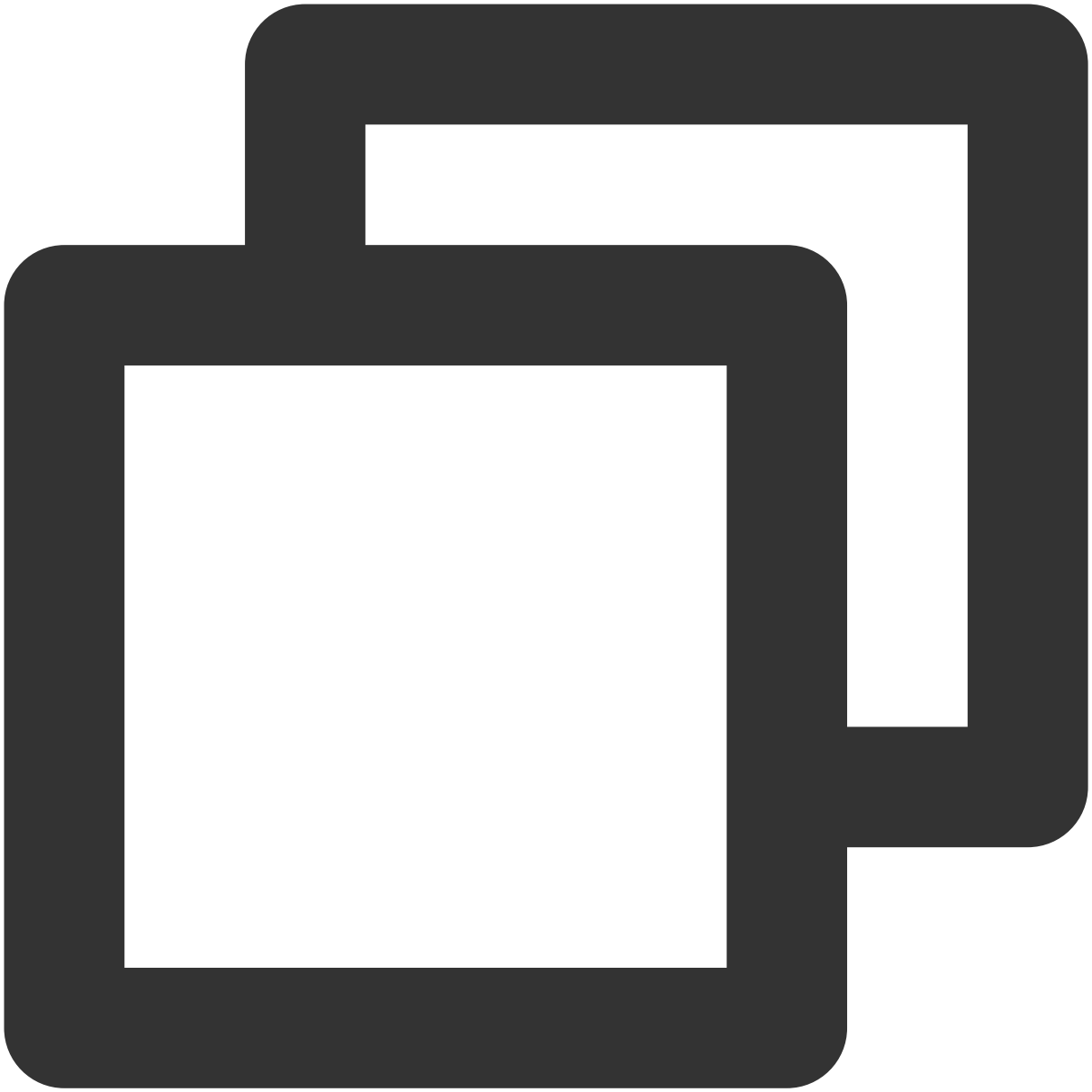
成功：成功则返回 TextAuditingResponse 文本审核任务结果对象。
失败：发生错误（例如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

查询文本审核任务

功能说明

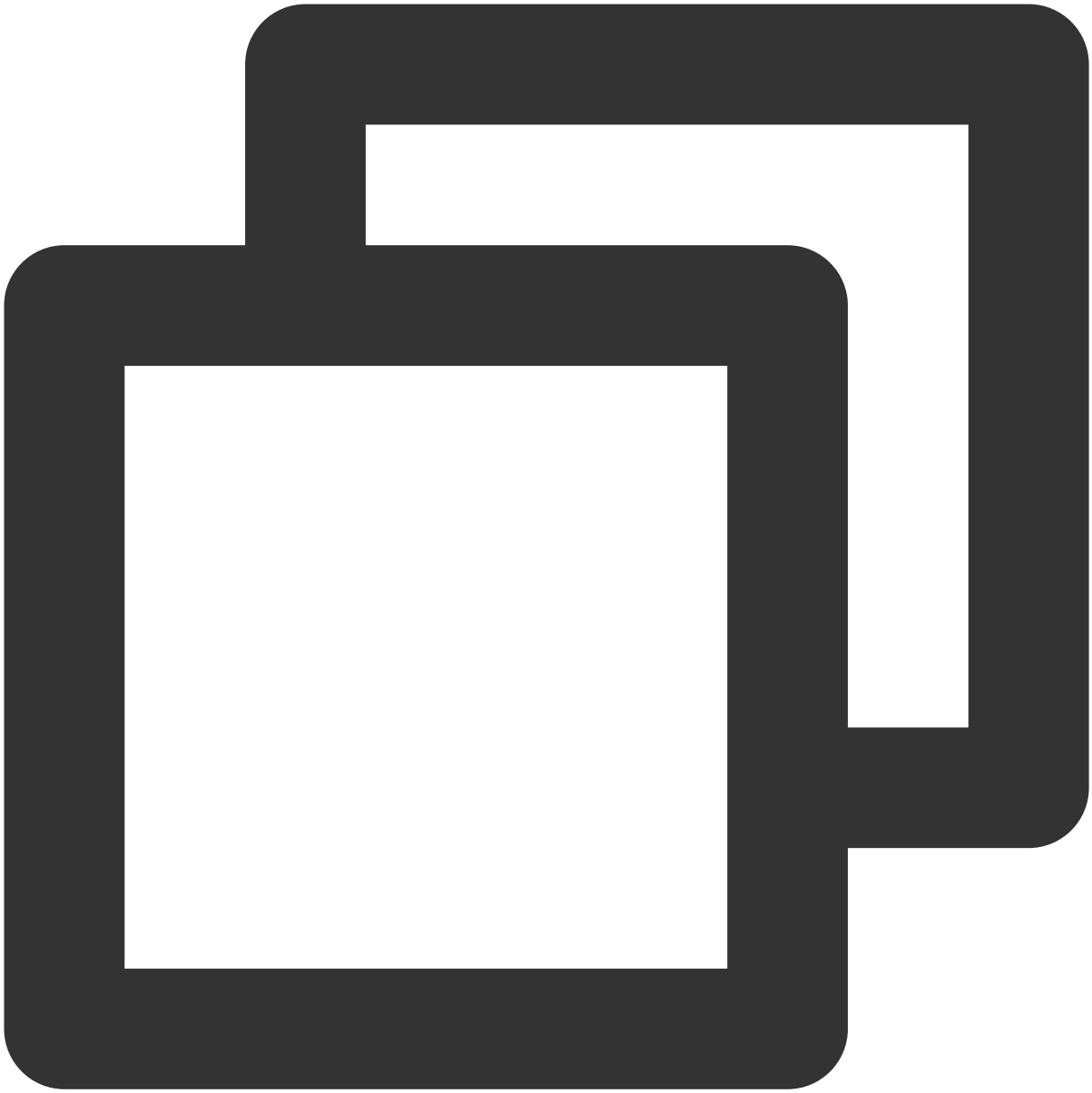
用于查询文本审核任务执行情况和结果。

方法原型



```
TextAuditingResponse describeAuditingTextJob(TextAuditingRequest request);
```

请求示例



```
//1.创建任务请求对象
TextAuditingRequest request = new TextAuditingRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("st68d08596f35011eb9324525400*****");
//3.调用接口,获取任务响应对象
TextAuditingResponse response = client.describeAuditingTextJob(request);
```

参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
jobId	需要查询的任务 ID	String	是

返回结果说明

成功： 返回 TextAuditingResponse 审核任务结果对象。

失败： 发生错误（例如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

回调信息处理

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象标准处理回调信息解析的 SDK 示例代码。

XML 回调内容解析

功能说明

将数据万象公共的 XML 回调信息解析为 Java 对象。

使用示例

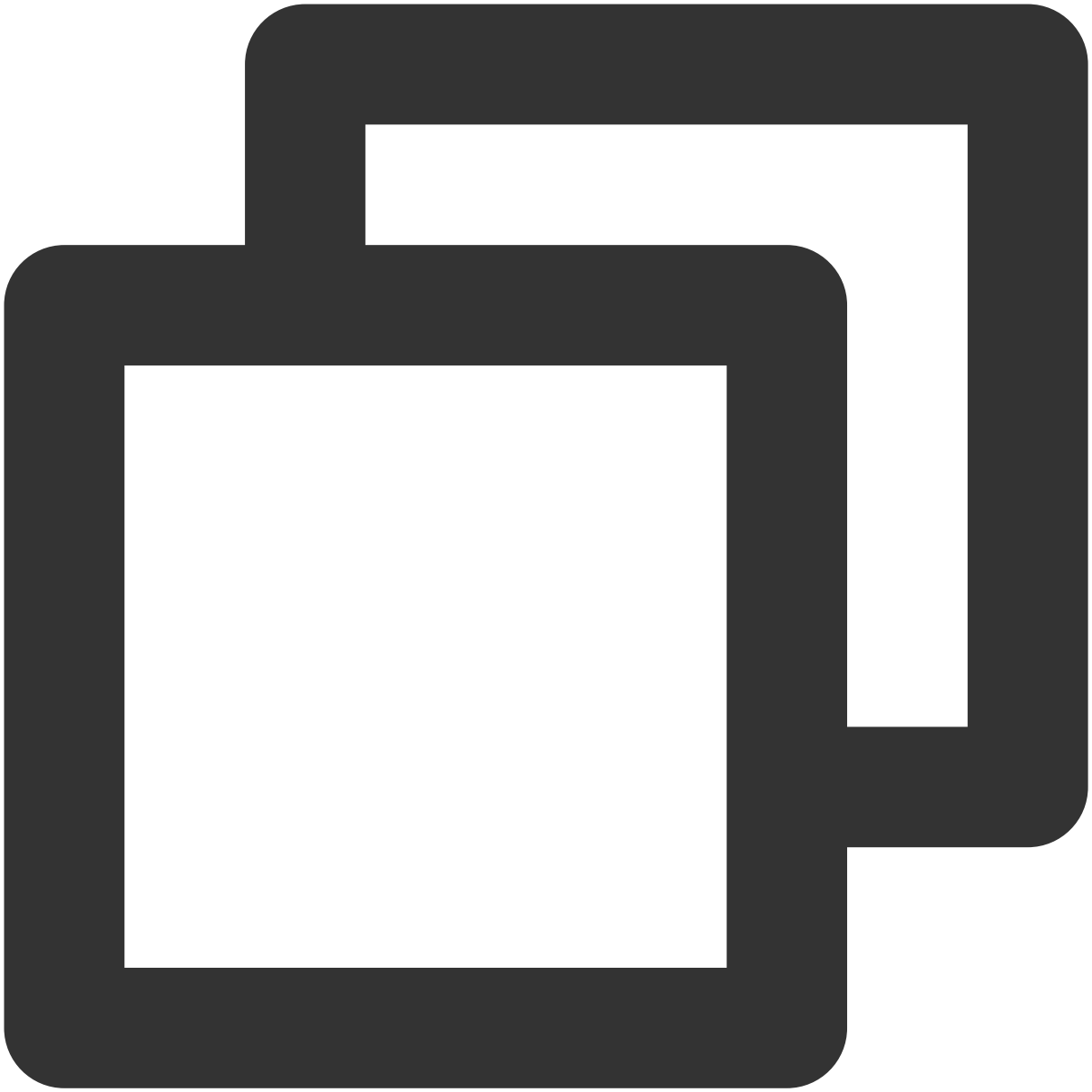
本示例用于解析回调数据，可通过获取 Unmarshallers 来进行 XML 数据的解析。

说明：

您可以通过查看 COSClient 的接口获取到需要使用的 XML 解析器，也可以通过查看功能对应的查询接口实现使用的是哪个 Unmarshaller。

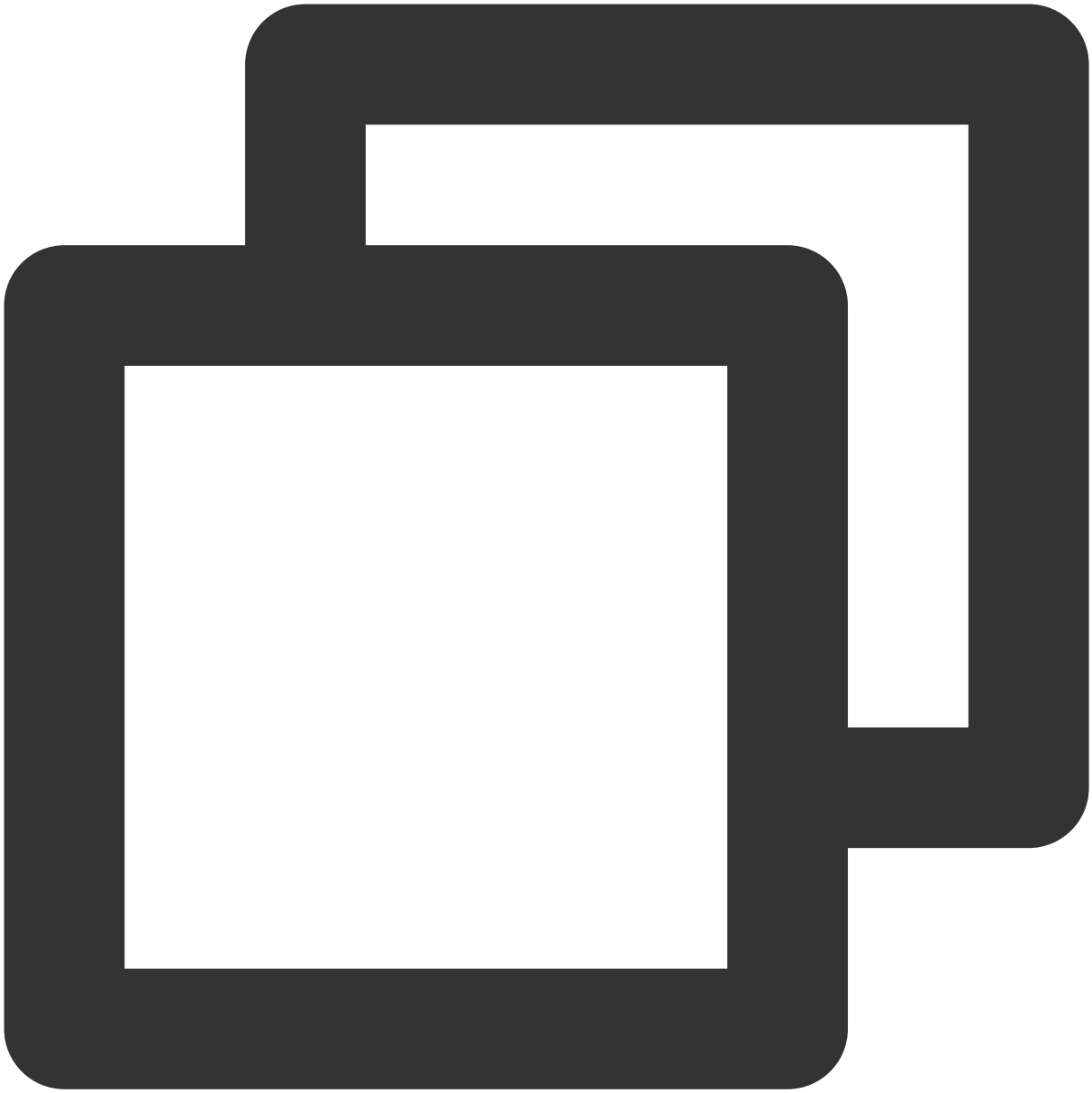
以文档预览回调信息为例，对应的查询接口为 describeDocProcessJob。

COSClient 中的代码片段，查看 invoke 方法中是使用哪个 Unmarshaller。



```
@Override
public DocJobResponse describeDocProcessJob(DocJobRequest request) {
  this.checkCIRequestCommon(request);
  CosHttpRequest<DocJobRequest> httpRequest = this.createRequest(request.getBucketN
  return this.invoke(httpRequest, new Unmarshallers.DescribeDocJobUnmarshaller());
}
```

使用 Unmarshaller 进行解析响应内容。



```
public static void processCINotifyResponse(String response) throws Exception {
  Unmarshallers.DescribeDocJobUnmarshaller describeDocJobUnmarshaller = new Unmarsh
  InputStream is = new ByteArrayInputStream(response.getBytes());
  DocJobResponse docJobResponse = describeDocJobUnmarshaller.unmarshall(is);
}
```

JSON 回调内容解析

功能说明

将数据万象公共的 JSON 回调信息解析为 Java 对象。

使用示例

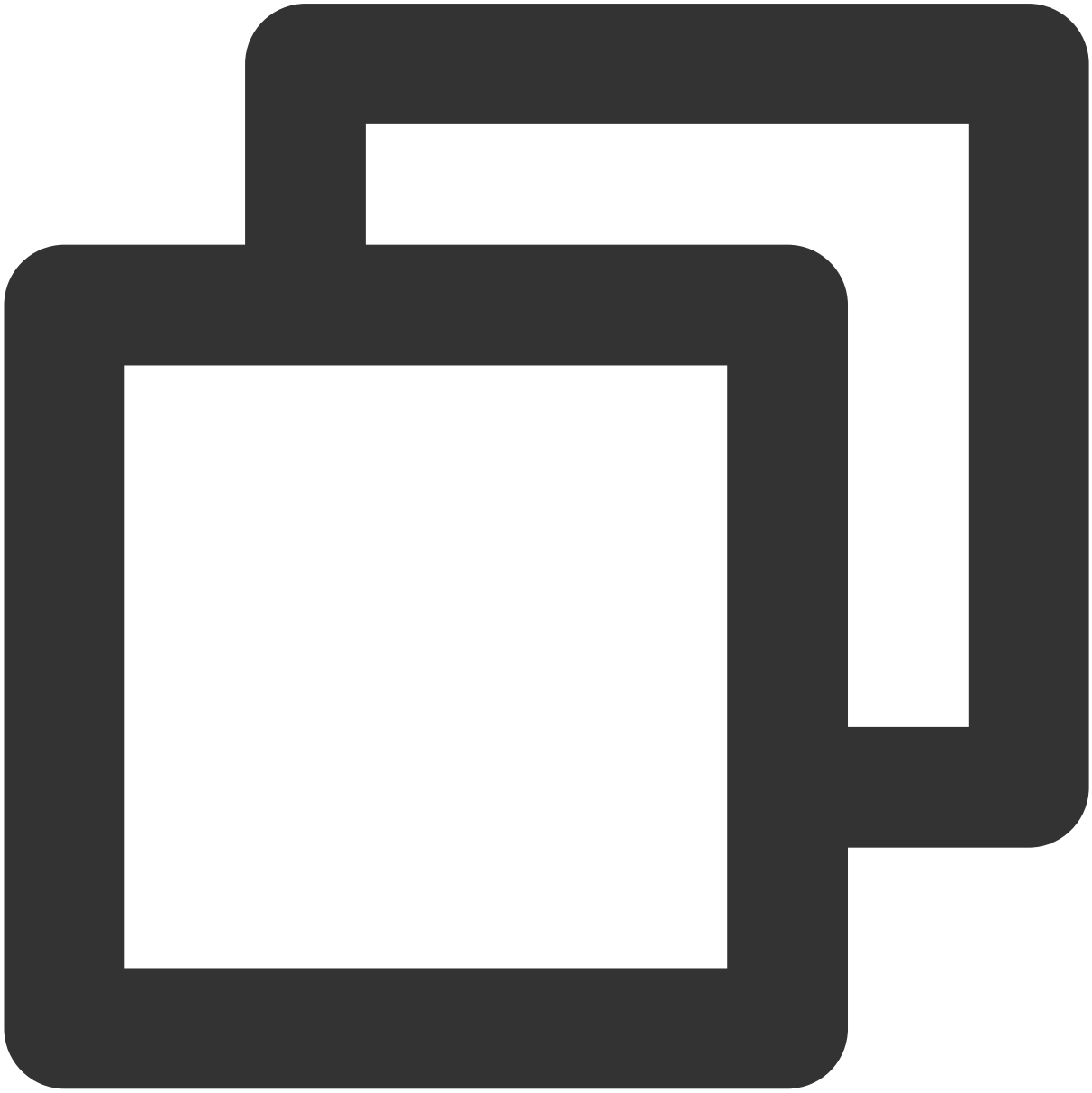
说明：

本质上处理 JSON 回调是将 JSON 转为 XML 后再进行处理。

Demo 中使用了 org.json 将 Json 转为 XML，并未在 SDK 的 Pom 中提供该依赖，需要自行导入。

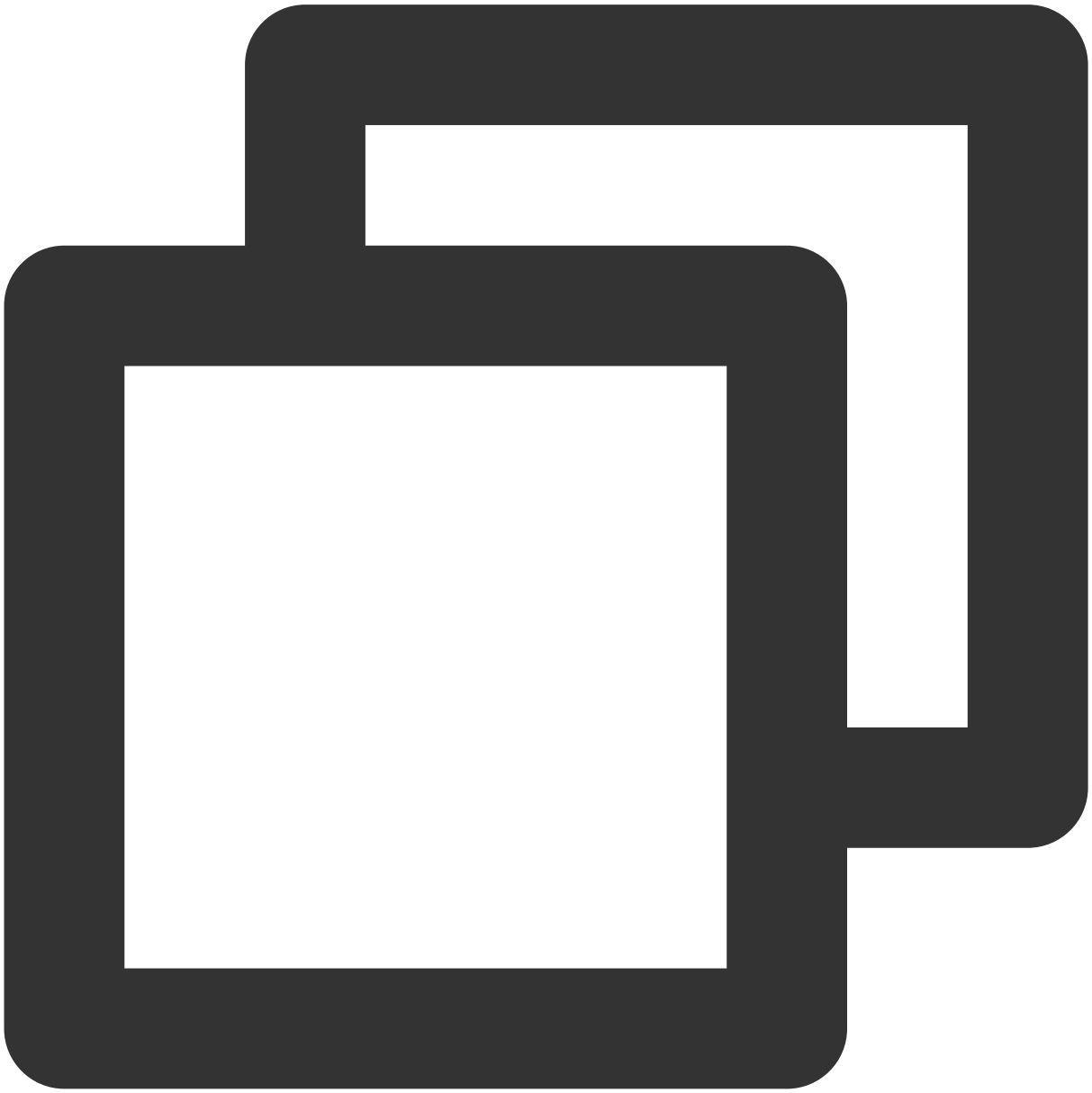
以图片内容审核回调接口为例，对应的查询接口为 describeAuditingImageJob。

COSClient 中的代码片段，查看 invoke 方法中是使用哪个 Unmarshaller。



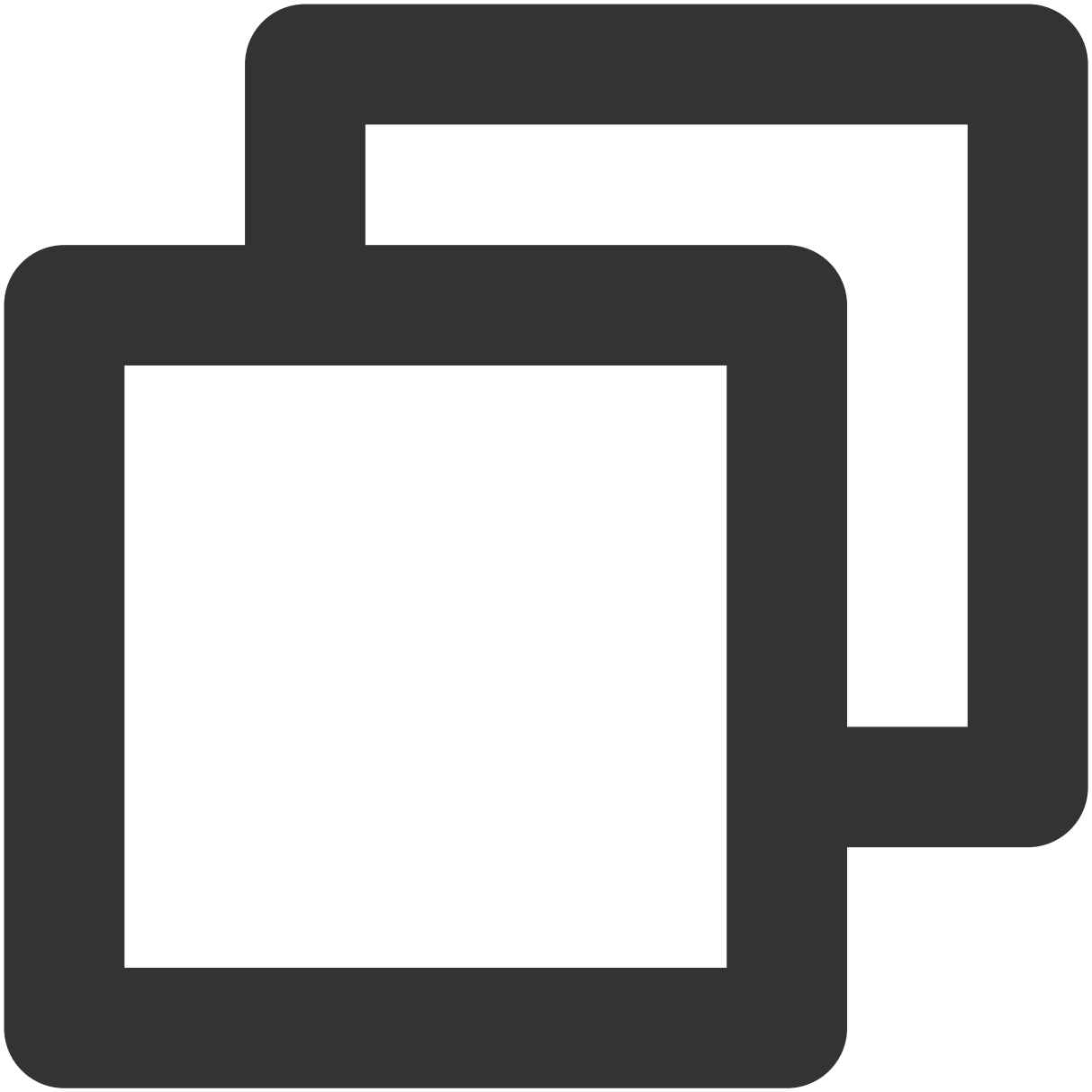
```
@Override
public ImageAuditingResponse describeAuditingImageJob(DescribeImageAuditingRequest
  rejectNull(imageAuditingRequest.getBucketName(),
          "The bucketName parameter must be specified setting the object tags");
  rejectNull(imageAuditingRequest.getJobId(),
          "The jobId parameter must be specified setting the object tags");
  CosHttpRequest<DescribeImageAuditingRequest> request = createRequest(imageAuditin
  return invoke(request, new Unmarshallers.ImageAuditingDescribeJobUnmarshaller());
}
```

将 JSON 响应转为 XML 后使用 Unmarshaller 进行解析响应内容。



```
//回调demo
public static void processJsonCINotifyResponse(String jsonResponse) throws Exceptio
  JSONObject response = new JSONObject(jsonResponse);
  JSONObject json = new JSONObject();
  json.put("Response",response);
  String xml = XML.toString(json);
  Unmarshallers.ImageAuditingDescribeJobUnmarshaller imageJobUnmarshaller = new Unm
  InputStream is = new ByteArrayInputStream(xml.getBytes());
  ImageAuditingResponse imageAuditingResponse = imageJobUnmarshaller.unmarshall(is)
}
```

上述示例使用的是 org.json.json，SDK 中并没有提供，如需使用请添加如下依赖。



```
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20180130</version>
</dependency>
```

异常处理

最近更新时间：2024-03-13 10:07:11

简介

调用 SDK 请求对象存储（Cloud Object Storage，COS）或数据万象（Cloud Infinite，CI）服务失败时，抛出的异常皆是 RuntimeExcpetion。
目前 SDK 常见的异常有 CosClientException，CosServiceException，CIServiceException 和 IllegalArgumentException。

客户端异常

CosClientException 是由于客户端无法和服务端完成正常的交互而导致的失败。例如客户端无法连接到服务端，无法解析服务端返回的数据，读取本地文件发生 IO 异常等。CosClientException 继承自 RuntimeException，没有自定义的成员变量，使用方法同 RuntimeException。

服务端异常

CosServiceException 和 CIServiceException 用于指交互正常完成，但是操作失败的场景。例如客户端访问一个不存在 Bucket，删除一个不存在的文件，没有权限进行某个操作，服务端故障异常等。
CosServiceException 包含了服务端返回的状态码，requestid，出错明细等。捕获异常后，建议对整个异常进行打印，异常包含了必须的排查因素。以下是异常成员变量的描述：

request 成员	描述	类型
requestId	请求 ID，用于表示一个请求，对于排查问题十分重要	String
traceId	辅助排查问题的 ID	String
statusCode	response 的 status 状态码，4xx 是指请求因客户端而失败，5xx 是服务端异常导致的失败。请参见 COS 错误码	String
errorType	枚举类，表示异常的种类，分为 Client，Service，Unknown	ErrorType
errorCode	请求失败时 body 返回的 Error Code 请参见 COS 错误码	String
errorMessage	请求失败时 body 返回的 Error Message 请参见 COS 错误码	String

版权所有：腾讯云计算（北京）有限责任公司

使用自助诊断工具

针对请求可能遇到不同的报错情况，我们为您提供了 CI 自助诊断工具，帮助您快速定位问题，调试报错代码。

使用步骤

1. 复制异常处理返回的 RequestId（请求 ID）。
2. 单击 CI 自助诊断工具，进入自助诊断页面。
3. 在顶部的 RequestId 输入框中，输入待诊断的 RequestId，并单击**开始诊断**。
4. 稍侯片刻，便能看到相应的智能诊断结果。

常见问题

若您在使用 Java SDK 过程中，有相关的疑问，请参见 常见问题 文档。

文件处理
文件哈希值计算

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于数据万象文件处理哈希值计算任务的相关 API 概览以及 SDK 示例代码。

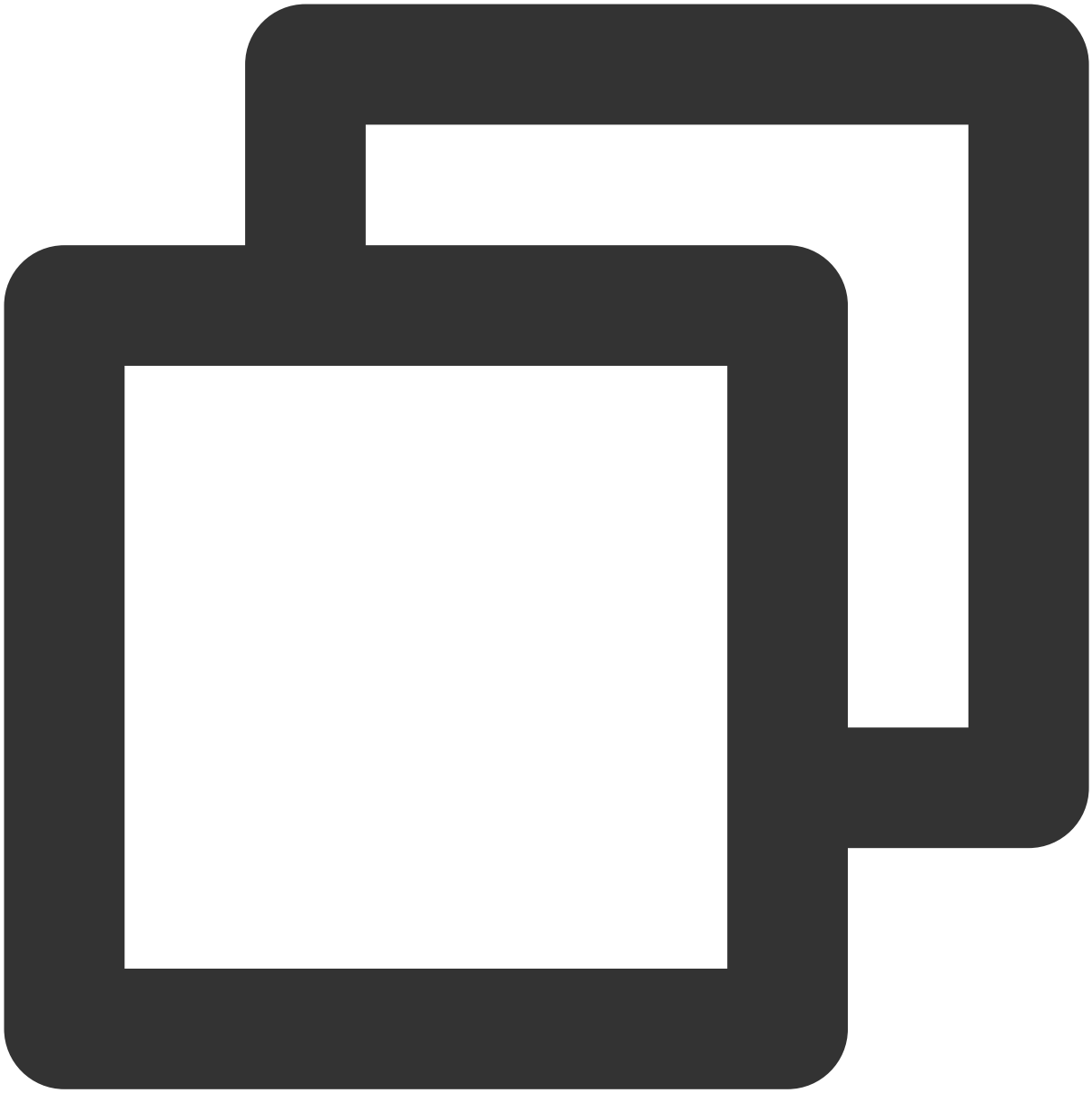
API	操作描述
提交哈希值计算任务	创建一个文件处理哈希值计算任务
查询哈希值计算结果	查询一个文件处理哈希值计算结果任务

提交哈希值计算任务

功能说明

以提交任务的方式进行文件哈希值计算，异步返回计算得到的哈希值。

方法原型



```
public FileProcessJobResponse createFileProcessJob(FileProcessRequest request);
```

参数说明

Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
tag	Request	表示任务的类型，哈希值计算任务类型为：FileHashCode。	String	是

operation	Request	包含哈希值计算的处理规则。	Container	是
queueId	Request	任务所在的队列 ID。	String	是
callBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
callBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
callBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
callBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情请参见 CallBackMqConfig	Container	否

Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
object	Request.Input	文件名，取值为文件在当前存储桶中的完整名称。	String	是

Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
fileHashCodeConfig	Request.Operation	指定哈希值计算的处理规则。	Container	是
userData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否

FileHashCodeConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
type	Request.Operation.FileHashCodeConfig	哈希值的算法类型，有效值：MD5、SHA1、SHA256。	String	是
addToHeader	Request.Operation.FileHashCodeConfig	是否将计算得到的哈希值添加至文件自定义 header, 有	String	否

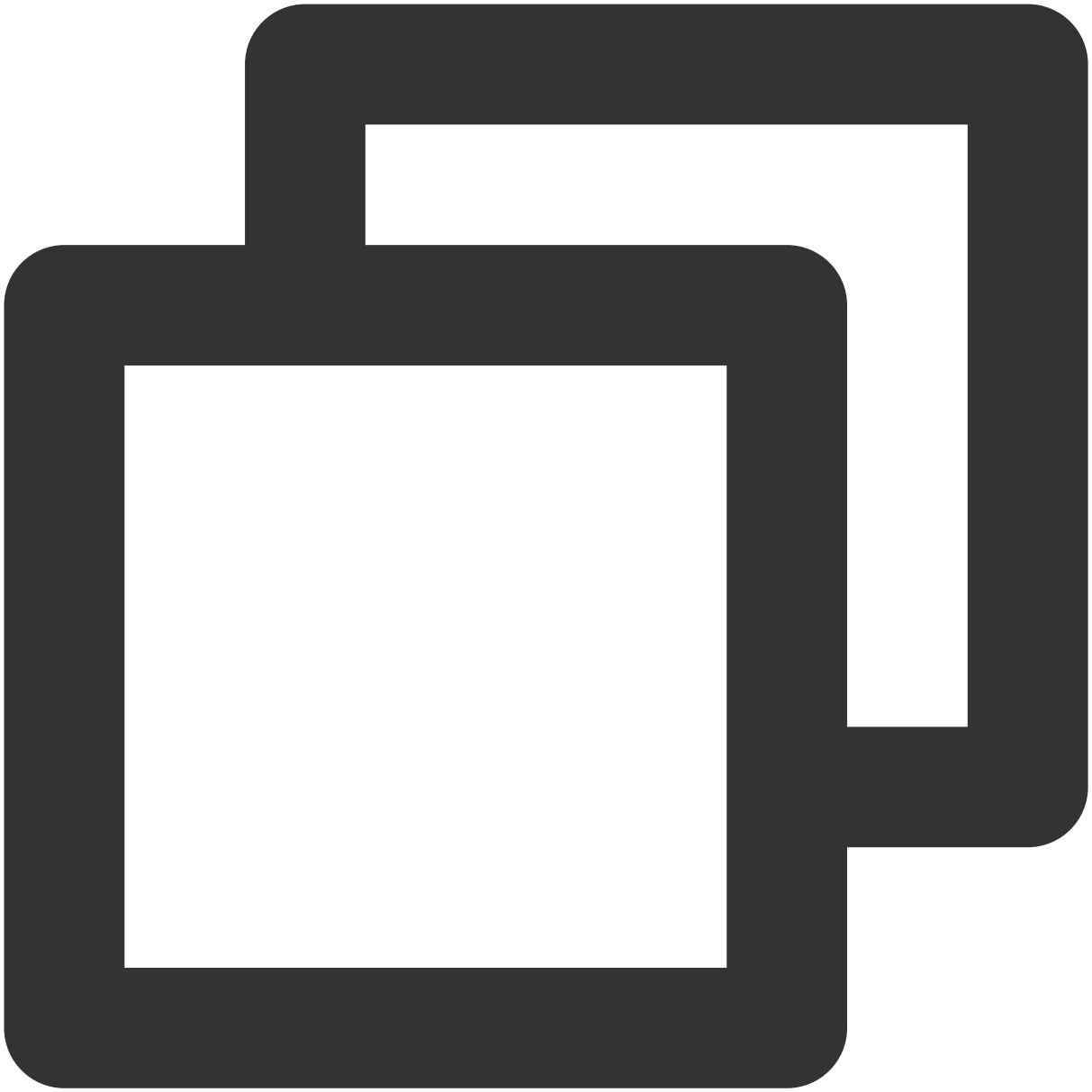
版权所有：腾讯云计算（北京）有限责任公司

		效值：true、false，默认值为 false。 自定义 header根据 `Type` 的值变化，例如 `Type` 值为MD5 时，自定义 heade r为 x-cos-meta-md5。		

返回结果说明

成功：返回 FileProcessJobResponse 对象响应信息。
失败：发生错误（如 Bucket 不存在），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
//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.FileHashCode);
request.getInput().setObject("input/1.mp4");
FileHashCodeConfig fileHashCodeConfig = request.getOperation().getFileHashCodeConfi
fileHashCodeConfig.setType("MD5");
fileHashCodeConfig.setAddToHeader("true");
request.setQueueId("p1ff062b35a494cf0ac4b572df22a****");
//3.调用接口,获取任务响应对象
```

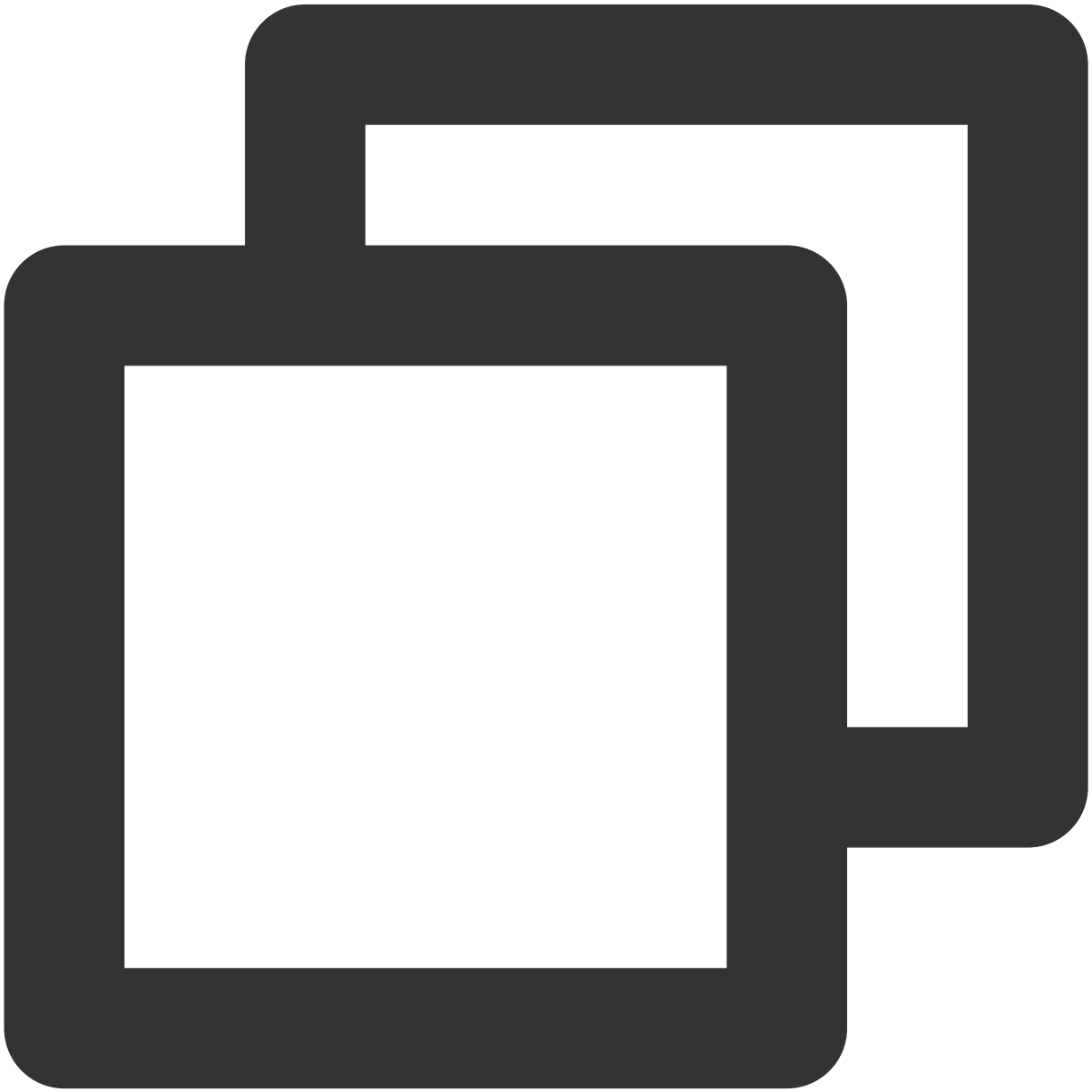
```
FileProcessJobResponse response = client.createFileProcessJob(request);
```

查询哈希值计算结果

功能说明

查询一个文件处理任务，根据任务 ID 查询任务详情。

方法原型



```
public FileProcessJobResponse describeFileProcessJob(FileProcessRequest request);
```

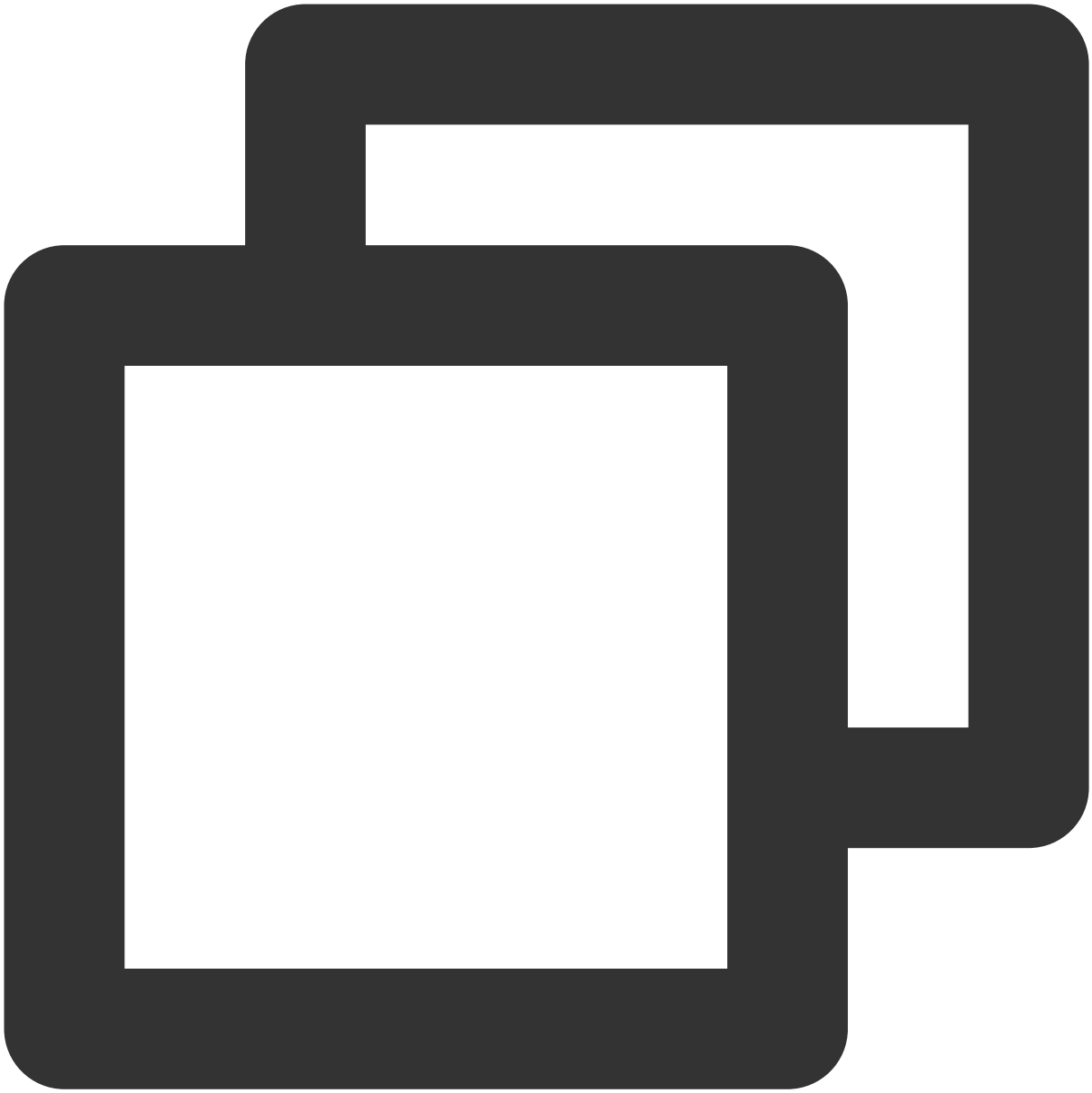
参数说明

参数名称	描述	类型	是否必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
jobId	要查询的任务 ID	String	是

返回结果说明

成功： 返回任务详情响应包装类，类中包含一个 FileProcessJobResponse 任务详情对象。
失败： 发生错误（如身份认证失败），抛出异常 CosClientException 或者 CosServiceException。详情请参见 异常处理。

请求示例



```
//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.setJobId("fda7eb1607b8411ed8c182156726*****");
//3.调用接口,获取任务响应对象
FileProcessJobResponse response = client.describeFileProcessJob(request);
```

JavaScript SDK 快速入门

最近更新时间：2024-03-13 10:07:10

下载与安装

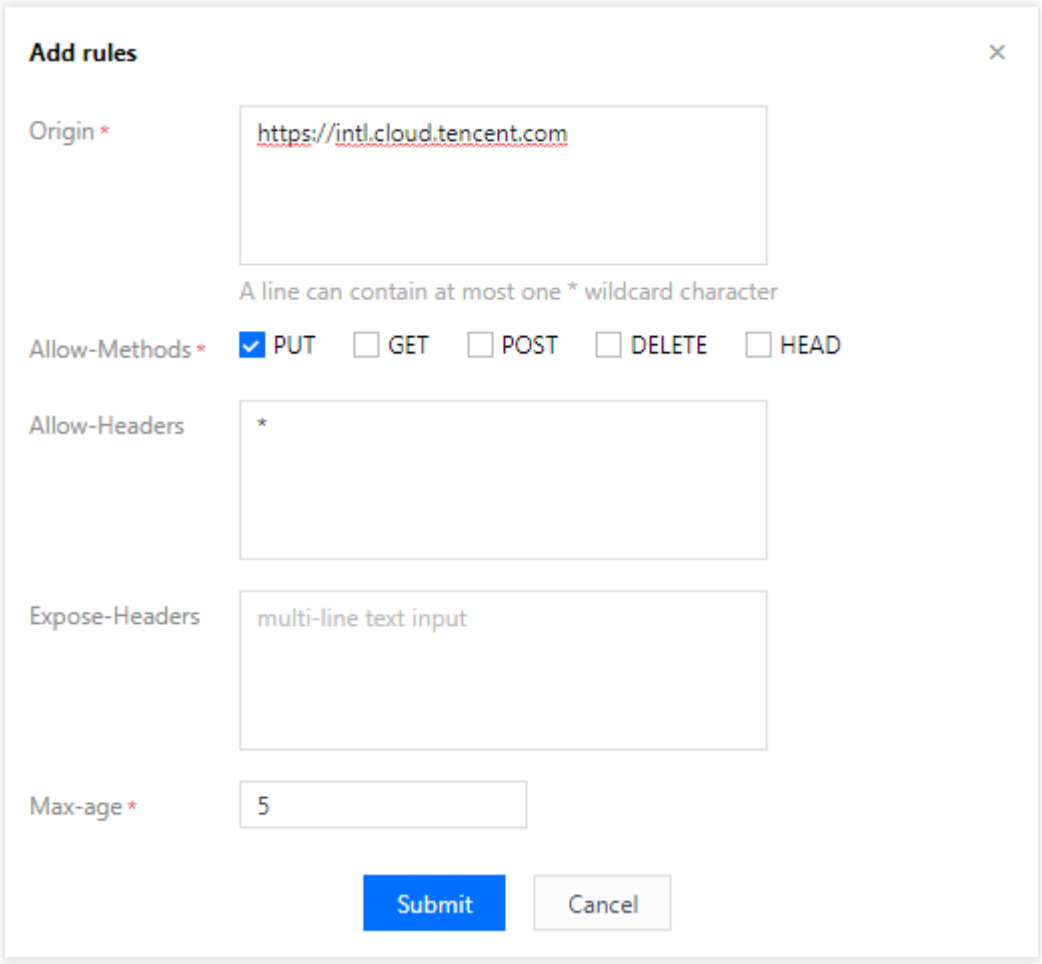
相关资源

对象存储 COS 的 XML JS SDK 源码下载地址：XML JavaScript SDK。
SDK 快速下载地址：XML JavaScript SDK。
演示示例 Demo 下载地址：XML JavaScript SDK Demo。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：JavaScript SDK 常见问题。

说明：
如果您在使用 XML 版本 SDK 时遇到函数或方法不存在等错误，请先将 XML 版本 SDK 升级到最新版再重试。

准备环境

1. JavaScript SDK 需浏览器支持基本的 HTML5 特性（支持 IE10 以上浏览器），以便支持 ajax 上传文件和计算文件 MD5 值。
2. 登录 对象存储控制台，创建存储桶。获取存储桶名称和 地域名称。
3. 登录 访问管理控制台，获取您的项目 SecretId 和 SecretKey。
4. 配置 CORS 规则，AllowHeader 需配成 `*`，ExposeHeaders 需要 ETag、Content-Length 以及其他 js 需要读取的 header 字段，如下图所示。操作详情请参见 设置跨域访问 文档。



说明：

关于本文中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。

关于跨端框架（例如 uni-app）的使用说明，使用 JavaScript SDK 开发后无法打包成正常使用的移动应用，例如 Android App、iOS App，需要使用对应的 Android SDK、iOS SDK。

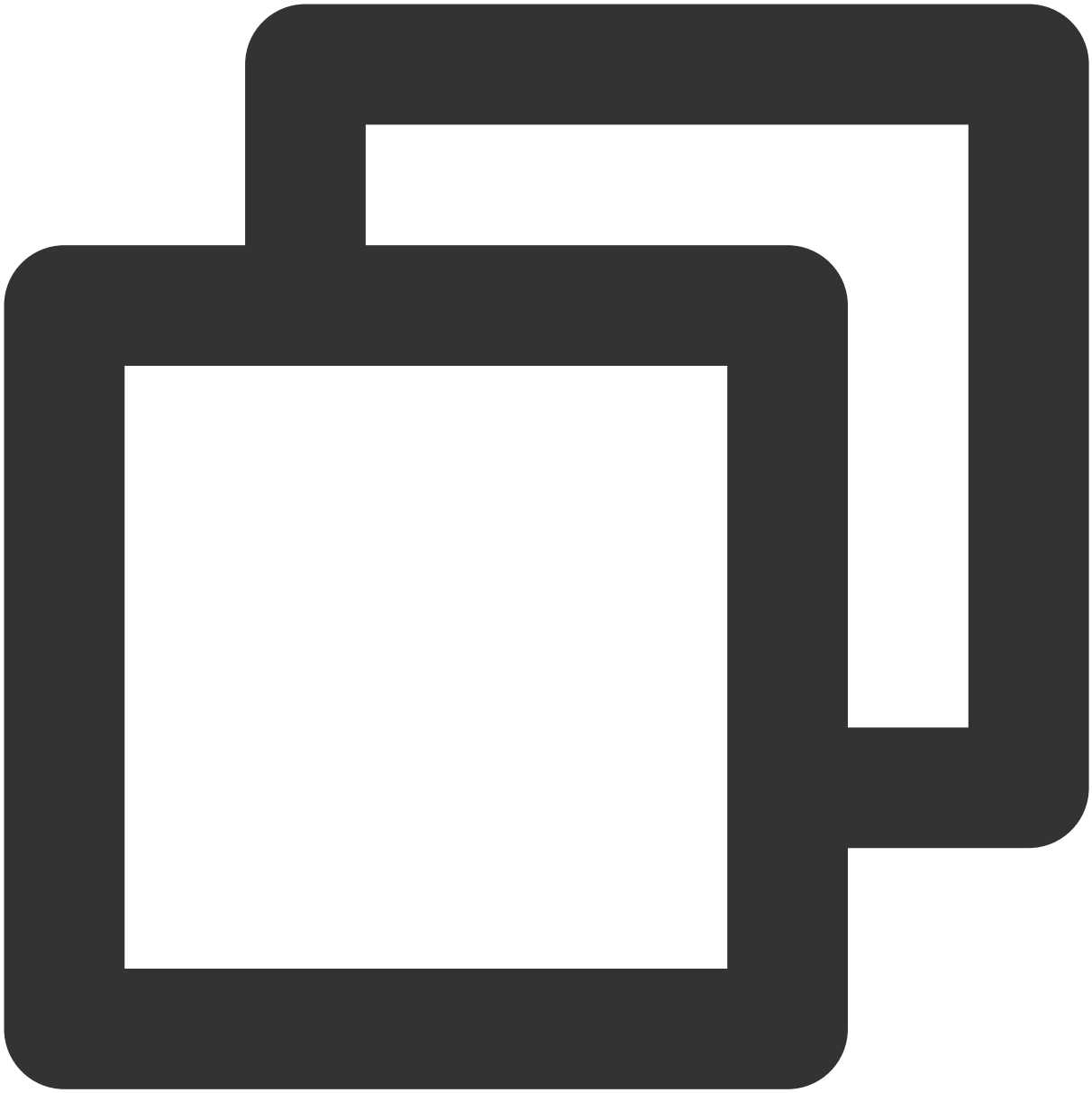
安装 SDK

您可以通过以下方式安装 SDK：

script 引入

说明：

建议将sdk下载到本地后引入，选择 地址一或 地址二 下载最新 cos-js-sdk-v5.min.js。

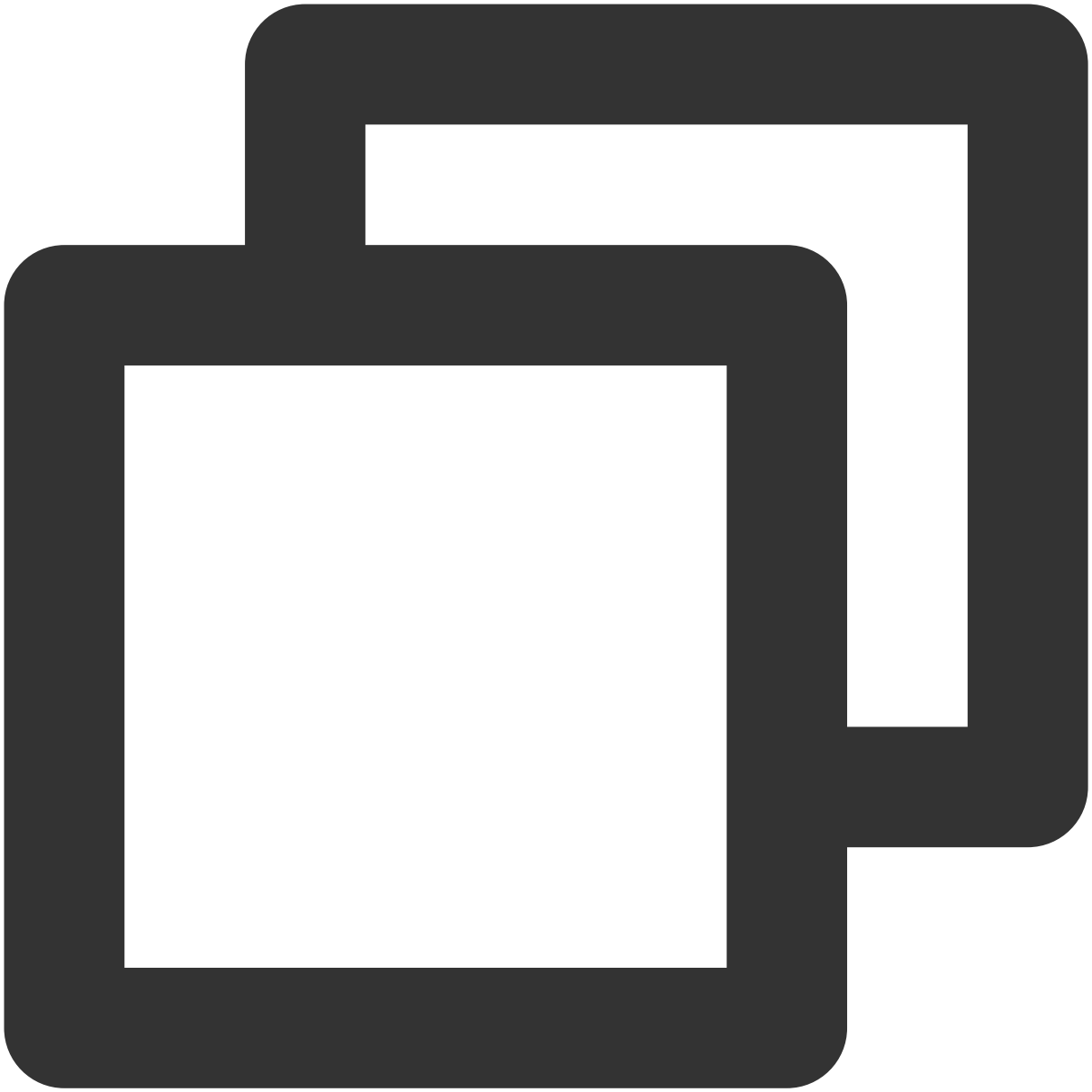


```
<!--src为本地路径 根据自己项目目录结构来调整-->
<script src="../dist/cos-js-sdk-v5.min.js"></script>
```

在 script 标签引用 SDK 时，SDK 占用了全局变量名 COS，通过它的构造函数可以创建 SDK 实例。

webpack 引入方式

通过 `npm i cos-js-sdk-v5 --save` 安装 SDK 依赖，支持 webpack 打包的场景，您可以用 npm 引入作为模块，代码如下:



```
var COS = require('cos-js-sdk-v5');

// 或

import COS from 'cos-js-sdk-v5';
```

开始使用

注意：

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

获取临时密钥

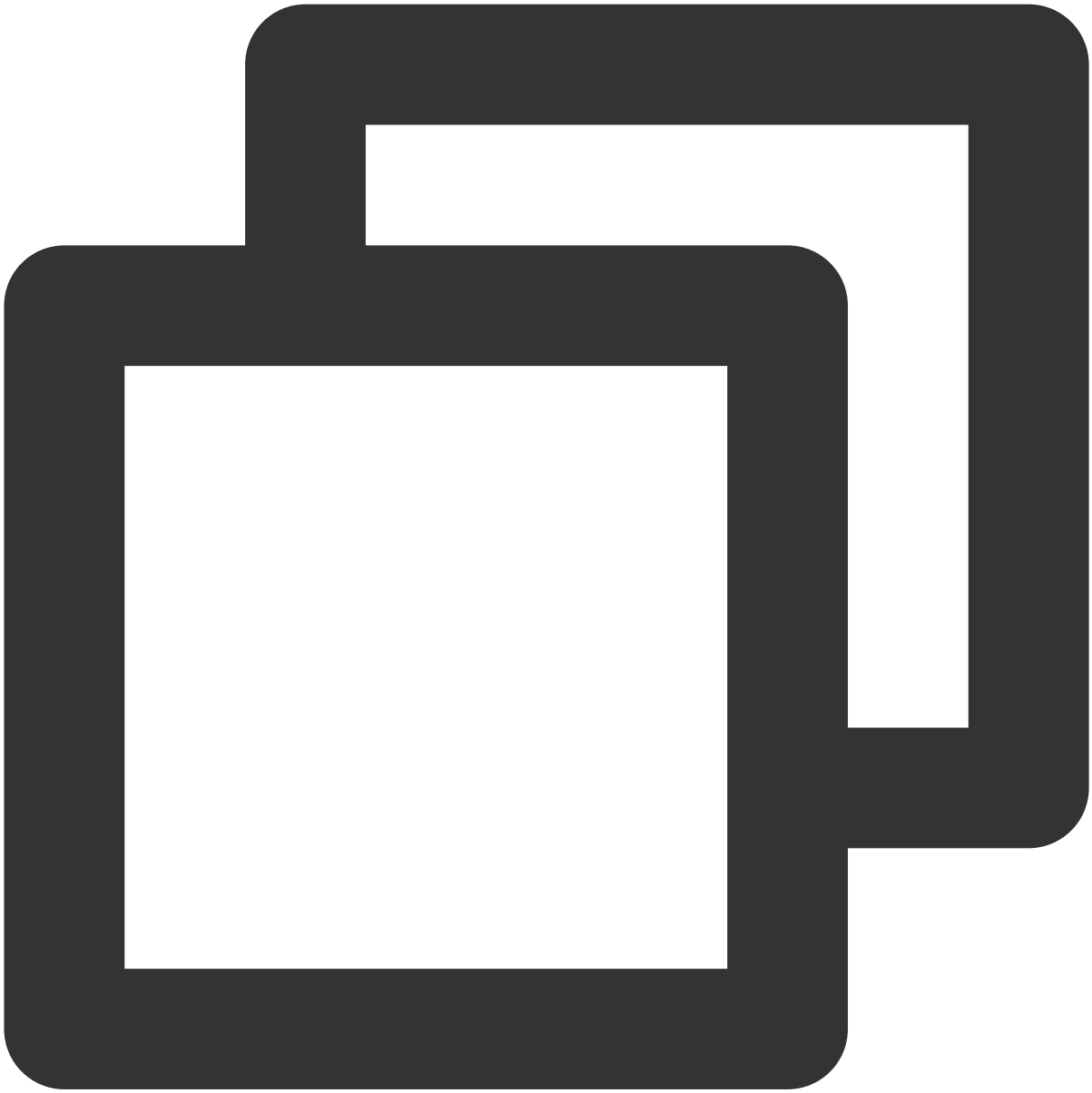
由于固定密钥放在前端会有安全风险，正式部署时我们推荐使用临时密钥的方式，实现过程为：前端首先请求服务端，服务端使用固定密钥调用 STS 服务申请临时密钥（具体内容请参见 临时密钥生成和使用指引 文档），然后返回临时密钥到前端使用。

注意：

如果站点有登录态，这个获取临时密钥接口，还需要加上登录态校验。

初始化

1. 创建 web.html，填入如下代码，并修改 web.html 中的存储桶名称和 Region。
2. 部署好后端的临时密钥服务，并修改 getAuthorization 里的密钥服务地址。
3. 把 web.html 放在 Web 服务器下，然后在浏览器访问页面，测试文件上传。web.html 文件示例代码如下：



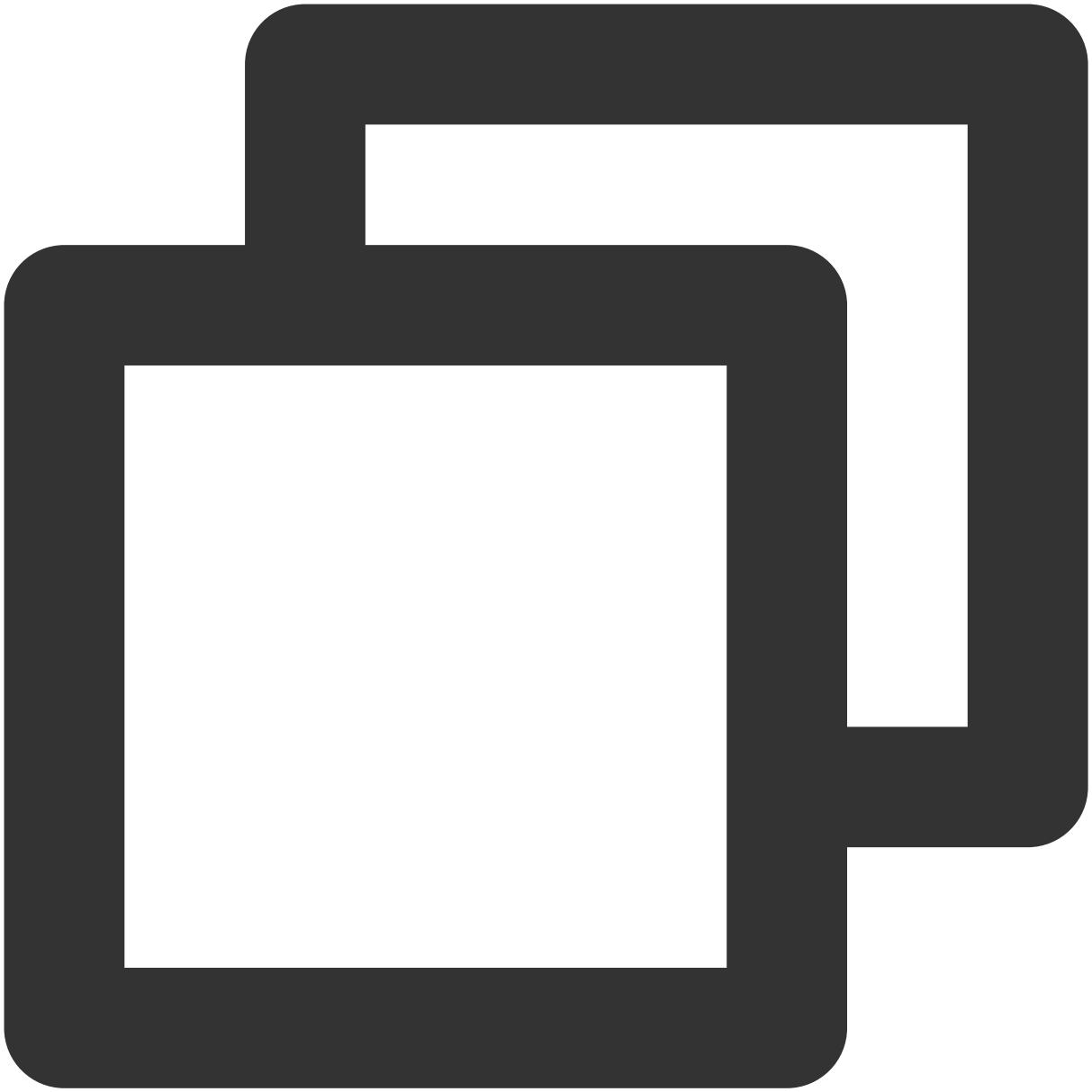
```
<input id="file-selector" type="file">
<script src="dist/cos-js-sdk-v5.min.js"></script>
<script>

// 存储桶名称，由 bucketname-appid 组成，appid 必须填入，可以在 COS 控制台查看存储桶名称。 ht
var Bucket = 'examplebucket-1250000000';  /* 存储桶，必须字段 */

// 存储桶 region 可以在 COS 控制台指定存储桶的概览页查看 https://console.tencentcloud.com/
// 关于地域的详情见 https://www.tencentcloud.com/document/product/436/6224
var Region = 'COS_REGION';     /* 存储桶所在地域，必须字段 */
```

```
// 初始化实例
var cos = new COS({
    // getAuthorization 必选参数
    getAuthorization: function (options, callback) {
        // 初始化时不会调用，只有调用 cos 方法（例如 cos.putObject）时才会进入
        // 异步获取临时密钥
        // 服务端 JS 和 PHP 例子：https://github.com/tencentyun/cos-js-sdk-v5/blob/mas
        // 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-st
        // STS 详细文档指引看：https://www.tencentcloud.com/document/product/436/14048

        var url = 'http://example.com/server/sts.php'; // url 替换成您自己的后端服务
        var xhr = new XMLHttpRequest();
        xhr.open('GET', url, true);
        xhr.onload = function (e) {
            try {
                var data = JSON.parse(e.target.responseText);
                var credentials = data.credentials;
            } catch (e) {
            }
            if (!data || !credentials) {
              return console.error('credentials invalid:\\n' + JSON.stringify(data,
            };
            callback({
              TmpSecretId: credentials.tmpSecretId,
              TmpSecretKey: credentials.tmpSecretKey,
              SecurityToken: credentials.sessionToken,
              // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致签名错误
              StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
              ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000000
            });
        };
        xhr.send();
    }
});

// 接下来可以通过 cos 实例调用 COS 请求。

</script>
```

配置项

使用示例

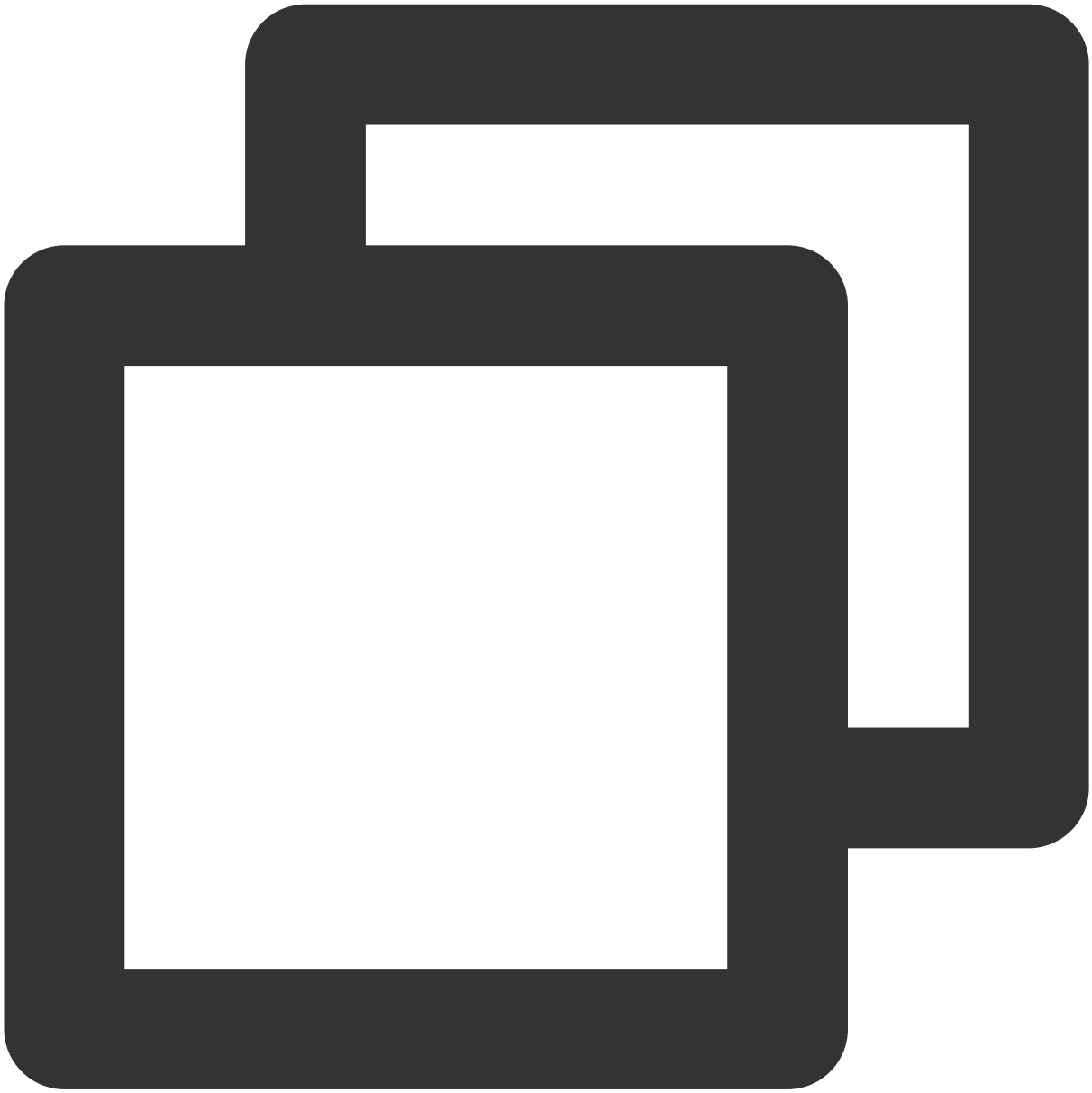
创建一个 COS SDK 实例，COS SDK 支持以下格式创建：
格式一（推荐）：后端通过获取临时密钥给到前端，前端计算签名。



```
var COS = require('cos-js-sdk-v5');
var cos = new COS({
    // getAuthorization 必选参数
    getAuthorization: function (options, callback) {
        // 异步获取临时密钥
        // 服务端 JS 和 PHP 例子：https://github.com/tencentyun/cos-js-sdk-v5/blob/mas
        // 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-st
        // STS 详细文档指引看：https://www.tencentcloud.com/document/product/436/14048

        var url = 'http://example.com/server/sts.php'; // url 替换成您自己的后端服务
        var xhr = new XMLHttpRequest();
```

```
        xhr.open('GET', url, true);
        xhr.onload = function (e) {
            try {
                var data = JSON.parse(e.target.responseText);
                var credentials = data.credentials;
            } catch (e) {
            }
            if (!data || !credentials) {
              return console.error('credentials invalid:\\n' + JSON.stringify(data,
            };
            callback({
              TmpSecretId: credentials.tmpSecretId,
              TmpSecretKey: credentials.tmpSecretKey,
              SecurityToken: credentials.sessionToken,
              // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致签名错误
              StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
              ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000000
            });
        };
        xhr.send();
    }
});
```

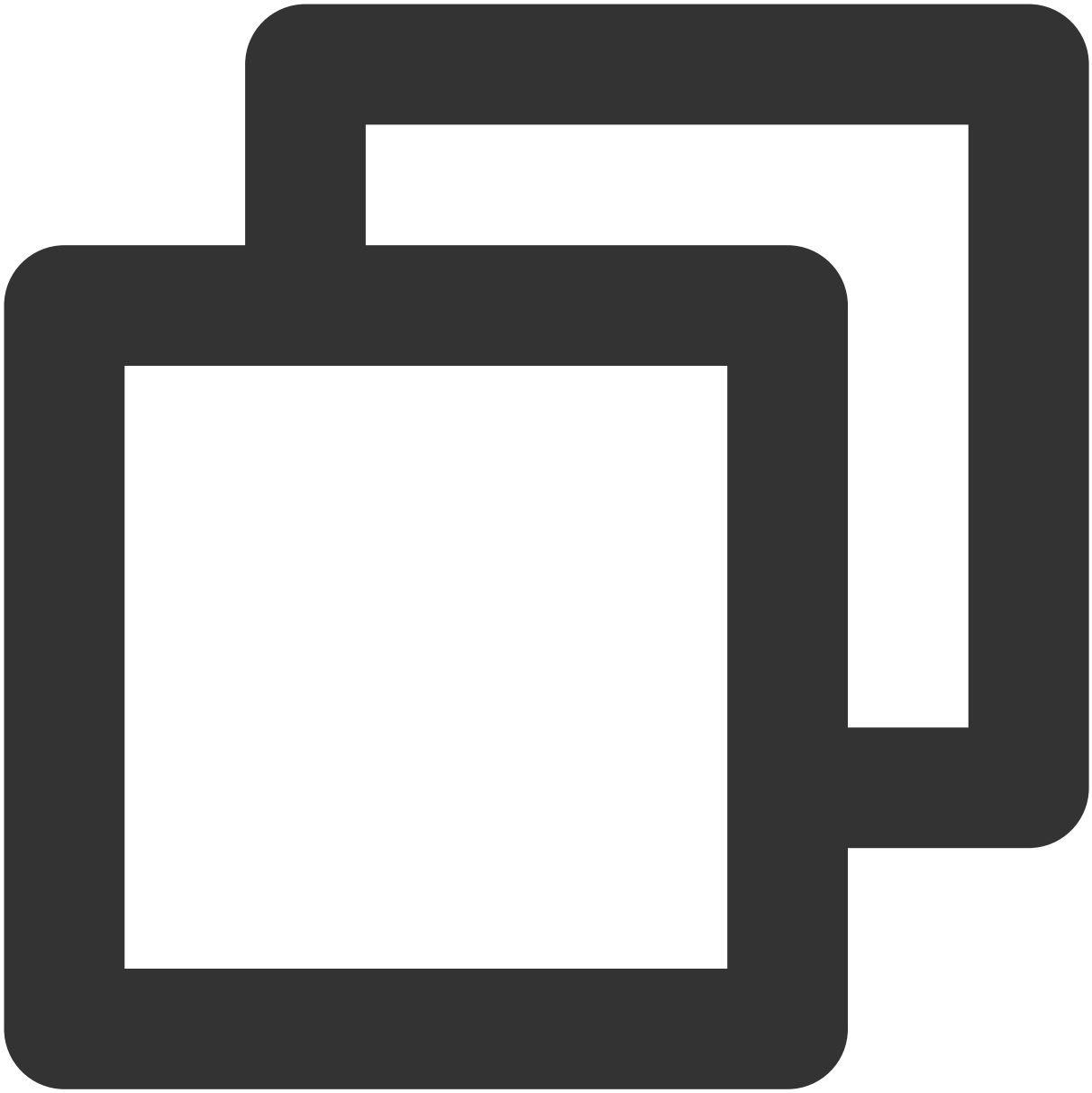
格式二（推荐）：细粒度控制权限，后端通过获取临时密钥给到前端，只有在相同请求时，前端才重复使用临时密钥，后端可以通过 Scope 细粒度控制权限。



```
var COS = require('cos-js-sdk-v5');
var cos = new COS({
    // getAuthorization 必选参数
    getAuthorization: function (options, callback) {
        // 服务端例子：https://github.com/tencentyun/qcloud-cos-sts-sdk/blob/master/s
        // 异步获取临时密钥
        var url = 'http://example.com/server/sts.php'; // url 替换成您自己的后端服务
        var xhr = new XMLHttpRequest();
        xhr.open('POST', url, true);
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.onload = function (e) {
```

```
            try {
                var data = JSON.parse(e.target.responseText);
                var credentials = data.credentials;
            } catch (e) {
            }
            if (!data || !credentials) {
                return console.error('credentials invalid:\\n' + JSON.stringify(dat
            };
            callback({
                TmpSecretId: credentials.tmpSecretId,
                TmpSecretKey: credentials.tmpSecretKey,
                SecurityToken: credentials.sessionToken,
                // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致签名
                StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
                ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000000
                ScopeLimit: true, // 细粒度控制权限需要设为 true，会限制密钥只在相同请求时
            });
        };
        xhr.send(JSON.stringify(options.Scope));
    }
});
```

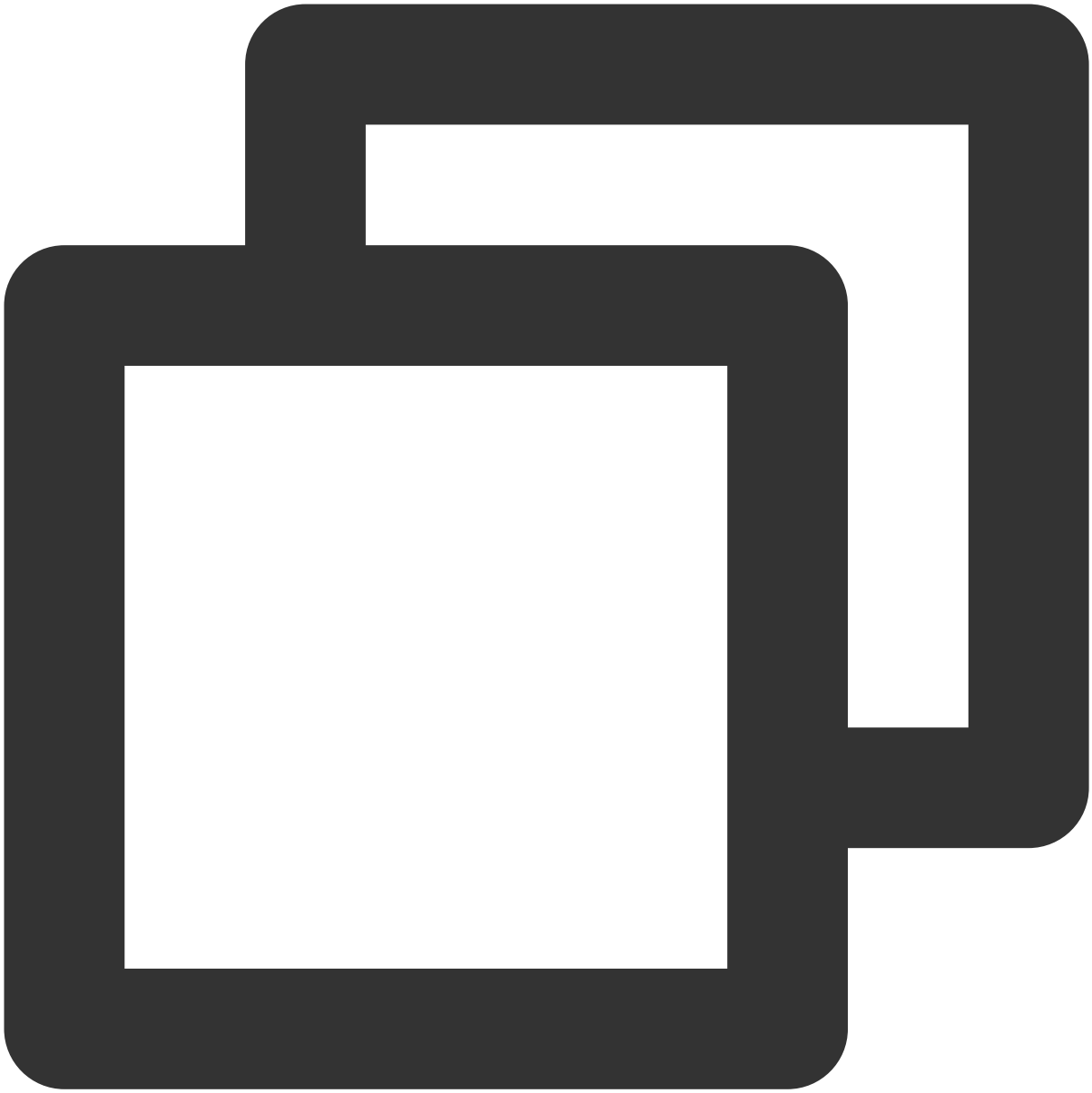
格式三（不推荐）：前端每次请求前都需要通过 getAuthorization 获取签名，后端使用固定密钥或临时密钥计算签名返回至前端。该格式分块上传权限不便控制，不推荐您使用此格式。



```
var COS = require('cos-js-sdk-v5');
var cos = new COS({
    // getAuthorization 必选参数
    getAuthorization: function (options, callback) {
        // 异步获取签名

        var url = 'http://example.com/server/sts.php'; // url 替换成您自己的后端服务
        var method = (options.Method || 'get').toLowerCase();
        var query = options.Query || {};
        var headers = options.Headers || {};
        var pathname = options.Pathname || '/';
```

```
        var xhr = new XMLHttpRequest();
        var data = {
            method: method,
            pathname: pathname,
            query: query,
            headers: headers,
        };
        xhr.open('POST', url, true);
        xhr.setRequestHeader('content-type', 'application/json');
        xhr.onload = function (e) {
            try {
                var data = JSON.parse(e.target.responseText);
            } catch (e) {
            }
            if (!data || !data.authorization) return console.error('authorization i
            callback({
                Authorization: data.authorization,
                // SecurityToken: data.sessionToken, // 如果使用临时密钥，需要把 sessio
            });
        };
        xhr.send(JSON.stringify(data));
    },
});
```

格式四（不推荐）：前端使用固定密钥计算签名，该格式适用于前端调试，若使用此格式，请避免泄露密钥。



```
var COS = require('cos-js-sdk-v5');

// SECRETID 和 SECRETKEY 请登录 https://console.tencentcloud.com/cam/capi 进行查看和管
var cos = new COS({
    SecretId: 'SECRETID',
    SecretKey: 'SECRETKEY',
});
```

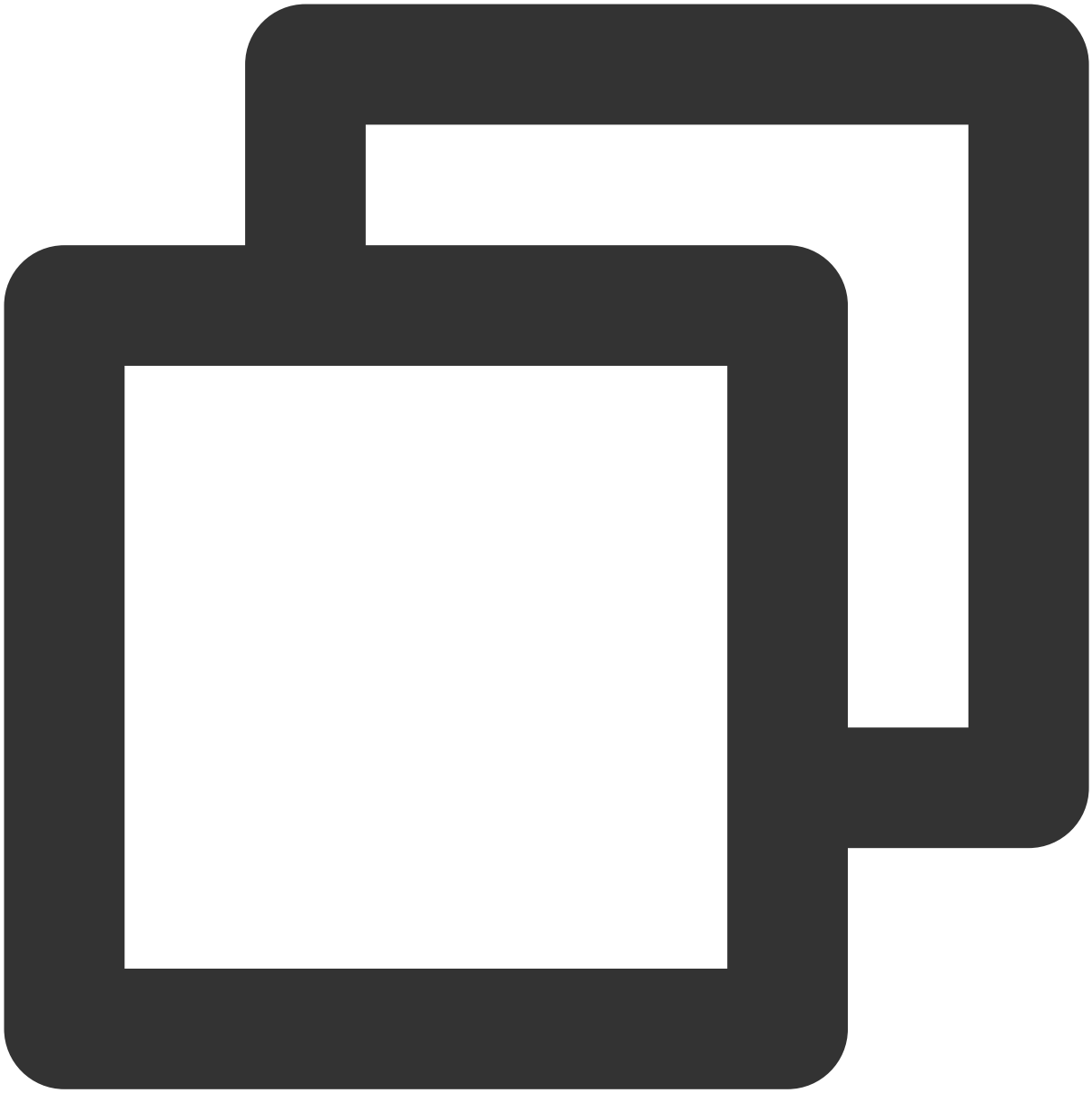
构造函数参数说明

参数名	参数描述	类型	是否必填
SecretId	用户的 SecretId	String	否
SecretKey	用户的 SecretKey，建议只在前端调试时使用，避免暴露密钥	String	否
FileParallelLimit	同一个实例下上传的文件并发数，默认值3	Number	否
ChunkParallelLimit	同一个上传文件的分块并发数，默认值3	Number	否
ChunkRetryTimes	分块上传及分块复制时，出错重试次数，默认值2（加第一次，请求共3次）	Number	否
ChunkSize	分块上传时，每片的字节数大小，默认值1048576（1MB）	Number	否
SliceSize	使用 uploadFiles 批量上传时，文件大小大于该数值将使用按分块上传，否则将调用简单上传，单位 Byte，默认值1048576（1MB）	Number	否
CopyChunkParallelLimit	进行分块复制操作中复制分块上传的并发数，默认值20	Number	否
CopyChunkSize	使用 sliceCopyFile 分块复制文件时，每片的大小字节数，默认值10485760（10MB）	Number	否
CopySliceSize	使用 sliceCopyFile 分块复制文件时，文件大小大于该数值将使用分块复制，否则将调用简单复制，默认值10485760（10MB）	Number	否
ProgressInterval	上传进度的回调方法 onProgress 的回调频率，单位 ms，默认值1000	Number	否
Protocol	发请求时用的协议，可选项 `https:`、`http:`，默认判断当前页面是 `http:` 时使用 `http:`，否则使用 `https:`	String	否
Domain	调用操作存储桶和对象的 API 时自定义请求域名。可以使用模板，如 `"{Bucket}.cos.{Region}.myqcloud.com"`，即在调用 API 时会使用参数中传入的 Bucket 和 Region 进行替换。	String	否
UploadQueueSize	上传队列最长大小，超出的任务如果状态不是 waiting、checking、uploading 会被清理，默认10000	Number	否
ForcePathStyle	强制使用后缀式模式发请求。后缀式模式中 Bucket 会放在域名后的 pathname 里，并且 Bucket 会加入签名	Boolean	否

版权所有：腾讯云计算（北京）有限责任公司

	pathname 计算，默认 false		
UploadCheckContentMd5	上传文件时校验 Content-MD5，默认 false。如果开启，上传文件时会对文件内容计算 MD5，大文件耗时较长	Boolean	否
getAuthorization	获取签名的回调方法，如果没有 SecretId、SecretKey 时，这个参数必选。 **注意: 该回调方法在初始化实例时传入，在使用实例调用接口时才会执行并获取签名。**	Function	否
Timeout	超时时间，单位毫秒，默认为0，即不设置超时时间	Number	否
UseAccelerate	是否启用全球加速域名，默认为 false。若改为 true，需要存储桶开启全球加速功能，详情请参见 开启全球加速。	Boolean	否

getAuthorization 回调函数说明的函数说明（使用格式一）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的回调参数说明：

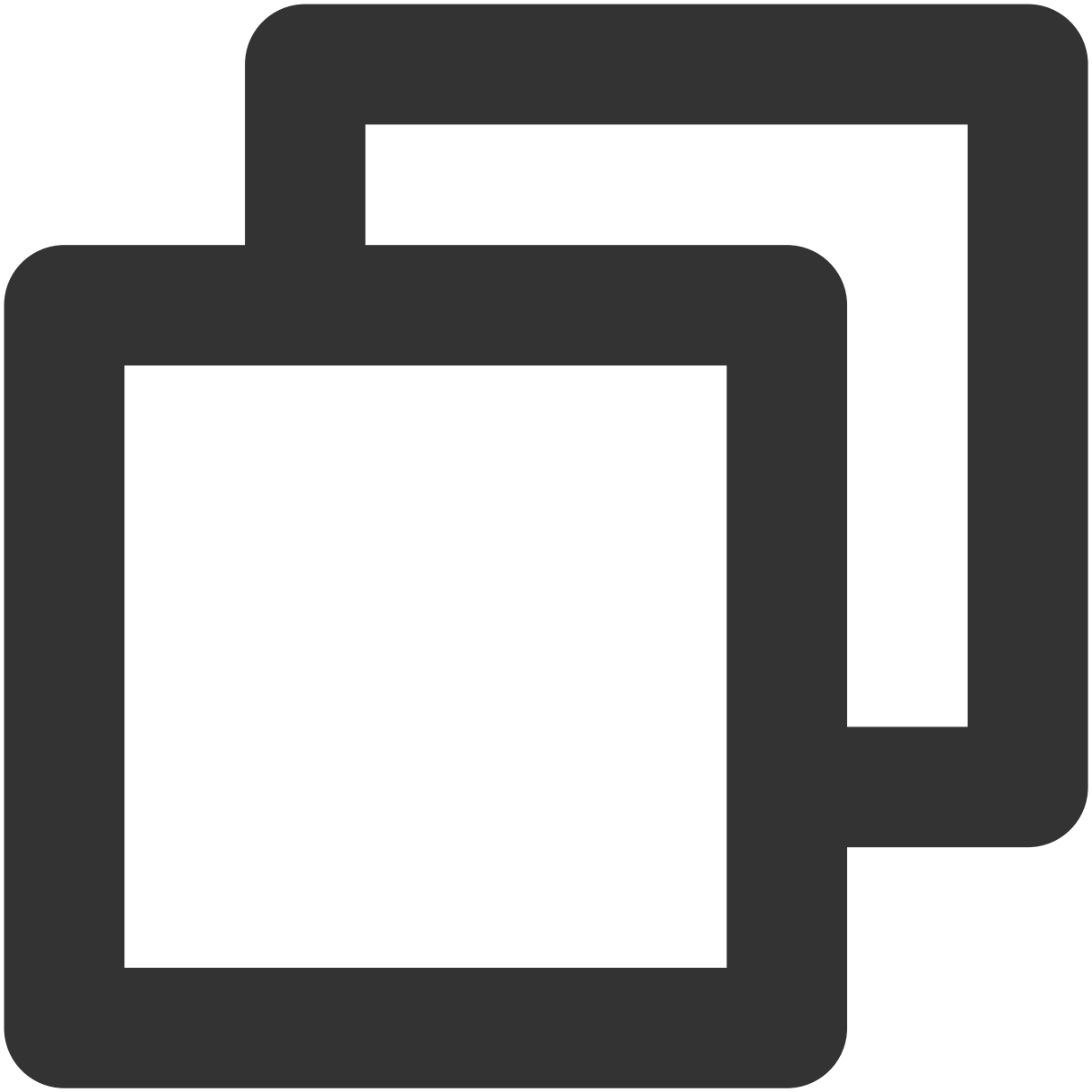
参数名	参数描述	类型
options	获取临时密钥需要的参数对象	Object
- Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String

- Region	存储桶所在地域，枚举值请参见 存储桶地域信息	String
callback	临时密钥获取完成后的回传方法	Function

获取完临时密钥后，callback 回传一个对象，回传对象的属性列表如下：

属性名	参数描述	类型	是否必填
TmpSecretId	获取回来的临时密钥的 tmpSecretId	String	是
TmpSecretKey	获取回来的临时密钥的 tmpSecretKey	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段	String	是
StartTime	密钥获取的开始时间，即获取时刻的时间戳，单位秒，startTime，如：1580000000，用于签名开始时间，传入该参数可避免前端时间偏差签名过期问题	String	否
ExpiredTime	获取回来的临时密钥的 expiredTime，超时时刻的时间戳，单位秒，如：1580000900	String	是

getAuthorization 回调函数说明（使用格式二）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的函数说明回调参数说明：

参数名	参数描述	类型
options	获取签名需要的参数对象	Object
- Method	当前请求的 Method	String
- Pathname	请求路径，用于签名计算	String

- Key	对象键（Object 的名称），对象在存储桶中的唯一标识，了解更多可参见 对象概述。 **注意:当使用实例请求的接口不是对象操作相关接口时，该参数为空。**	String
- Query	当前请求的 query 参数对象，{key: 'val'} 的格式	Object
- Headers	当前请求的 header 参数对象，{key: 'val'} 的格式	Object
callback	临时密钥获取完成后的回调	Function

getAuthorization 计算完成后，callback 回传参数支持两种格式：
格式一：回传鉴权凭证字符串 Authorization。
格式二：回传一个对象，对象属性列表如下：

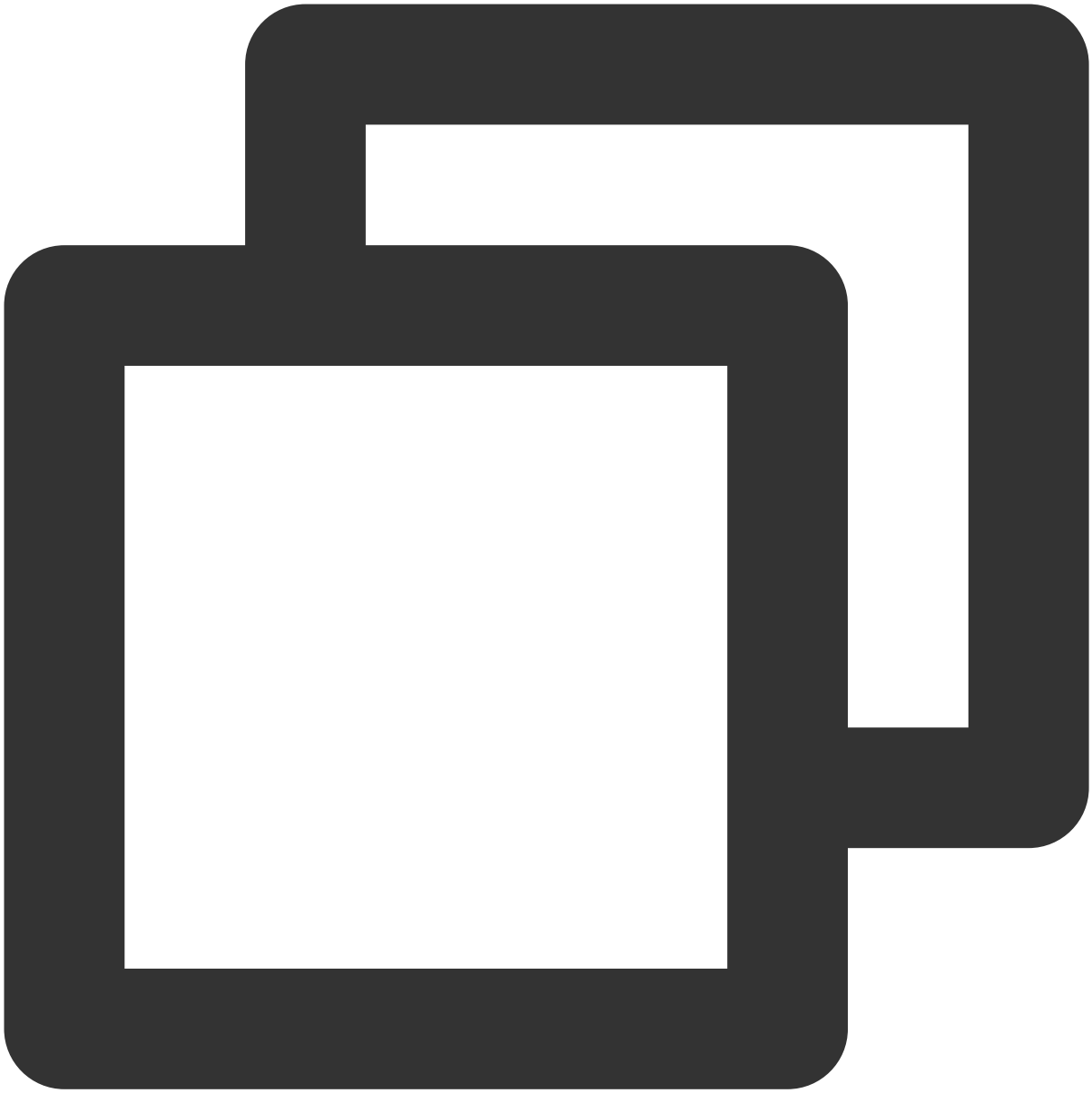
属性名	参数描述	类型	是否必填
Authorization	计算得到的签名字符串	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段	String	否

获取鉴权凭证

实例本身鉴权凭证可以通过实例化时传入的参数控制如何或获取，有三种获取方式：
1. 实例化时，传入 SecretId、SecretKey，每次需要签名都由实例内部计算。
2. 实例化时，传入 getAuthorization 回调，每次需要签名通过这个回调计算完，返回签名至实例。
3. 实例化时，传入 getAuthorization 回调，调用回调时返回临时密钥凭证，在临时密钥凭证过期后会再次调用该回调。

开启 beacon 上报

为了持续跟踪和优化 SDK 的质量，给您带来更好的使用体验，我们在 SDK 中引入了 腾讯灯塔 SDK。
说明：
腾讯灯塔只对 COS 侧的请求性能进行监控，不会上报业务侧数据。
若是想开启该功能，请先确保 SDK 版本升级到1.4.0及以上，然后在初始化中指定 EnableTracker 为 true。

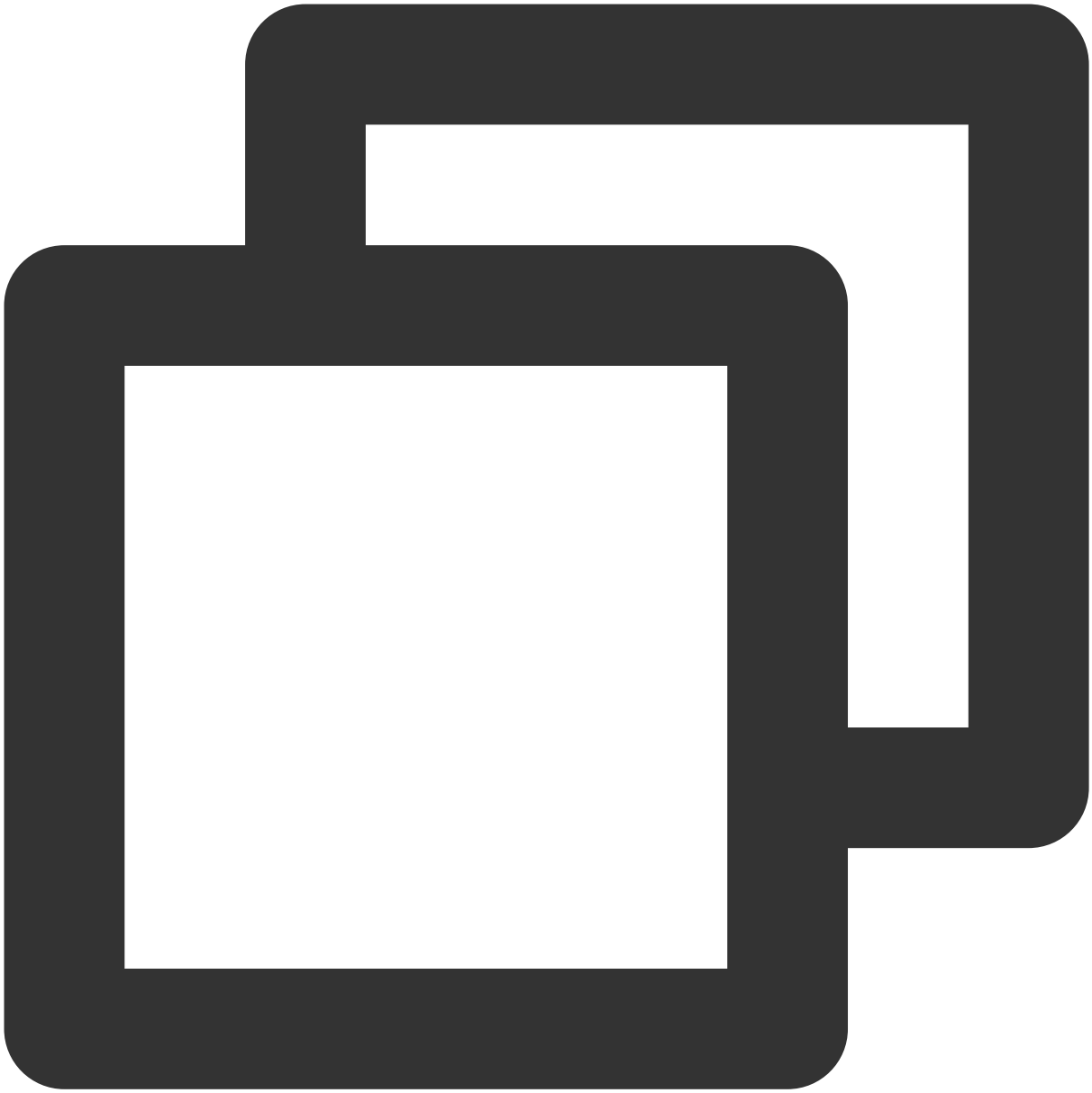


```
new COS({
  EnableTracker: true,
})
```

使用方式

回调方式

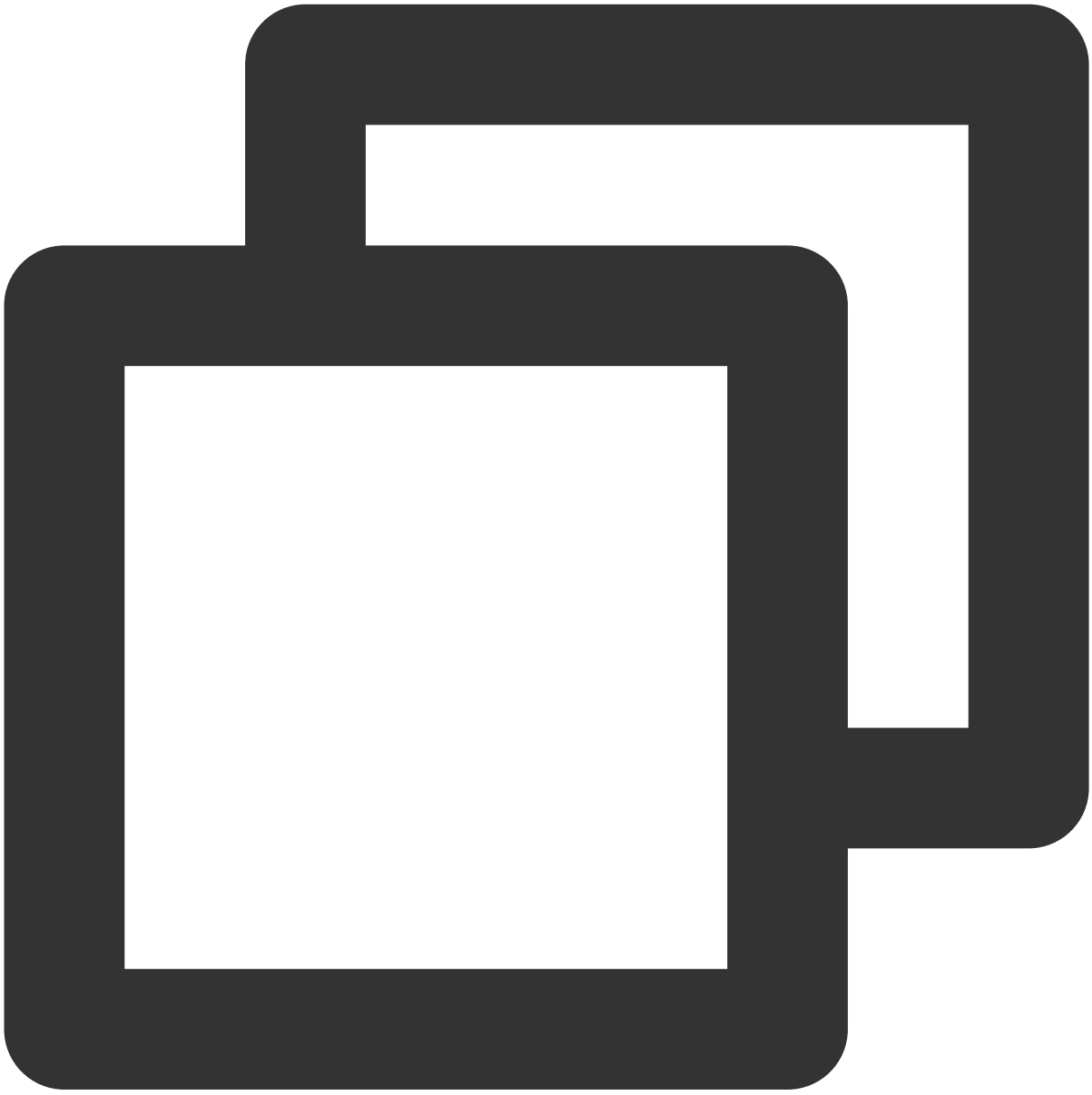
文档里默认使用回调方式，使用代码如下:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }, function(err, data) {
  if (err) {
    console.log('上传出错', err);
  } else {
    console.log('上传成功', data);
  }
});
```

Promise

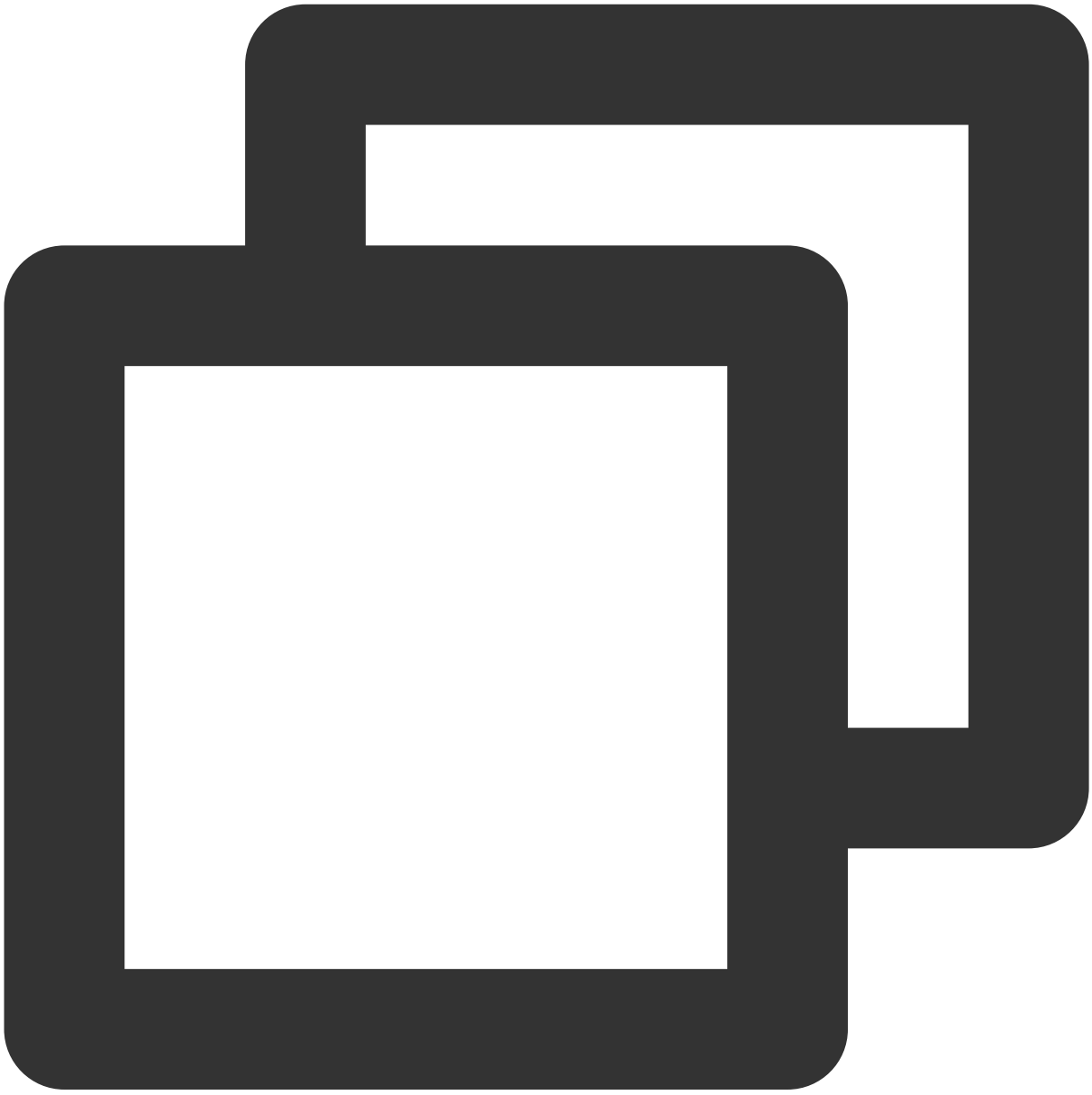
SDK 同样支持 Promise 方式调用，例如上述回调方式的代码等同于以下代码:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }).then(data => {
  console.log('上传成功', data);
}).catch(err => {
  console.log('上传出错', err);
});
```

同步方式

同步方式基于 JavaScript 的 async 和 await(使用时请注意浏览器兼容性)，上述回调方式的代码等同于以下代码:



```
async function upload() {
  // 这里省略初始化过程和上传参数
  var cos = new COS({ ... });
  try {
    var data = await cos.uploadFile({ ... });
    return { err: null, data: data }
  } catch (err) {
    return { err: err, data: null };
```

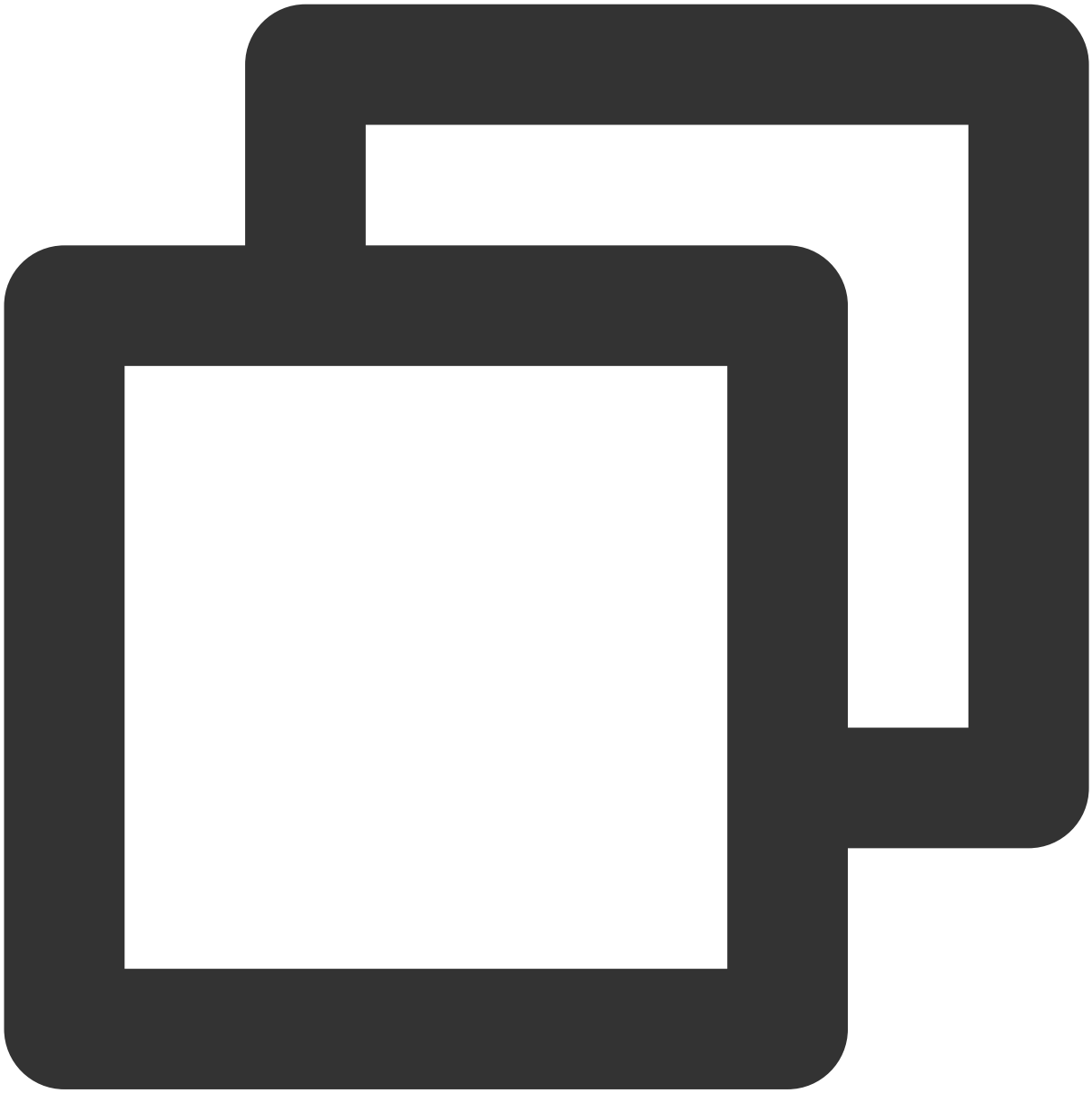
```
    }
  }
  // 可以同步拿到请求的返回值,这里举例说明,实际返回的数据格式可以自定义
  var uploadResult = await upload();
  if (uploadResult.err) {
    console.log('上传出错', uploadResult.err);
  } else {
    console.log('上传成功', uploadResult.data);
  }
```

注意：

cos.getObjectUrl 目前只支持回调方式。

使用技巧

通常情况下我们只需要创建一个 COS SDK 实例，然后在需要调用 SDK 方法的地方直接使用这个实例即可，示例代码如下：

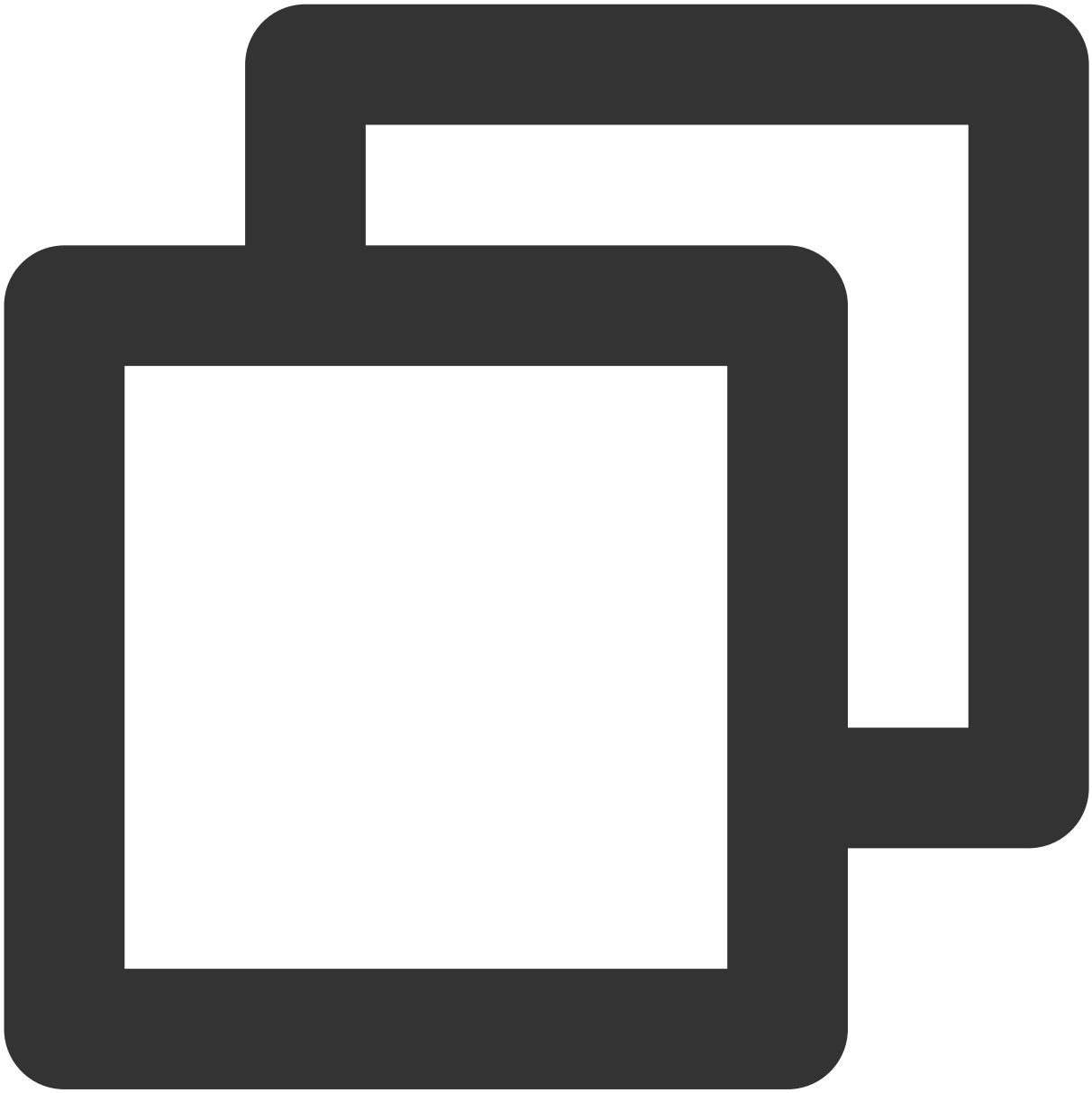


```
/* vue 项目举例 */

/* 新建一个 cos.js, 导出 cos 实例 */
import COS from 'cos-js-sdk-v5'; // 通过 npm 安装的 SDK
const cos = new COS({
  ....
});
export default cos;

/* 单页面里 page.vue */
<template>
```

```
  <input id="fileSelector" type="file" @change="upload" />
</template>
<script>
  /* 引入上方新建的 cos.js路径 */
  import cos from 'cos';
  export default {
    data() {},
    methods: {
      upload(e) {
        const file = e.target.files && e.target.files[0];
        /* 直接调用 cos sdk 的方法 */
        cos.uploadFile({
          Bucket: 'examplebucket-1250000000', /* 填写自己的 bucket，必须字段 */
          Region: 'COS_REGION',      /* 存储桶所在地域，必须字段 */
          Key: '1.jpg',              /* 存储在桶里的对象键（例如1.jpg，a/b/test.txt），必
          Body: file, // 上传文件对象
          SliceSize: 1024 * 1024 * 5,      /* 触发分块上传的阈值，超过5MB 使用分块上传，小于
        });
      }
    },
  }
</script>
```

以下是部分常用接口例子，更详细的初始化方法请参见 demo 示例。

查询对象列表



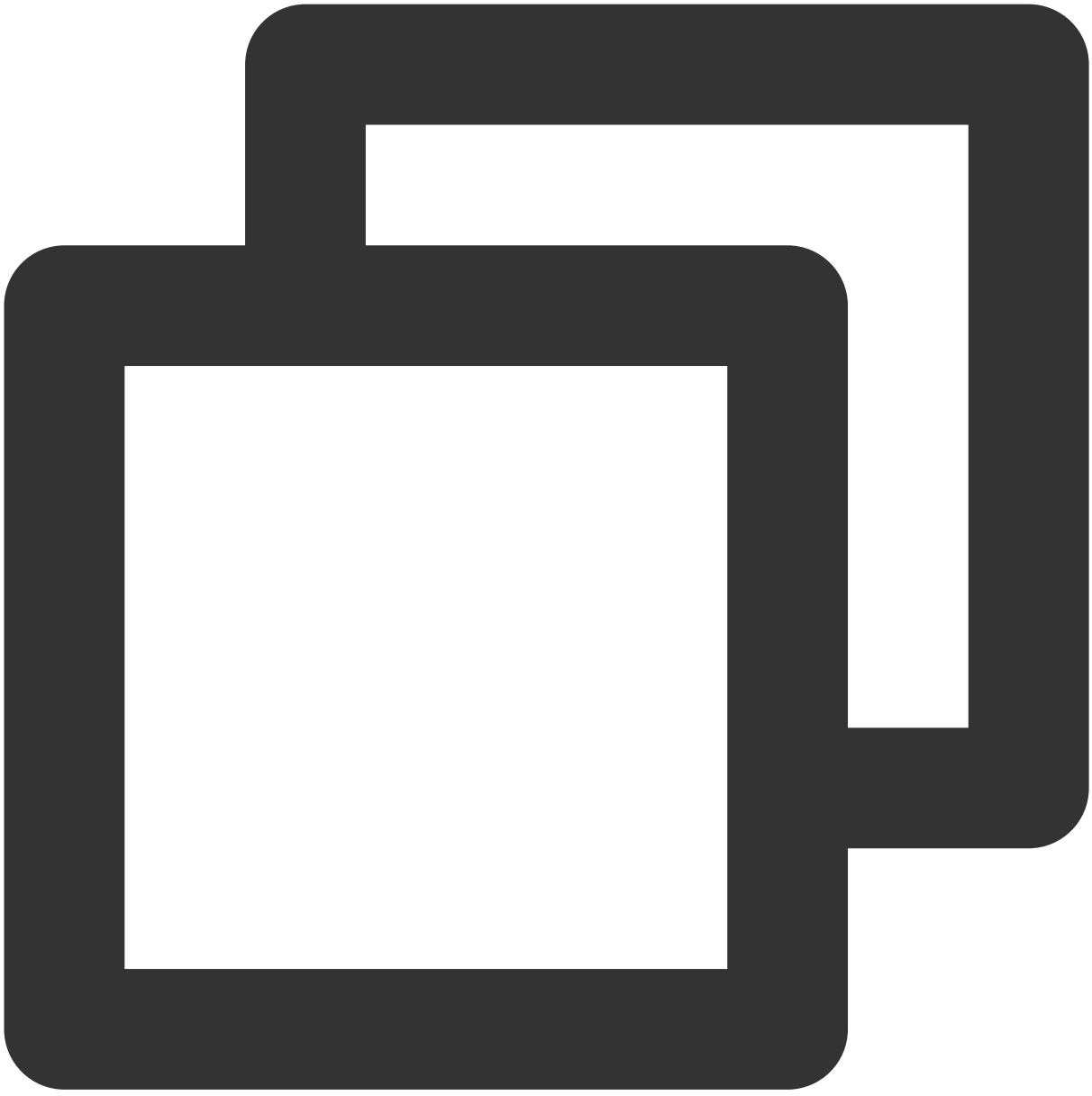
```
cos.getBucket({
    Bucket: 'examplebucket-1250000000', /* 填写自己的 bucket，必须字段 */
    Region: 'COS_REGION',      /* 存储桶所在地域，必须字段 */
    Prefix: 'a/',              /* 列出目录 a 下所有文件，非必须 */
}, function(err, data) {
    console.log(err || data.Contents);
});
```

上传对象

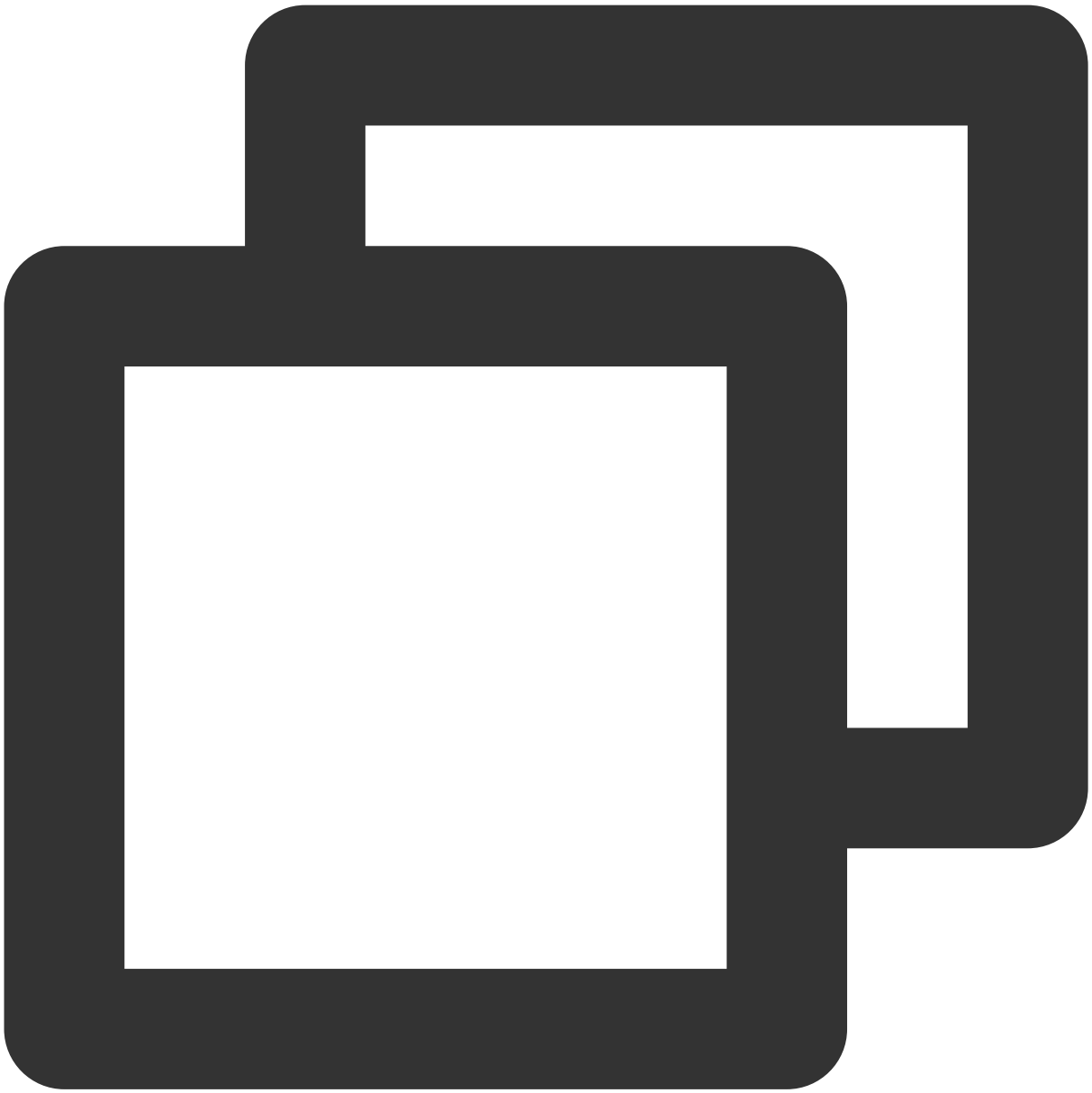
强烈推荐使用高级上传接口 uploadFile，自动针对小文件使用简单上传，大文件使用分块上传，性能更好。详情请参见 高级上传 文档。

若使用临时密钥方式，需同时授权 `简单上传对象` 和 `分块上传` 的权限。请参考授权指引。

常见上传错误排查，请参考 常见问题。



```
<!-- html 页面 DOM 元素 -->

<!-- 选择要上传的文件 -->
<input id="fileSelector" type="file" />
<!-- 点击按钮上传 -->
<input id="submitBtn" type="submit" />
```

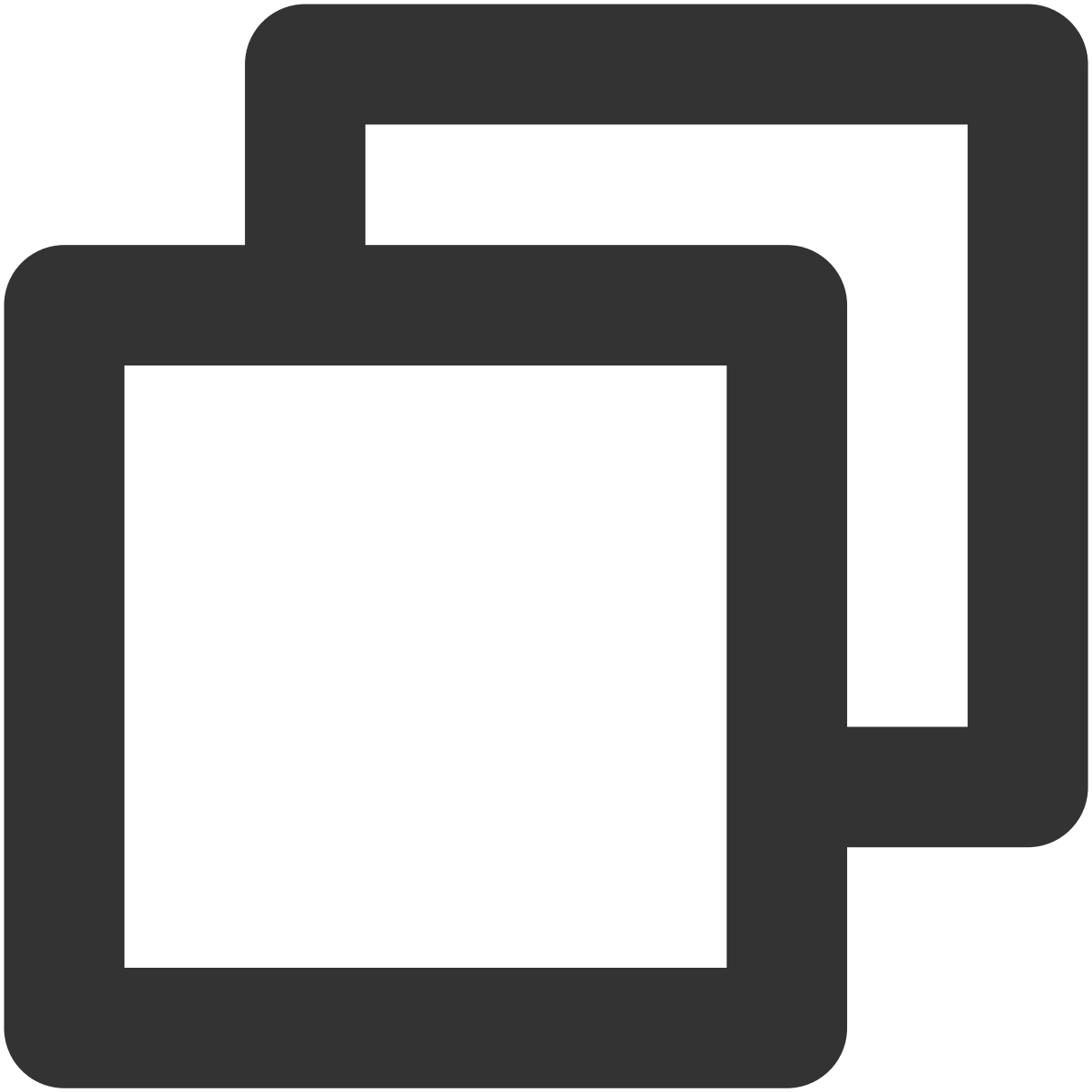


```
function handleFileInUploading(file) {
  cos.uploadFile({
      Bucket: 'examplebucket-1250000000', /* 填写自己的 bucket，必须字段 */
      Region: 'COS_REGION',      /* 存储桶所在地域，必须字段 */
      Key: '1.jpg',              /* 存储在桶里的对象键（例如:1.jpg，a/b/test.txt，图片.jp
      Body: file, // 上传文件对象
      SliceSize: 1024 * 1024 * 5,      /* 触发分块上传的阈值，超过5MB使用分块上传，小于5MB使
      onProgress: function(progressData) {
          console.log(JSON.stringify(progressData));
      }
```

```
  }, function(err, data) {
      if (err) {
        console.log('上传失败', err);
      } else {
        console.log('上传成功');
      }
  });
}

/* 选择文件 */
document.getElementById('submitBtn').onclick = function (e) {
  var file = document.getElementById('fileSelector').files[0];
  if (!file) {
    document.getElementById('msg').innerText = '未选择上传文件';
    return;
  }
  handleFileInUploading(file);
};
```

下载对象

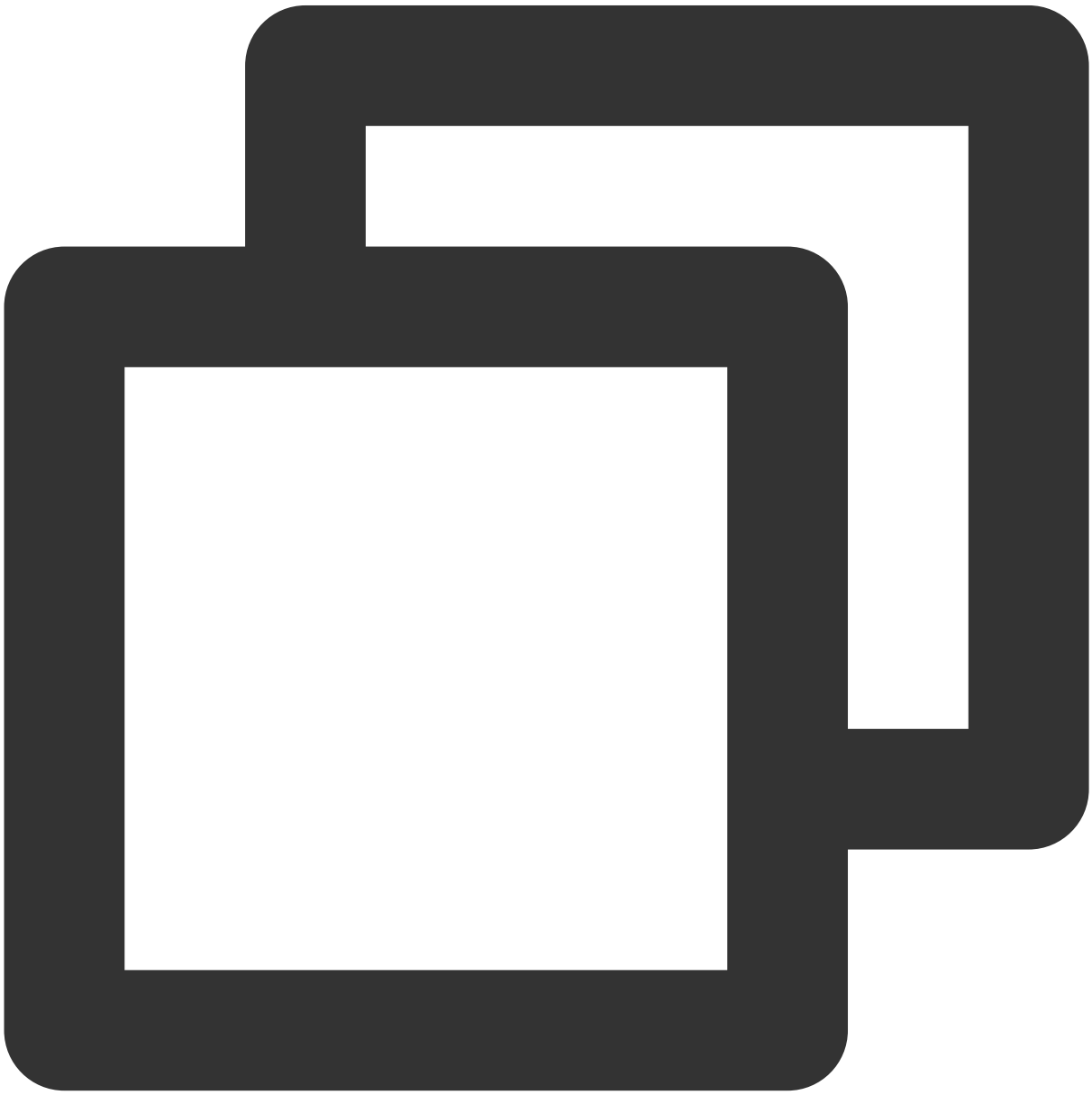
浏览器里通过生成预签名 url 并触发浏览器下载的方式实现



```
cos.getObjectUrl({
    Bucket: 'examplebucket-1250000000', /* 填写自己的 bucket，必须字段 */
    Region: 'COS_REGION',     /* 存储桶所在地域，必须字段 */
    Key: '1.jpg',              /* 存储在桶里的对象键（例如1.jpg，a/b/test.txt），必须字段
}, function(err, data) {
    if (err) return console.log(err);
     /* 通过指定 response-content-disposition=attachment 实现强制下载 */
    var downloadUrl = data.Url + (data.Url.indexOf('?') > -1 ? '&' : '?') + 'respon
    /* 可拼接 filename 来实现下载时重命名 */
    /* downloadUrl += ';filename=myname'; */
    // （推荐使用 window.open()方式）这里是新窗口打开 url，如果需要在当前窗口打开，可以使用隐藏
```

```
    window.open(downloadUrl);
});
```

删除对象



```
cos.deleteObject({
    Bucket: 'examplebucket-1250000000', /* 填写自己的 bucket，必须字段 */
    Region: 'COS_REGION',      /* 存储桶所在地域，必须字段 */
    Key: '1.jpg',              /* 存储在桶里的对象键（例如1.jpg，a/b/test.txt），必须字段
}, function(err, data) {
```

```
    console.log(err || data);
});
```

图片处理
基础图片处理

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于基础图片处理的相关的 API 概览以及 SDK 示例代码。

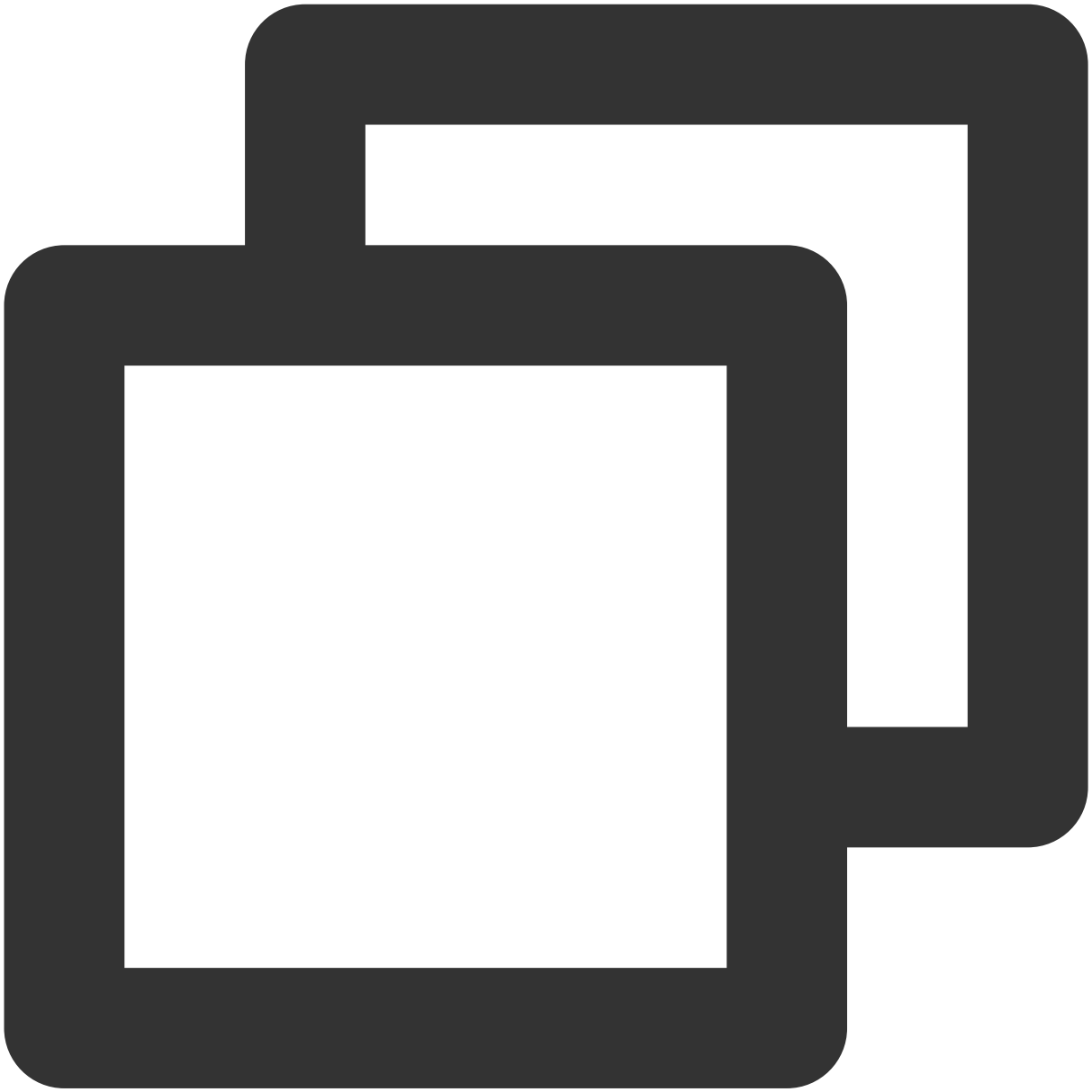
服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

上传时使用图片处理

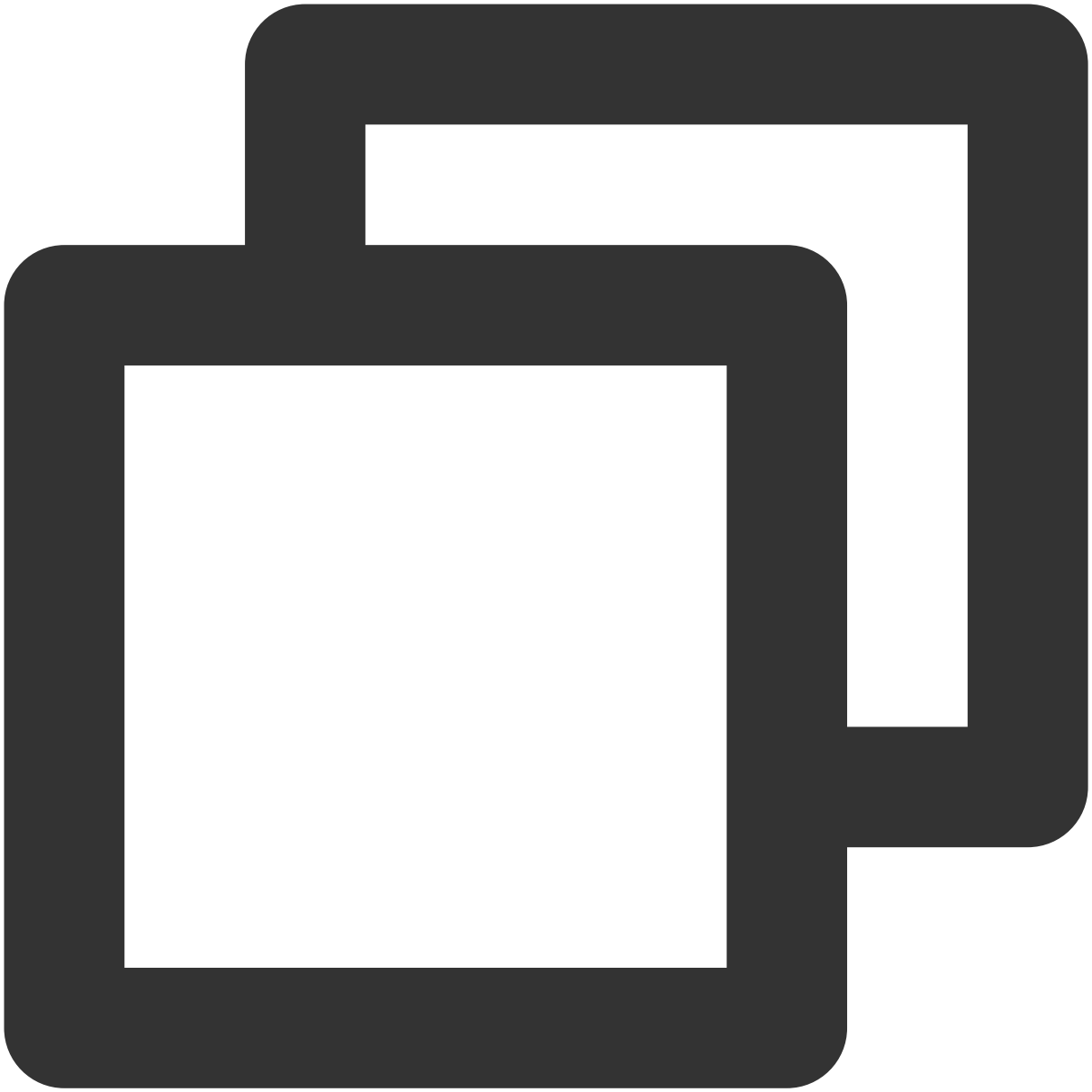
下面示例展示了如何在上传图片时自动实现图片处理。
图片上传完成后，COS 会存储原始图片和已处理过的图片。后续用户可以通过普通的下载请求获取处理结果。

示例代码

版权所有：腾讯云计算（北京）有限责任公司



```
<!-- html 页面 DOM 元素 -->

<!-- 选择要上传的文件 -->
<input id="fileSelector" type="file" />
<!-- 点击按钮上传 -->
<input id="submitBtn" type="submit" />
```



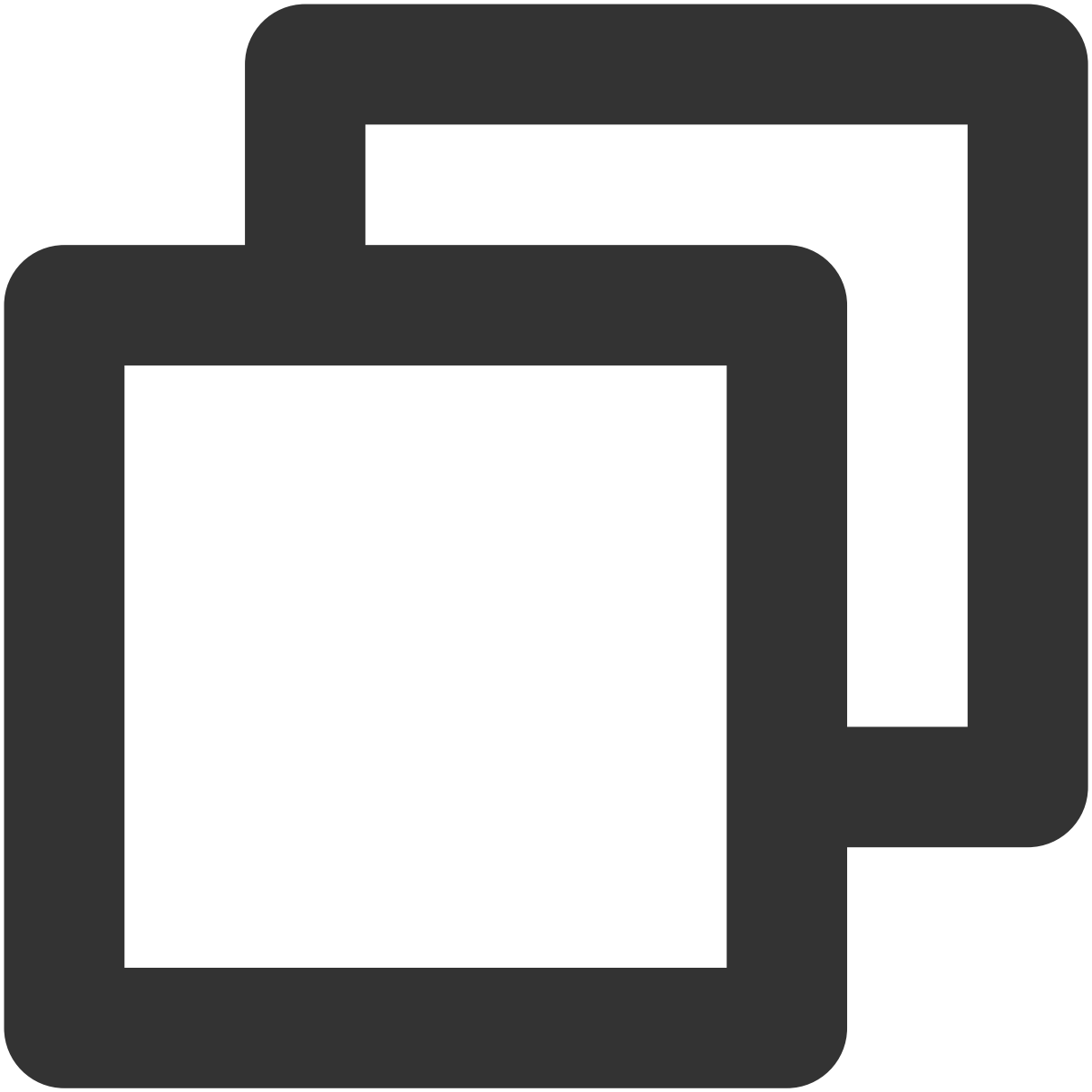
```
function handleFileInUploading(file) {
  cos.putObject(
    {
      Bucket: 'examplebucket-1250000000',
      Region: 'COS_REGION',
      Key: file.name,
      Body: file,
      Headers: {
        // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
        'Pic-Operations':
          '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jpg", "rule": "i
```

```
        },
      },
      function (err, data) {
        console.log(err || data);
      },
    );
}

document.getElementById('submitBtn').onclick = function (e) {
  var file = document.getElementById('fileSelector').files[0];
  if (!file) {
    document.getElementById('msg').innerText = '未选择上传文件';
    return;
  }
  handleFileInUploading(file);
};
```

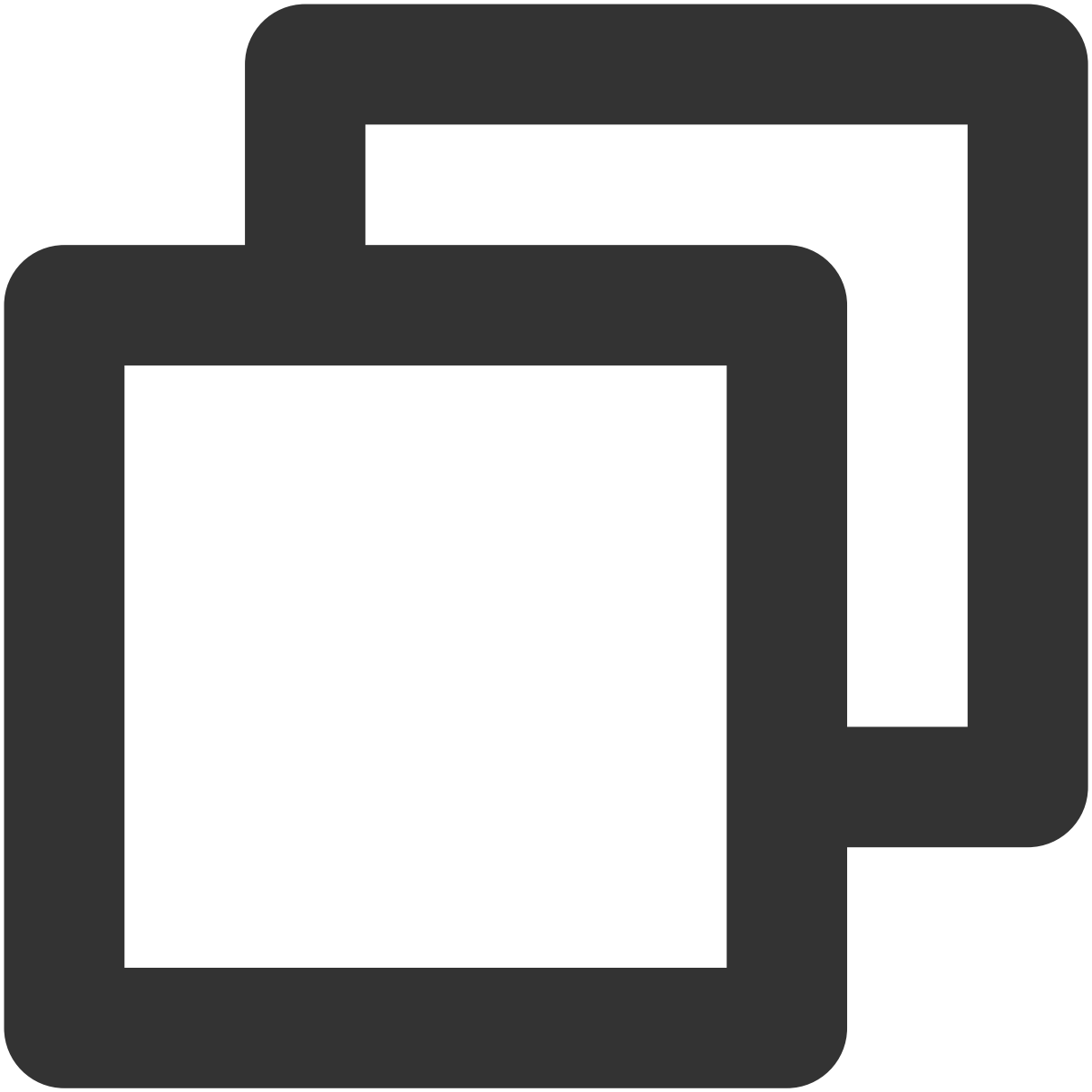
对云上数据进行图片处理

下面示例展示了如何在对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

示例代码



```
<!-- html 页面 DOM 元素 -->

<!-- 点击按钮请求对云上数据进行图片处理 -->
<input id="submitBtn" type="submit" />
```



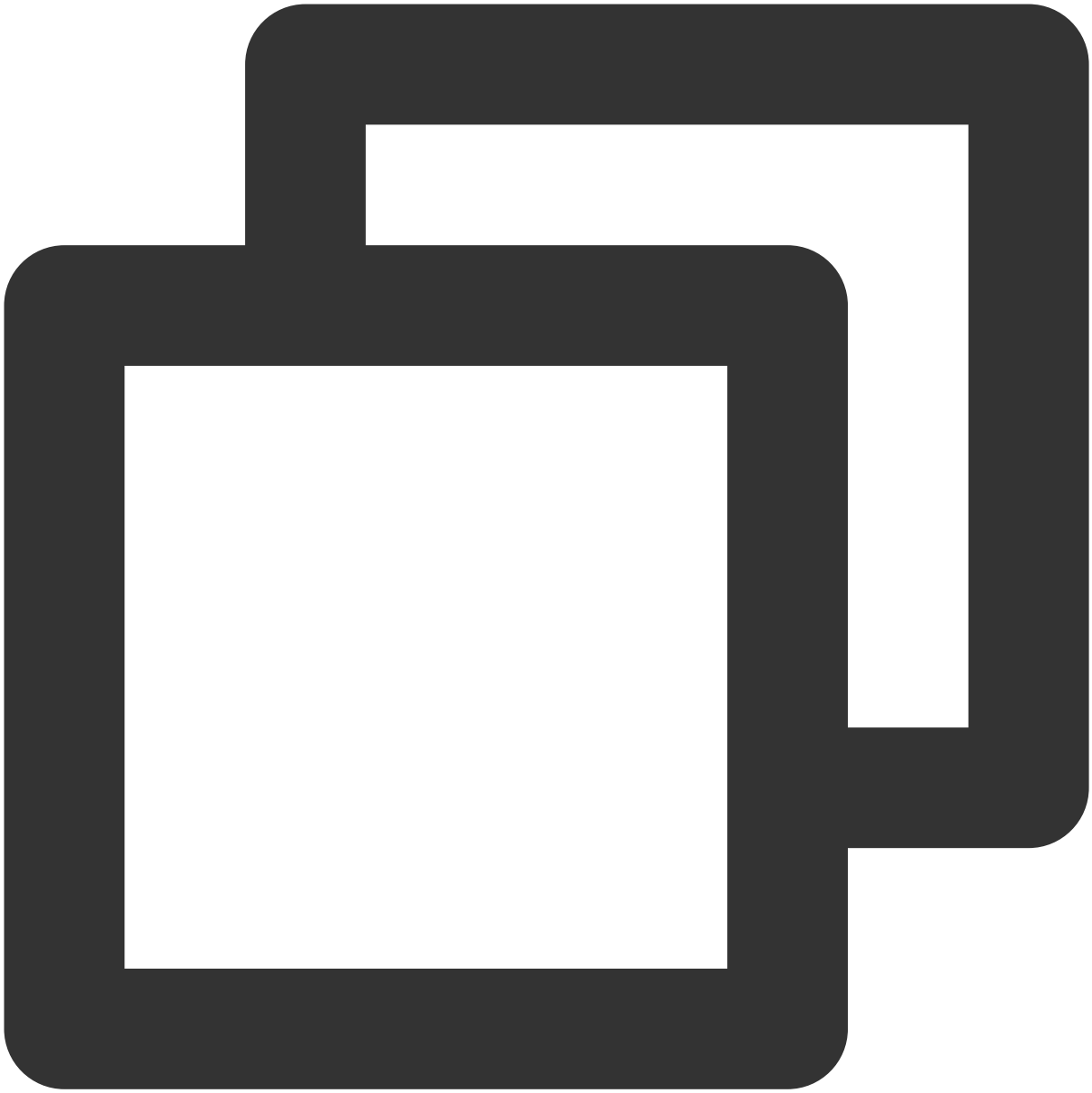
```
function handleFileInBucket() {
  cos.request(
    {
      Bucket: 'examplebucket-1250000000',
      Region: 'COS_REGION',
      Key: 'exampleImage',
      Method: 'POST',
      Action: 'image_process',
      Headers: {
        // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
        'Pic-Operations':
```

```
            '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jpg", "rule": "i
        },
      },
      function (err, data) {
        console.log(err || data);
      },
    );
}

document.getElementById('submitBtn').onclick = function (e) {
  handleFileInBucket();
};
```

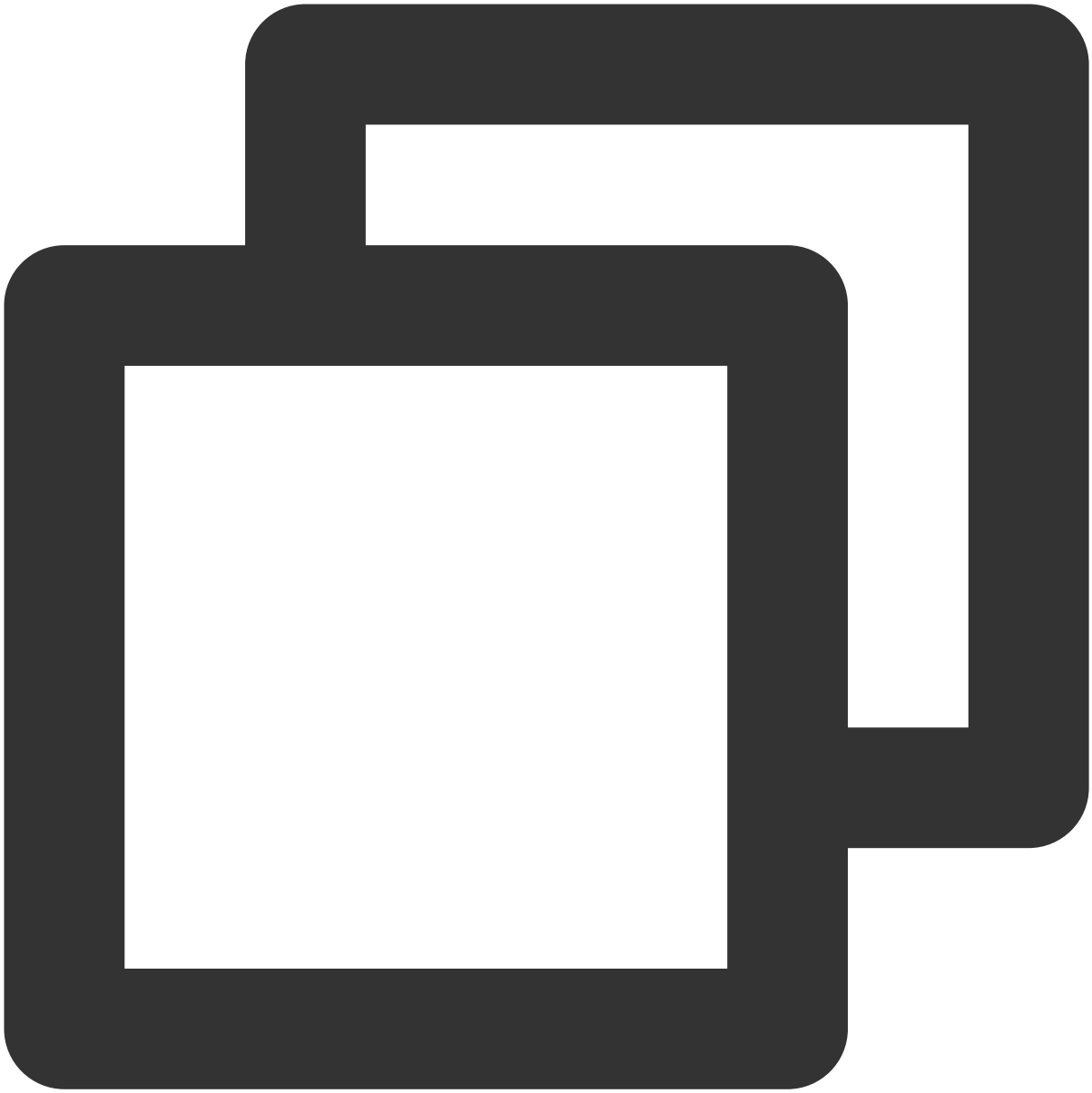
下载时使用图片处理

下面示例展示了如何在下载图片时实现图片处理。

示例代码



```
<!-- html 页面 DOM 元素 -->

<!-- 点击按钮下载文件并在下载时使用图片处理 -->
<input id="submitBtn" type="submit" />
```

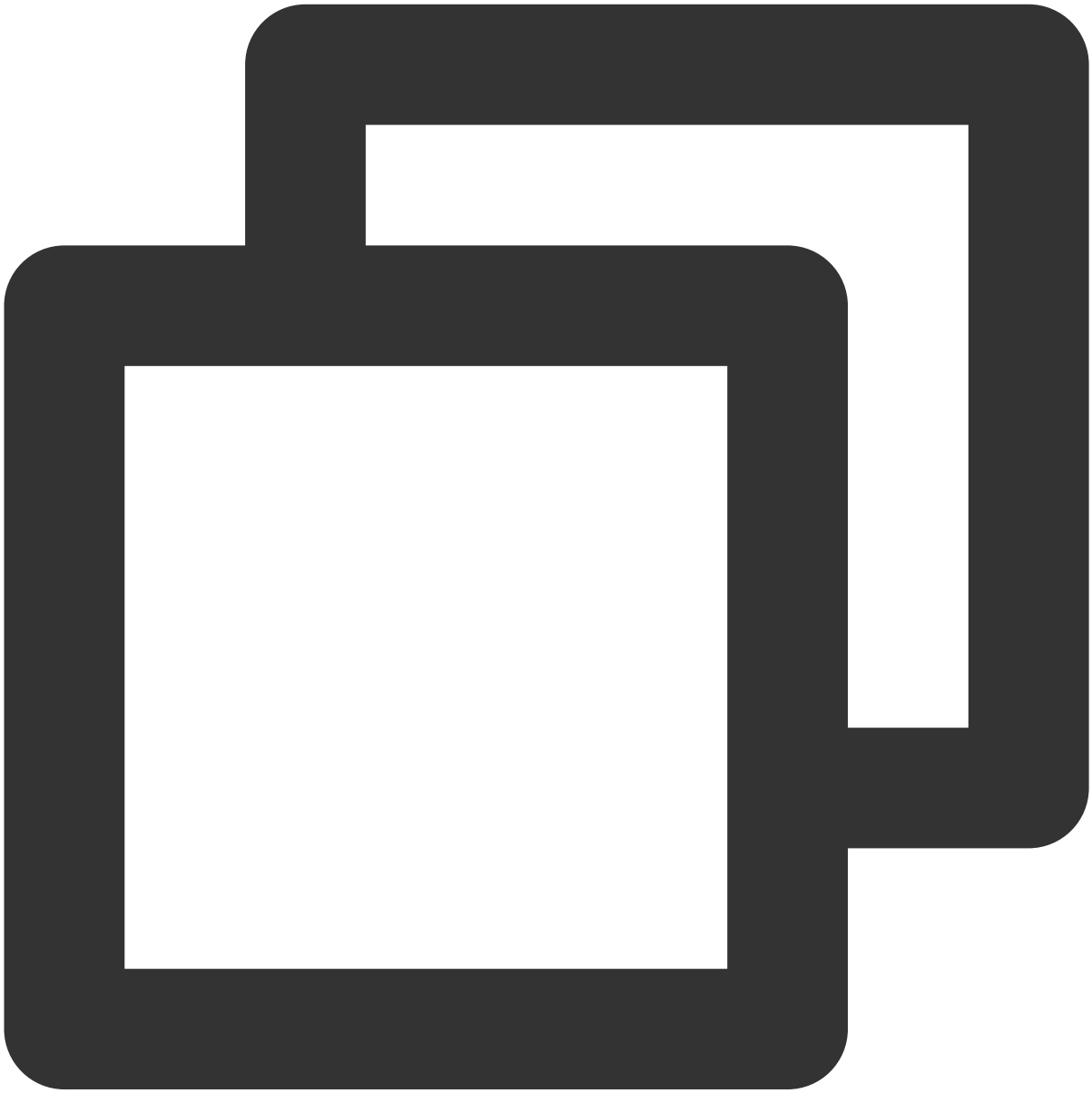


```
function getObject() {
  cos.getObject(
    {
      Bucket: 'examplebucket-1250000000',
      Region: 'COS_REGION',
      Key: 'exampleobject',
      QueryString: `imageMogr2/thumbnail/200x/`,
    },
    function (err, data) {
      console.log(err || data);
    },
```

```
    );
}
```

生成带图片处理参数的签名 URL

示例代码



```
// 生成带图片处理参数的文件签名 URL，过期时间设置为 30 分钟。
cos.getObjectUrl(
```

```
  {
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Query: {
        `imageMogr2/thumbnail/200x/`: ''
    },
    Expires: 1800,
    Sign: true,
  },
  function (err, data) {
    if (data) {
      console.log(data.URL);
    }
  },
);

// 生成带图片处理参数的文件 URL，不带签名。
cos.getObjectUrl(
  {
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Query: {
      `imageMogr2/thumbnail/200x/`: ''
    },
    Sign: false,
  },
  function (err, data) {
    if (data) {
      console.log(data.URL);
    }
  },
);
```

盲水印

最近更新时间：2024-03-13 10:07:10

简介

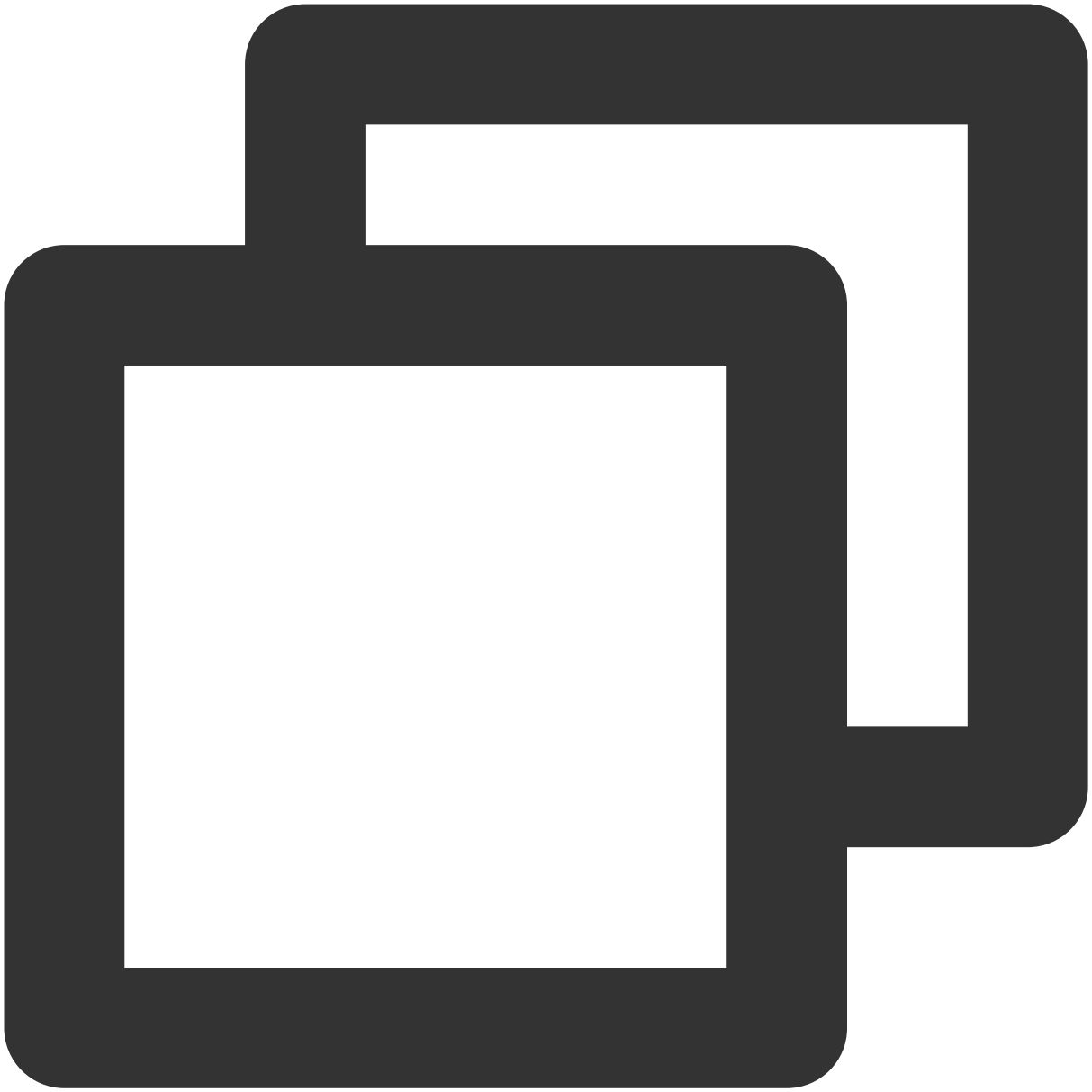
本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。
关于盲水印 API 文档请参见 盲水印。

添加盲水印

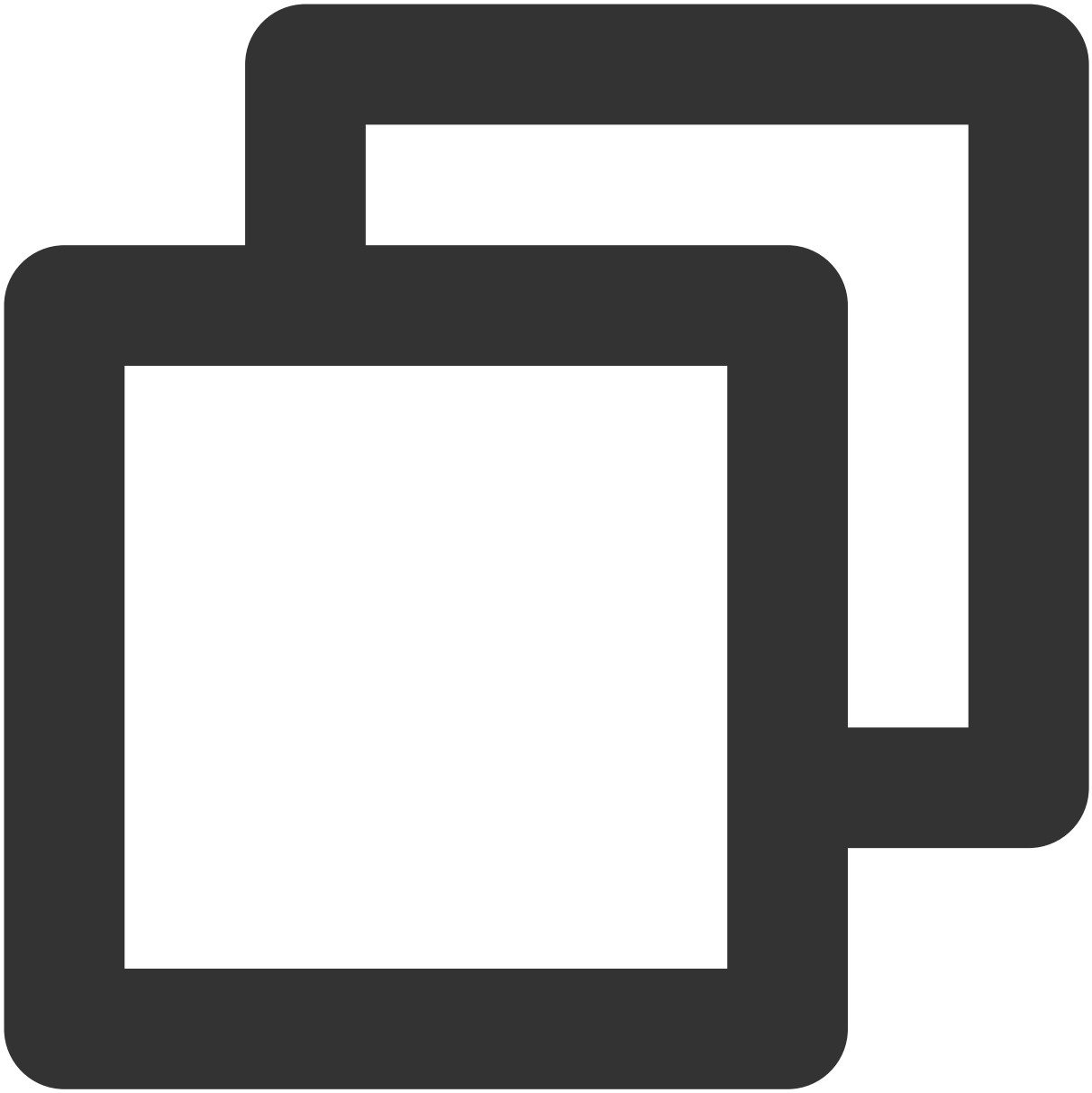
功能说明

盲水印支持在上传时添加以及下载时添加。

示例代码一：上传时添加盲水印



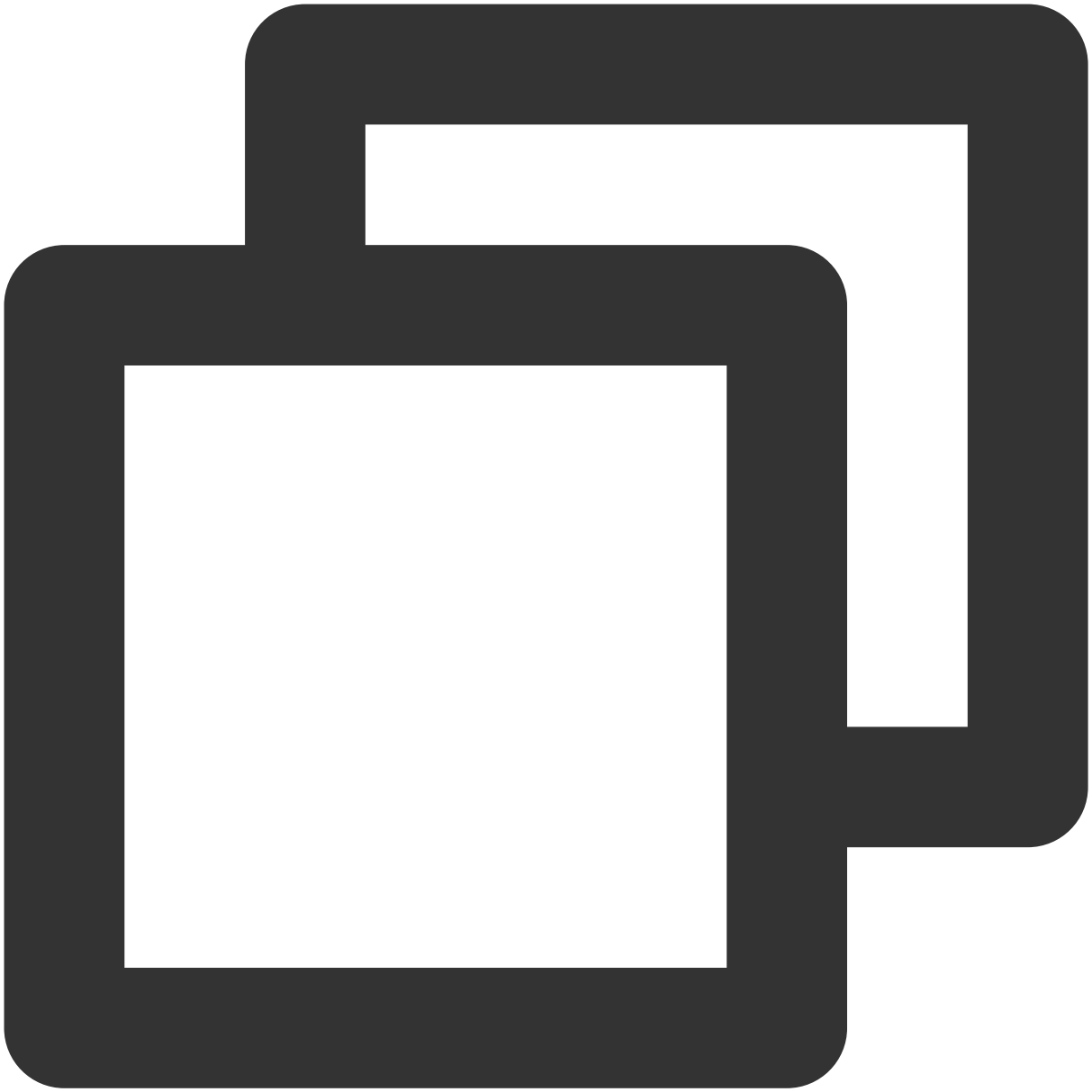
```
<!-- html 页面 DOM 元素 -->

<!-- 选择要上传的文件 -->
<input id="fileSelector" type="file" />
<!-- 点击按钮上传 -->
<input id="submitBtn" type="submit" />
```



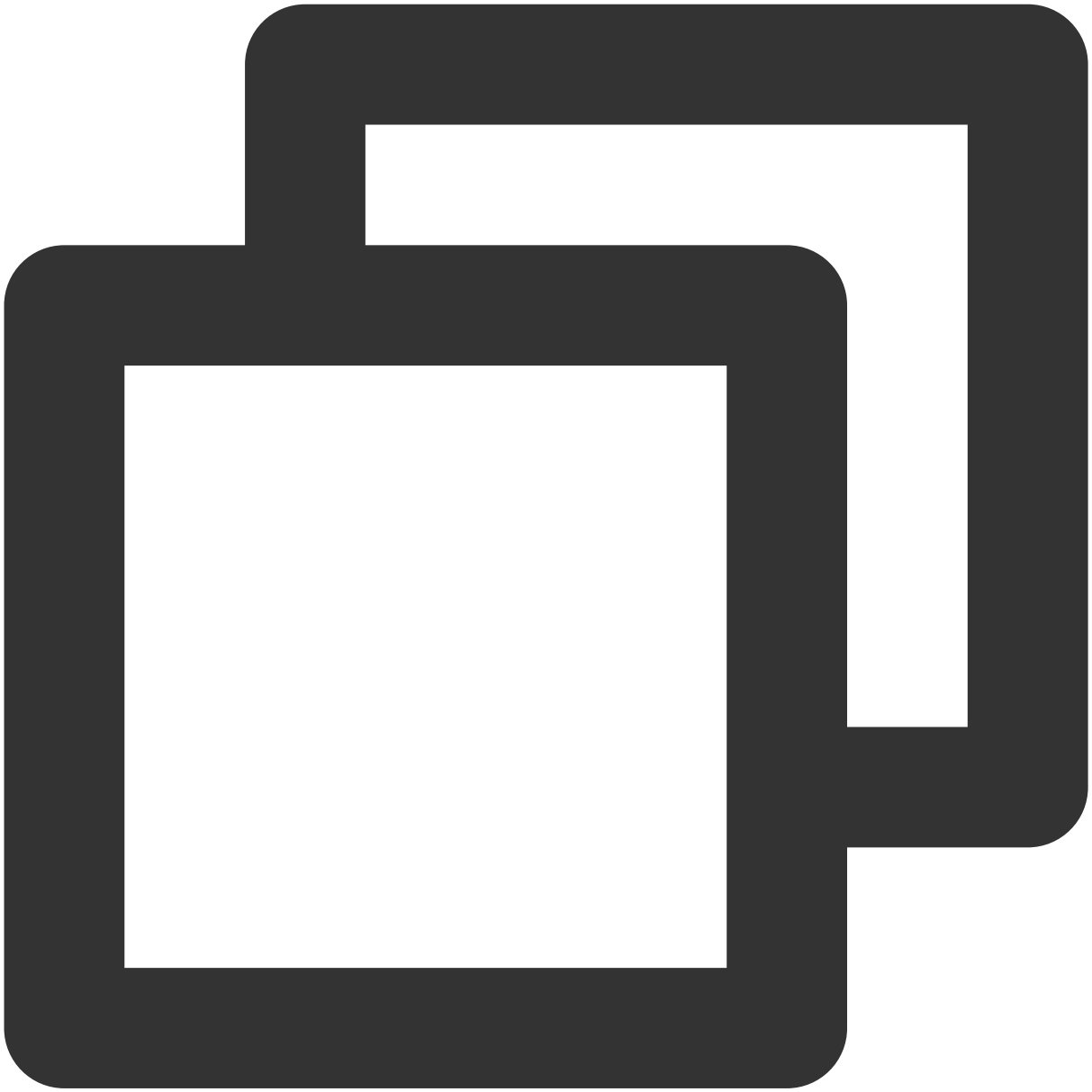
```
function handleFileInUploading(file) {
  cos.putObject(
    {
      Bucket: 'examplebucket-1250000000',
      Region: 'COS_REGION',
      Key: file.name,
      Body: file,
      Headers: {
        // 万象持久化接口，上传时持久化
        'Pic-Operations':
          '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jpg", "rule": "w
```

```
        },
      },
      function (err, data) {
        console.log(err || data);
      },
    );
}

document.getElementById('submitBtn').onclick = function (e) {
  var file = document.getElementById('fileSelector').files[0];
  if (!file) {
    document.getElementById('msg').innerText = '未选择上传文件';
    return;
  }
  handleFileInUploading();
};
```

示例代码二：下载时添加盲水印



```
<!-- html 页面 DOM 元素 -->

<!-- 点击按钮下载文件并在下载时使用图片处理 -->
<input id="submitBtn" type="submit" />
```



```
function getObject() {
  cos.getObject(
    {
      Bucket: 'examplebucket-1250000000',
      Region: 'COS_REGION',
      Key: 'exampleobject',
      QueryString: `watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucG
    },
    function (err, data) {
      console.log(err || data);
    },
```

```
    );
}
```

版权所有：腾讯云计算（北京）有限责任公司

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于媒体 bucket 接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

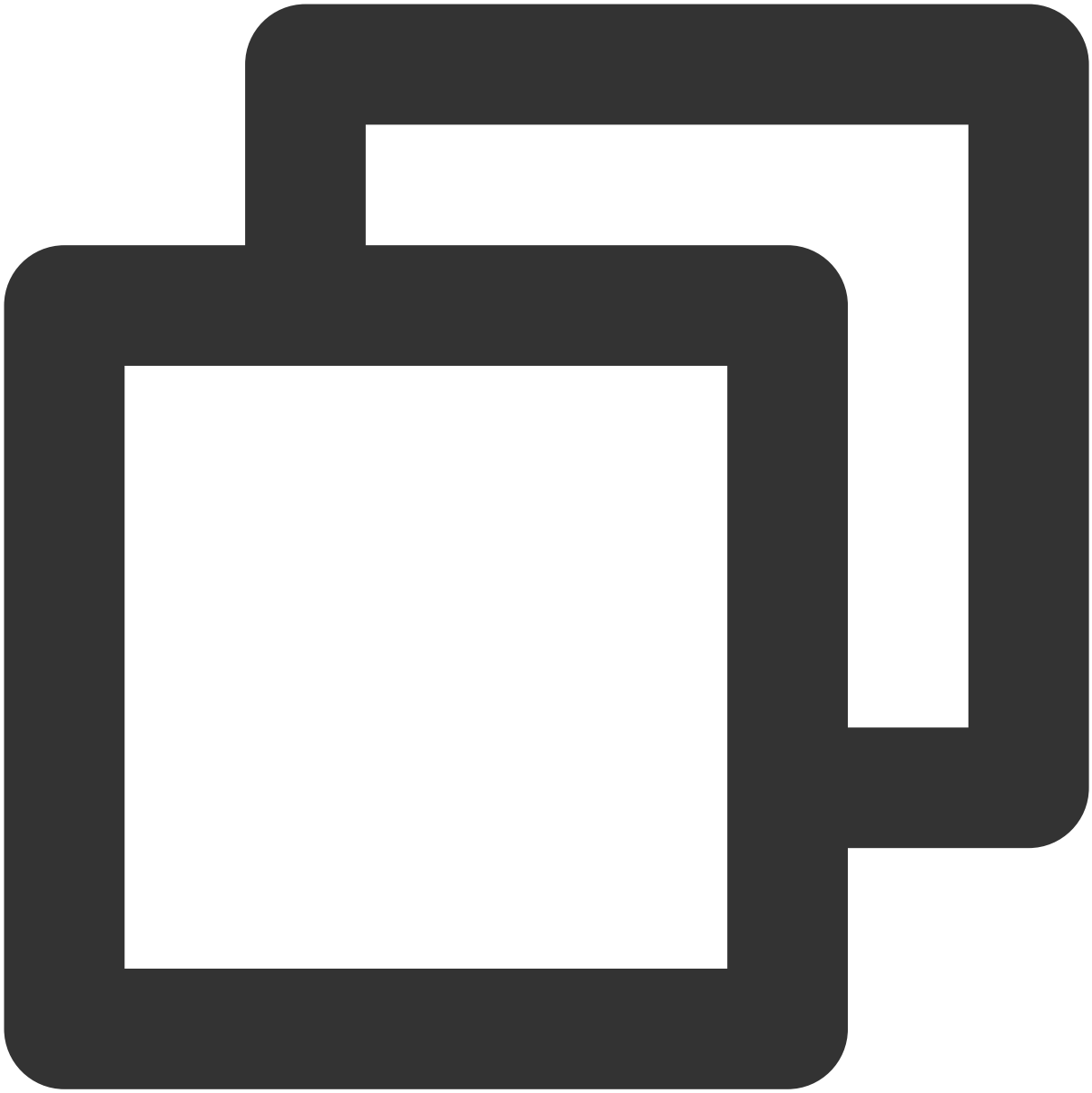
功能说明

用于查询已经开通媒体处理功能的存储桶。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

使用示例



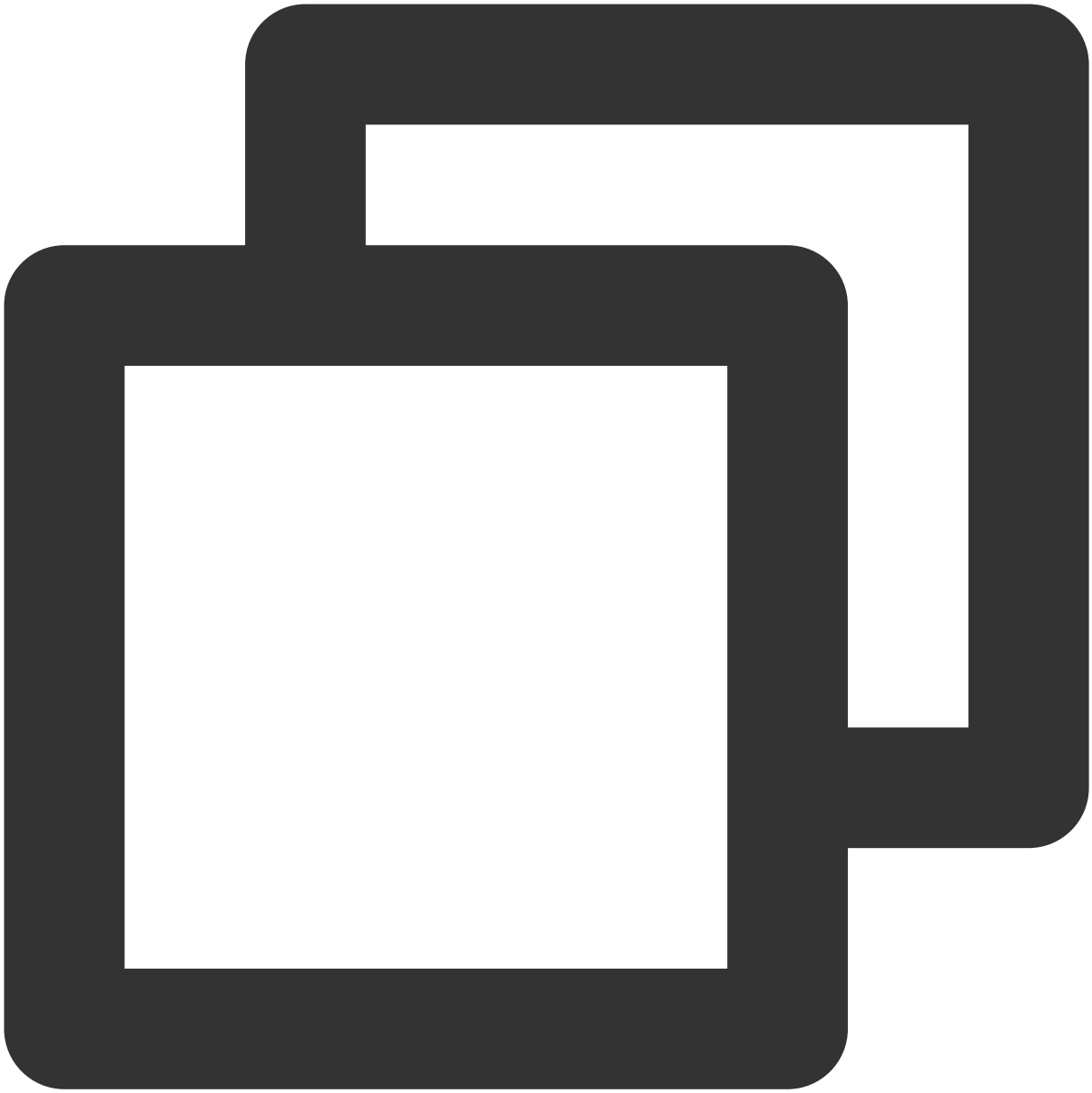
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
var host = 'ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/mediabucket';
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
```

```
    Key: 'mediabucket', /** 固定值，必须 */
    Url: url,
    Query: {
        pageNumber: '1', /** 第几页，非必须 */
        pageSize: '10', /** 每页个数，非必须 */
        // regions: 'ap-chengdu', /** 地域信息，例如'ap-beijing'，支持多个值用逗号分隔如'a
        // bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-125000000
        // bucketName: 'test', /** 存储桶名称前缀，前缀搜索，例如'test'，支持多个值用逗号分
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
PageNumber	第几页	String	否
PageSize	每页个数	String	否
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	String	否
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	String	否
BucketName	存储桶名称前缀，前缀搜索	String	否

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- Response	响应结果	Object
- - RequestId	请求的唯一 ID	String
- - TotalCount	媒体 Bucket 总数	String
- - PageNumber	当前页数，同请求中的 pageNumber	String
- - PageSize	每页个数，同请求中的 pageSize	String
- - MediaBucketList	媒体 Bucket 列表	Object
- - - BucketId	存储桶 ID	String
- - - Name	存储桶名称，同 BucketId	String
- - - Region	所在的地域	String
- - - CreateTime	创建时间	String

媒体信息接口

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

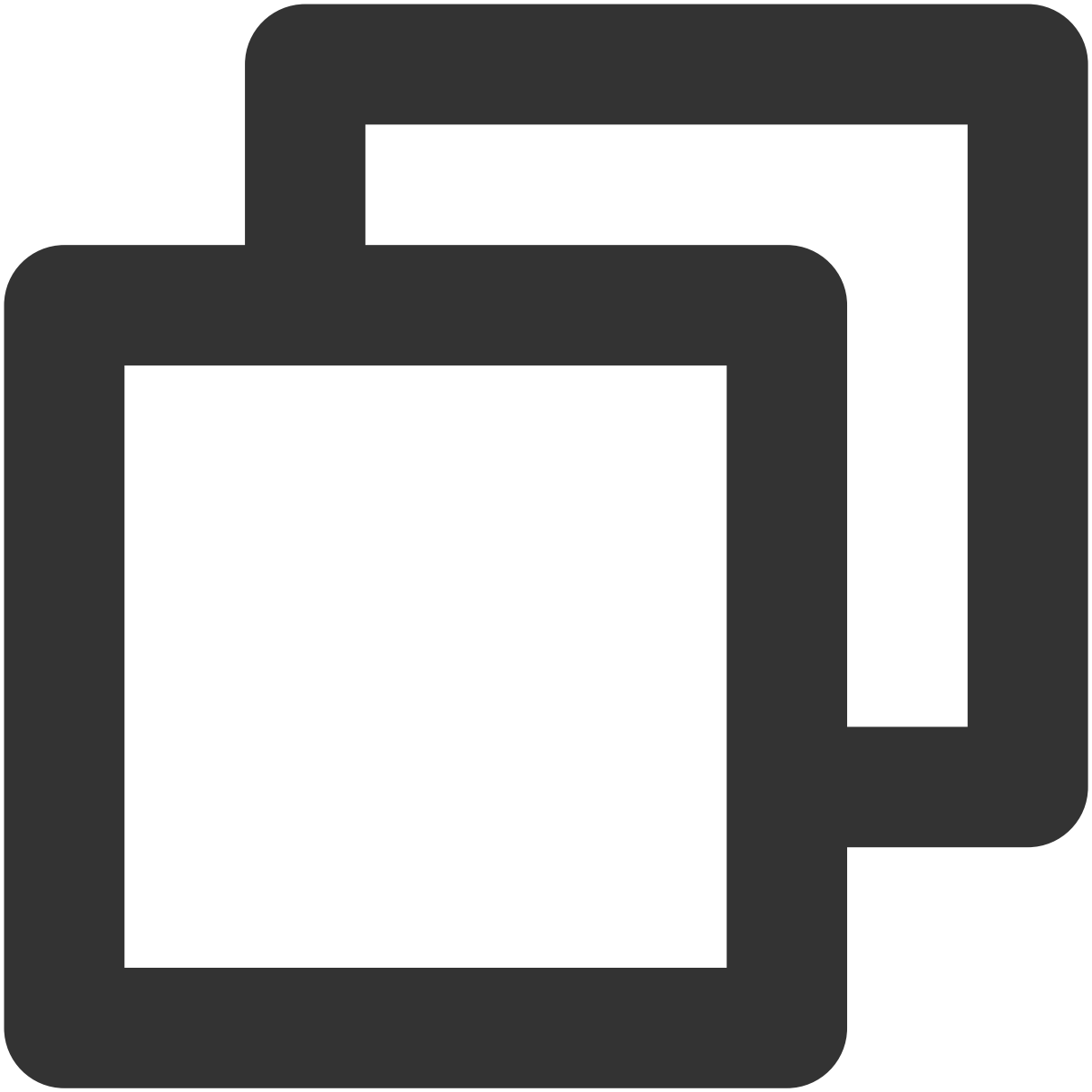
功能说明

用于查询媒体文件的信息。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



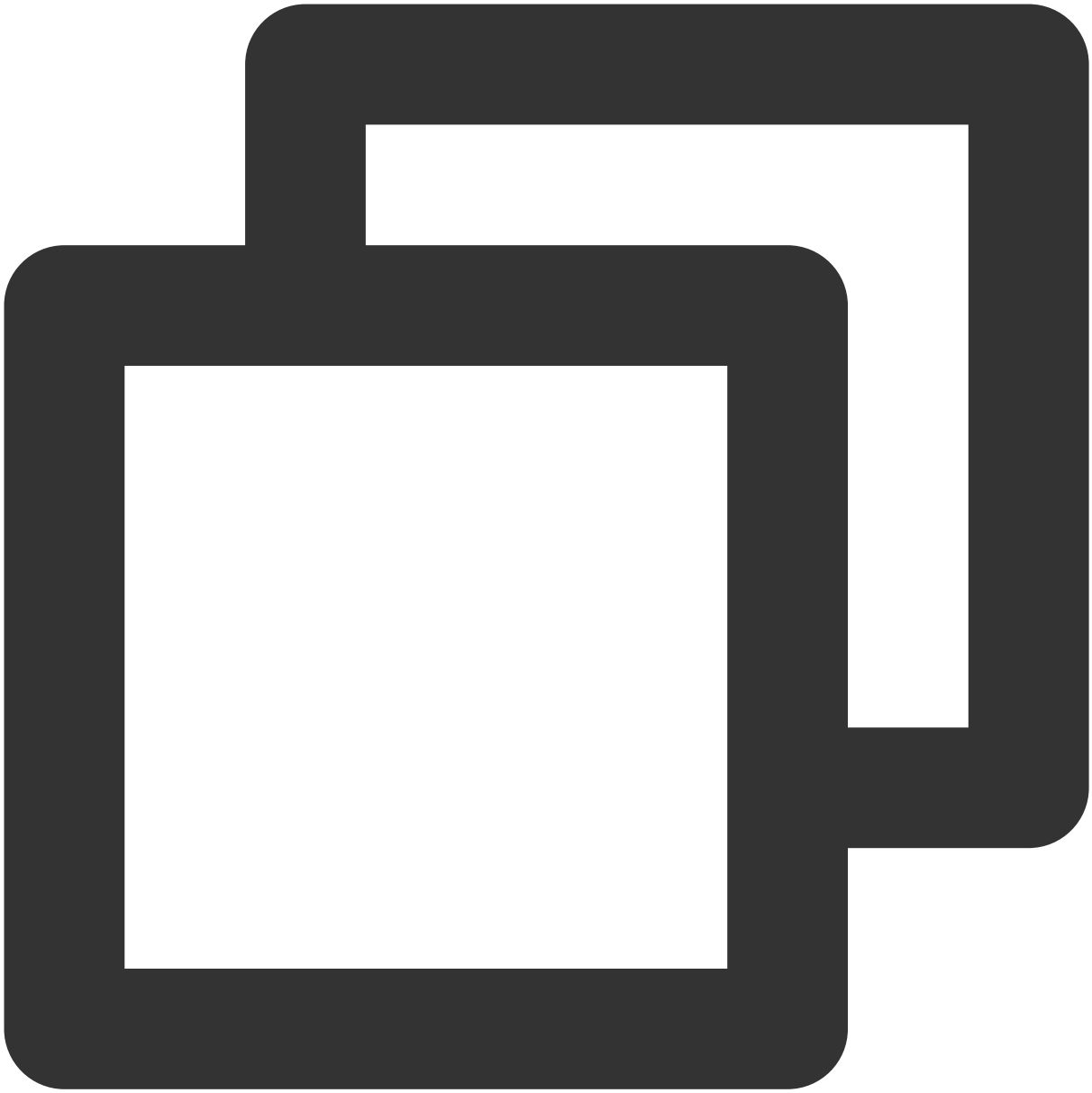
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

```
        'ci-process': 'videoinfo' /** 固定值，必须 */
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
ci-process	操作类型，固定使用 videoinfo	String	是

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Response	响应结果	Object
- - MediaInfo	媒体的详细信息	Object
- - - Stream	流信息	Object
- - - - Video	视频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - Profile	视频编码档位	String
- - - - - Height	视频高，单位 px	Number
- - - - - Width	视频宽，单位 px	Number
- - - - - HasBFrame	是否有B帧。1表示有，0表示无	Number
- - - - - RefFrames	视频编码的参考帧个数	Number
- - - - - Sar	采样宽高比	String
- - - - - Dar	显示宽高比	String
- - - - - PixFormat	像素格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - FieldOrder	场的顺序	String
- - - - - Level	视频编码等级	Number
- - - - - Fps	视频帧率	Number
- - - - - AvgFps	平均帧率	String
- - - - - Timebase	时基	String
- - - - - StartTime	视频开始时间，单位为秒	Number
- - - - - Duration	视频时长，单位为秒	Number
- - - - - Bitrate	比特率，单位为 kbps	Number
- - - - - NumFrames	总帧数	Number
- - - - - Language	语言	String
- - - - Audio	音频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - SampleFmt	采样格式	String
- - - - - SampleRate	采样率	Number
- - - - - Channel	通道数量	Number
- - - - - ChannelLayout	通道格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - Timebase	时基	String
- - - - - StartTime	音频开始时间，单位秒	Number
- - - - - Duration	音频时长，单位秒	Number
- - - - - Bitrate	比特率，单位 kbps	Number
- - - - - Language	语言	String
- - - - Subtitle	字幕信息	Container
- - - - - Index	该流的编号	Number
- - - - - Language	语言，und 表示无查询结果	String
- - - Format	格式信息	Object
- - - - NumStream	Stream（包含 Video、Audio、Subtitle）的数量	Number
- - - - NumProgram	节目的数量	Number
- - - - FormatName	容器格式名字	String
- - - - FormatLongName	容器格式的详细名字	String
- - - - StartTime	起始时间，单位为秒	Number
- - - - Duration	时长，单位为秒	Number
- - - - Bitrate	比特率，单位为 kbps	Number
- - - - Size	大小，单位为 Byte	Number

版权所有：腾讯云计算（北京）有限责任公司

媒体截图接口

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

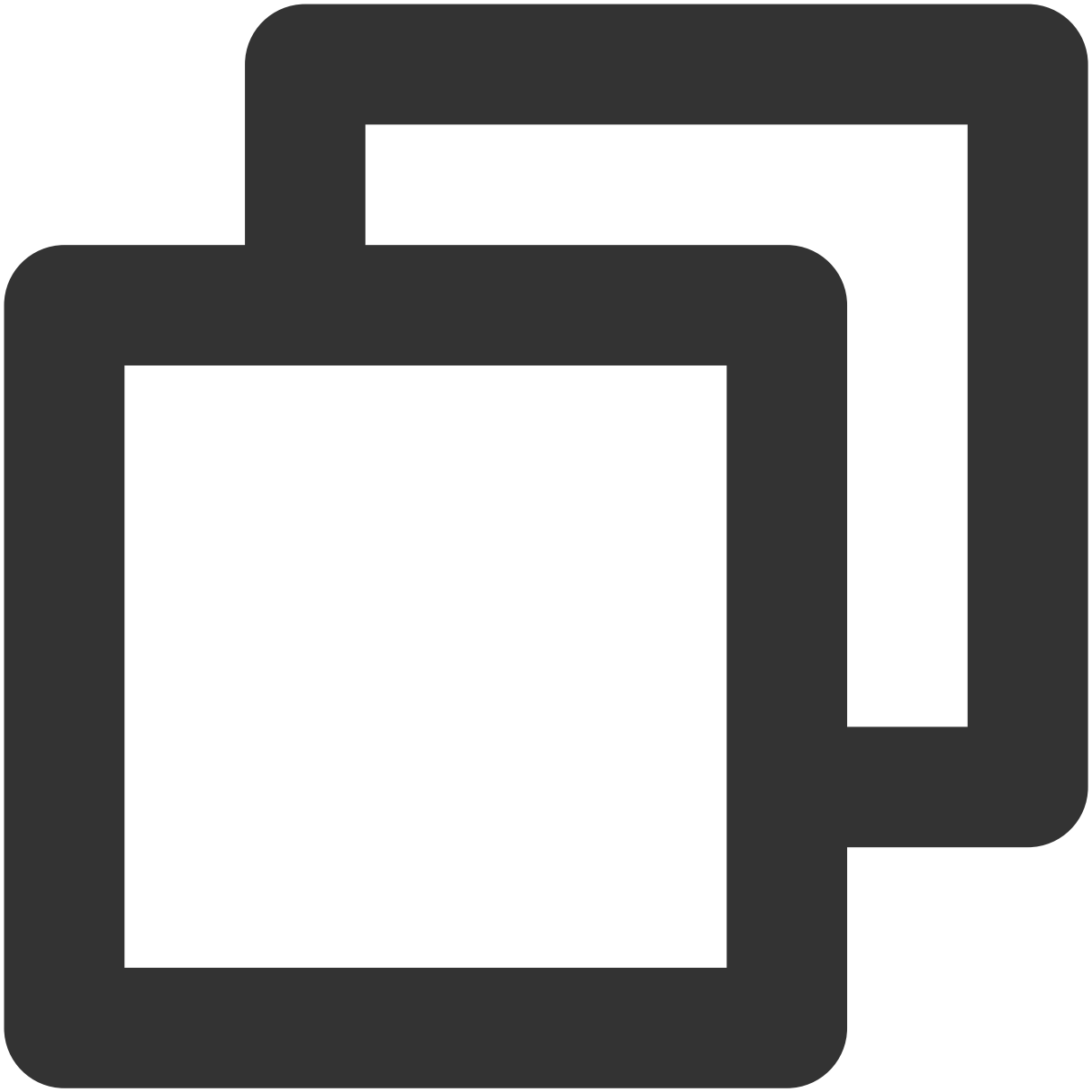
功能说明

用于查询媒体文件在某个时间的截图。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



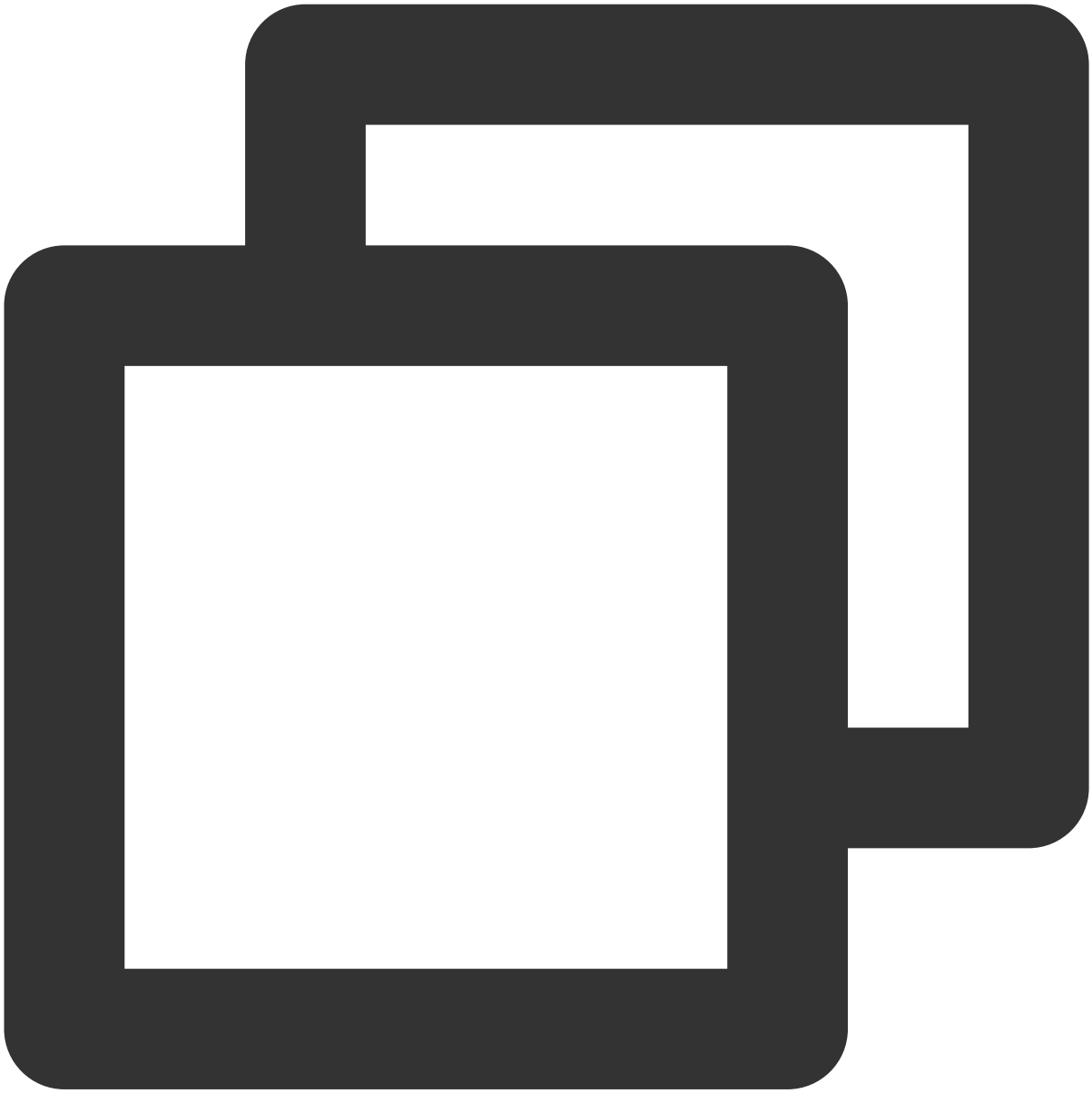
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

```
            '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，默认为'exactframe'，非必须 */
        },
        RawBody: true,
    },
    function(err, data){
        console.log(err || data);
    });
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	对象键（Object 的名称），对象在存储桶中的唯一标识，详情请参见 对象概述	String	是
ci-process	操作类型，固定使用 snapshot	String	是
time	截图的时间点，单位为秒	Number	是
width	截图的宽。默认为0	Number	否
height	截图的高。默认为0。当 width 和 height 都为0时，表示使用视频的宽高；如果单个为0，则以另外一个值按视频宽高比例自动适应	Number	否
format	截图的格式，支持 jpg 和 png，默认 jpg	String	否
rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转off：不旋转，默认值为 auto	String	否
mode	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，默认值为 exactframe	String	否

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Body	返回的文件内容，默认为 String 形式	String

内容审核
图片审核

最近更新时间：2024-03-13 10:07:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于图片审核的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

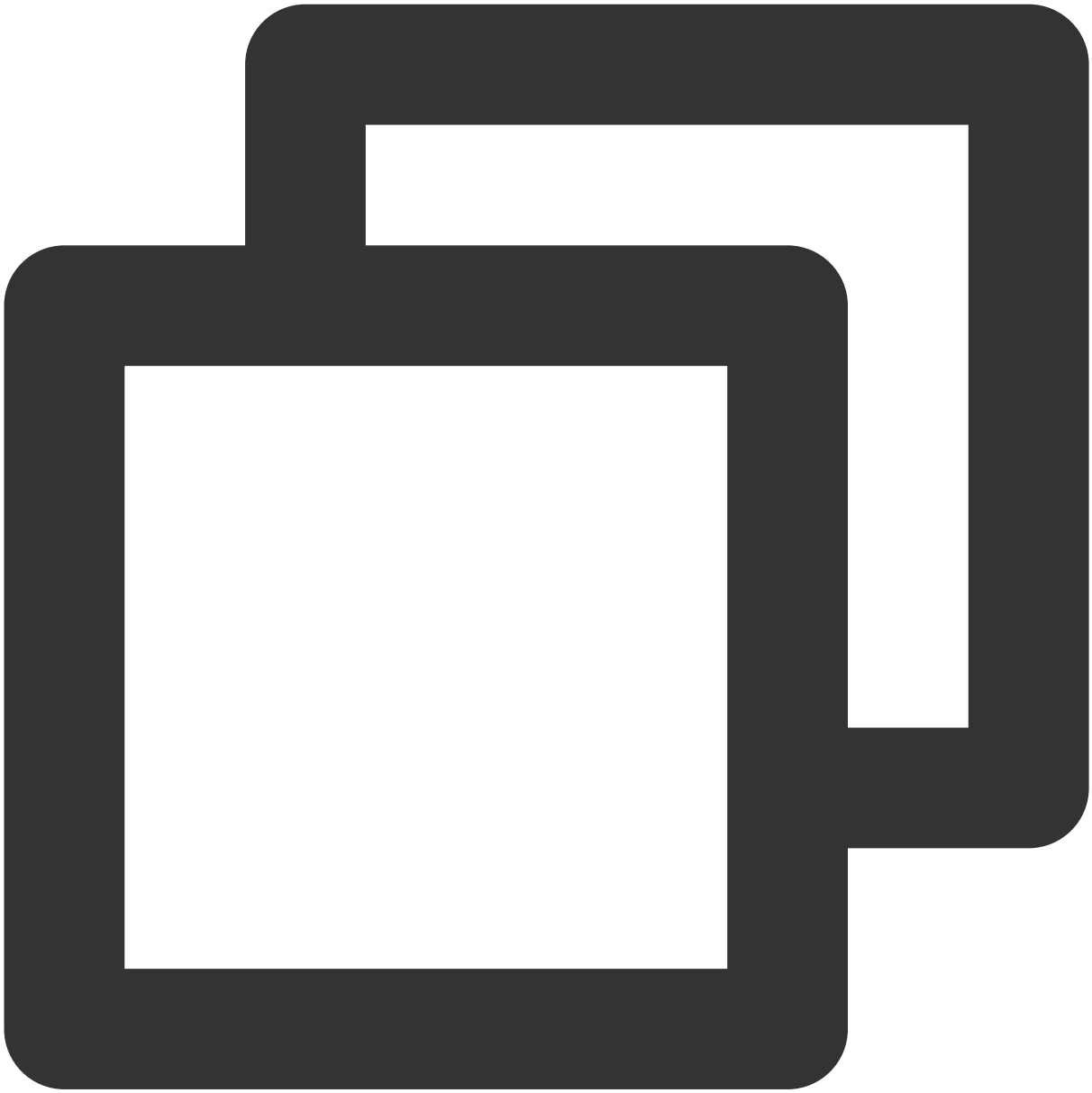
API	操作描述
图片同步审核	对对象存储（Cloud Object Storage，COS）存量数据进行涉黄、违法违规以及广告引导类图片的扫描
图片批量审核	对多个图片进行批量审核

图片同步审核

功能说明

图片审核的存量扫描功能通过借助数据万象的持久化处理接口，实现对 COS 存量数据的涉黄、违法违规以及广告引导类图片的扫描。

使用示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function getImageAuditing() {
  cos.request({
      Bucket: config.Bucket,
      Region: config.Region,
      Method: 'GET',
      Key: '1.png',  /* 存储桶内的图片文件，必须字段 */
```

```
        Query: {
            'ci-process': 'sensitive-content-recognition', /** 固定值，必须 */
            'biz-type': '', /** 审核类型，非必须 */
            'detect-type': 'porn,ads', /** 审核策略，不填写则使用默认策略，非必须 */
            'detect-url': '', /** 审核任意公网可访问的图片链接，非必须 */
            'interval': 5, /** 审核 GIF 动图时，每隔interval帧截取一帧，非必须 */
            'max-frames': 5,  /** 审核 GIF 动图时，最大截帧数，非必须 */
            'large-image-detect': '0', /** 是否需要压缩图片后再审核，非必须 */
            'dataid': '123', /** 自定义图片标识，非必须 */
        },
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

Query 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
biz-type	Query	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否
detect-type	Query	审核类型审核类型，porn（涉黄识别）、ads（广告识别）。可选择多种识别类型用 `,` 连接。	String	否
object-key	Query	图片文件在 bucket 中的位置。	String	是
detect-url	Query	您可以通过填写 `detect-url` 审核任意公网可访问的图片链接。 不填写 `detect-url` 时，后台会默认审核 `ObjectKey` 填写 `detect-url` 时，后台会审核 `detect-url` 链接，无需再填写 `ObjectKey` `detect-url` 示例： http://www.example.com/abc.jpg	String	否
interval	Query	审核 GIF 动图时，可使用该参数进行截帧配置，代表截帧的间隔。例如值设为5，则表示从第1帧开始截取，每隔5帧截取一帧，默认值5	Int	否
max-frames	Query	针对 GIF 动图审核的最大截帧数量，需大于0。例如值设为5，则表示最大截取5帧，默认值为5。	Int	否

large-image-detect	Query	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。 注：压缩最大支持32M的图片，且会收取压缩费用。	String	否
dataid	Query	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否

返回结果说明

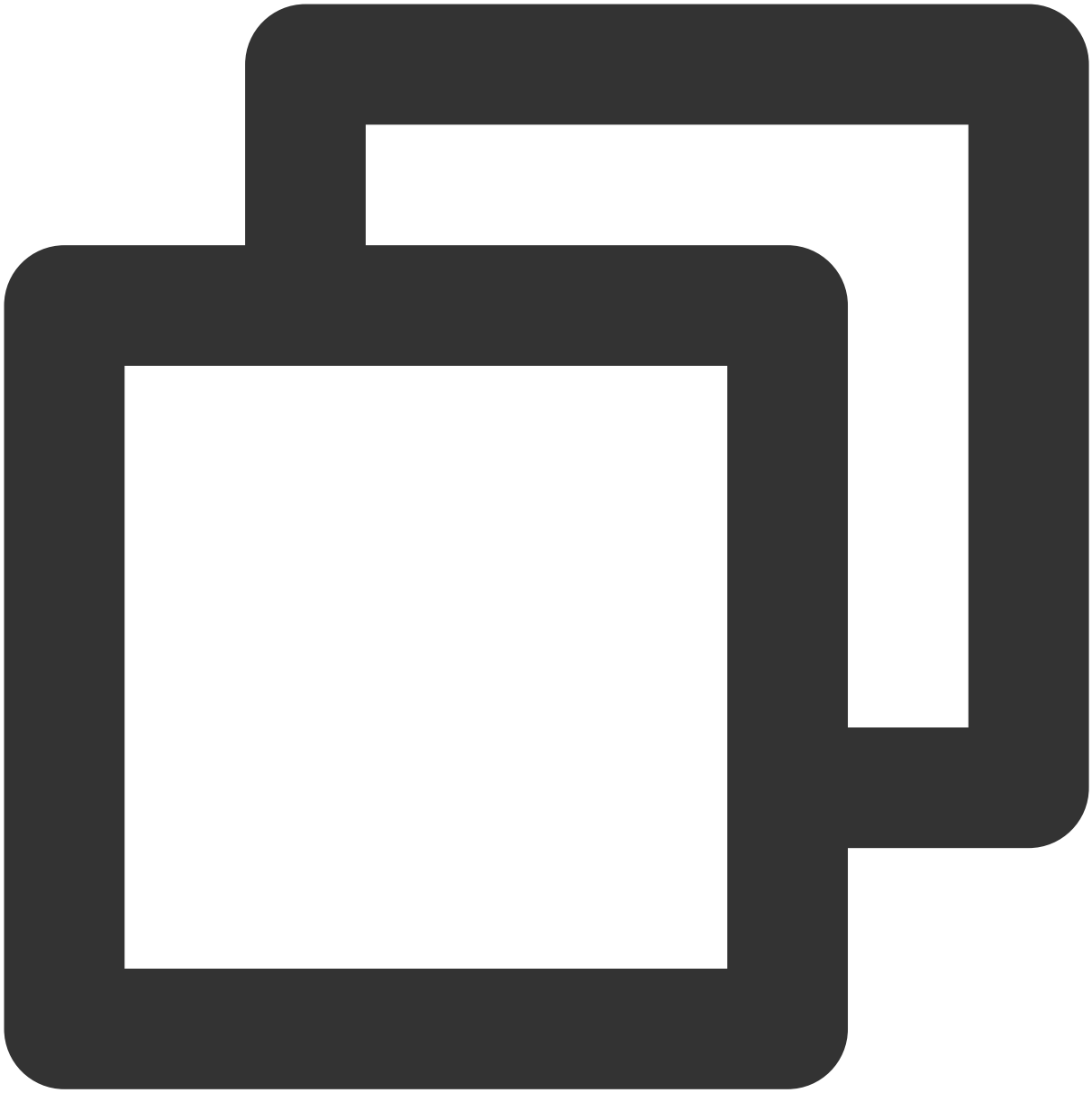
详情请参见 图片同步审核。

图片批量审核

功能说明

图片批量审核接口为同步请求方式，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。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function postImagesAuditing() {
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/image/auditing';
  var body = COS.util.json2xml({
    Request: {
      Input: [{
```

```
          Object: '1.png', /* 需要审核的图片，存储桶里的路径 */
        }, {
          Object: 'a/6.png', /* 需要审核的图片，存储桶里的路径 */
        }],
        Conf: {
          BizType: '', /* 不填写代表默认策略 */
          DetectType: 'Porn' /* 只审核是否涉黄 */
        }
      }
    });
    cos.request({
        Bucket: config.Bucket,
        Region: config.Region,
        Method: 'POST',
        Url: url,
        Key: '/image/auditing', /** 固定值，必须 */
        ContentType: 'application/xml', /** 固定值，必须 */
        Body: body
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

request的数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	图片批量审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容，如有多个图片，请传入多个 Input 结构。	Container Array	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下，使用其中一种：

节点名称（关键字）	父节点	描述	类型	是否必选

Object	Request.Input	存储在 COS 存储桶中的图片文件名称，例如在目录 test 中的文件 image.jpg，则文件名称为 test/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Url	Request.Input	图片文件的链接地址，例如 http://a-1250000.cos.ap-shanghai.myqcloud.com/image.jpg。Object 和 Url 只能选择其中一种。	String	否
Interval	Request.Input	截帧频率，GIF 图检测专用，默认值为5，表示从第一帧（包含）开始每隔5帧截取一帧	Int	否
MaxFrames	Request.Input	最大截帧数量，GIF 图检测专用，默认值为5，表示只截取 GIF 的5帧图片进行审核，必须大于0	Int	否
DataId	Request.Input	图片标识，该字段在结果中返回原始内容，长度限制为512字节	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以 `,` 分隔，例如：Porn,Ads。	String	是
BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否

注意：

通过 Object 进行审核为内网操作，不会产生额外的外网流量。

通过 Url 进行审核，会产生图片所在源站对应的外网流量。

返回结果说明

详情请参见 图片批量审核。

视频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于视频审核的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

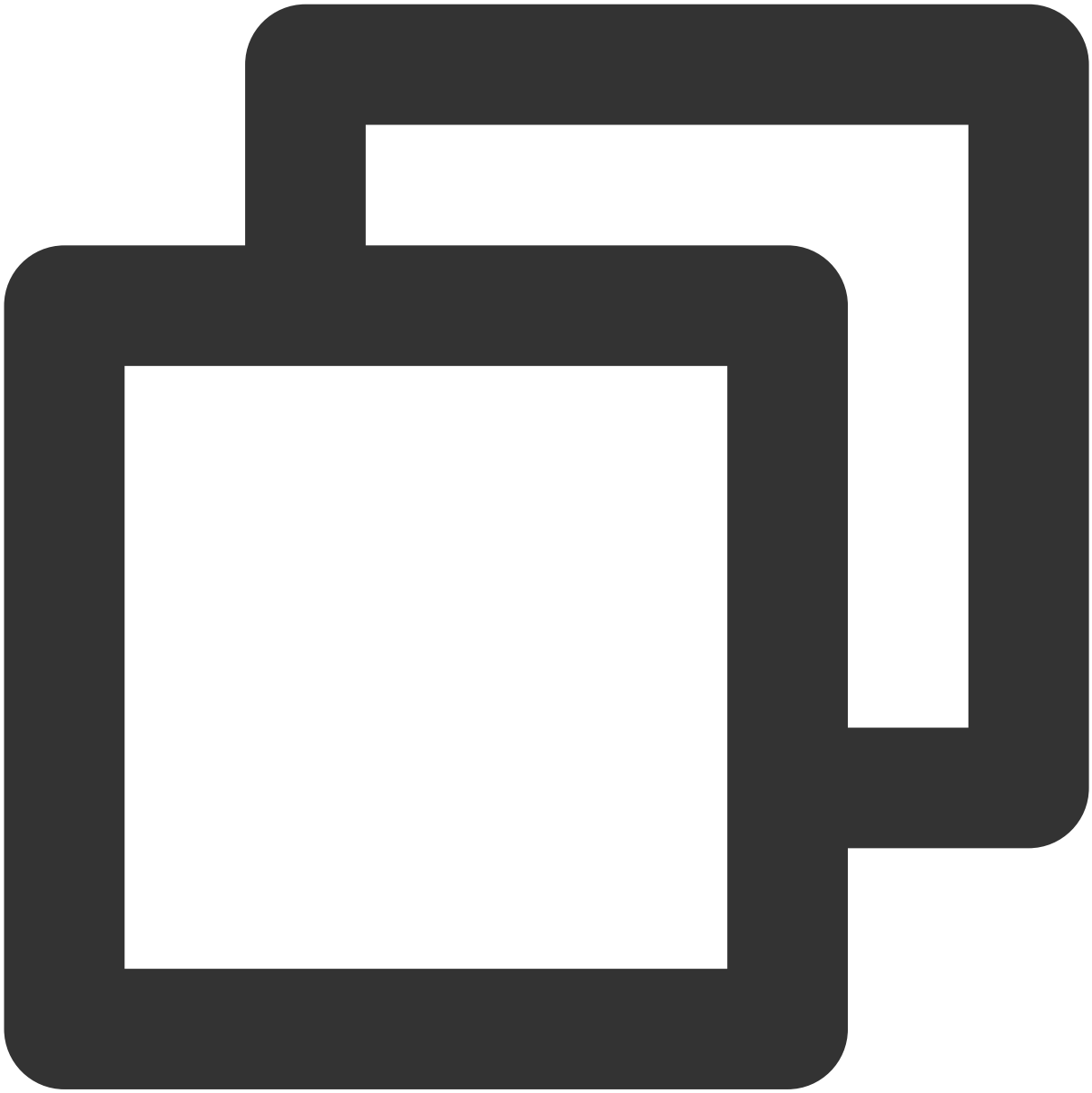
API	操作描述
提交视频审核任务	用于提交一个视频审核任务
查询视频审核任务结果	用于查询指定的视频审核任务

提交视频审核任务

功能说明

用于提交一个视频审核任务。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function postVideoAuditing() {
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/video/auditing';
  var body = COS.util.json2xml({
    Request: {
      Input: {
```

```
          Object: '1.mp4', /* 需要审核的视频文件，存储桶里的路径 */
        },
        Conf: {
          BizType: '',
          DetectType: 'Porn',
          Snapshot: {
            Count: 1000, // 视频截帧数量
          },
          DetectContent: 1, // 是否审核视频声音，0-只审核视频不审核声音；1-审核视频+声音
        }
      }
    });
    cos.request({
        Bucket: config.Bucket,
        Region: config.Region,
        Method: 'POST',
        Url: url,
        Key: '/video/auditing', /** 固定值，必须 */
        ContentType: 'application/xml', /** 固定值，必须 */
        Body: body
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的视频。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	当前 COS 存储桶中的视频文件名称，例如在目录 test 中的文件 video.mp4，则文件名称为 test/video.mp4。	String	是
Url	Request.Input	视频文件的链接地址，例如 http://examplebucket-	String	否

		1250000000.cos.ap-shanghai.myqcloud.com/test.mp4。Object 和 Url 只能选择其中一种。		

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	否
Snapshot	Request.Conf	视频画面的审核通过视频截帧能力截取出一定量的截图，通过对截图逐一审核而实现的，该参数用于指定视频截帧的配置。	Container	是
Callback	Request.Conf	回调地址，以 `http://` 或者 `https://` 开头的地址。	String	否
CallbackVersion	Request.Conf	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String	否
BizType	Request.Conf	审核策略，不带审核策略时使用默认策略。	String	否
DetectContent	Request.Conf	用于指定是否审核视频声音，当值为0时：表示只审核视频画面截图；值为1时：表示同时审核视频画面截图和视频声音。默认值为0。	Integer	否

Container 类型 Snapshot 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Mode	Request.Conf.Snapshot	截帧模式。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片。Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧。	String	否
Count	Request.Conf.Snapshot	视频截帧数量，范围为(0, 10000]。	String	否
TimeInterval	Request.Conf.Snapshot	视频截帧频率，范围为(0, 60]，单位为秒，	Float	否

		支持 float 格式，执行精度精确到毫秒。		

返回结果说明

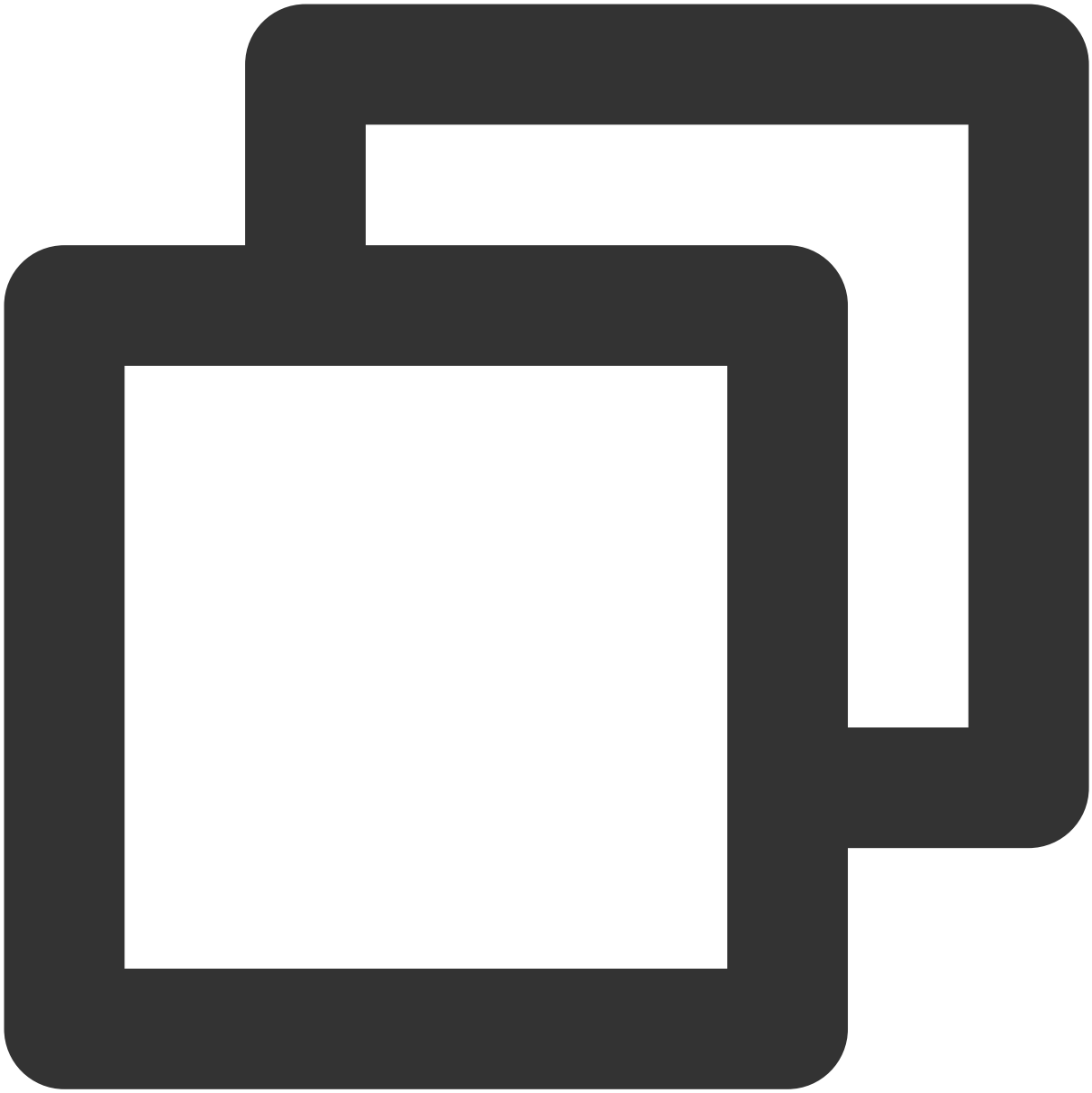
详情请参见 提交视频审核任务。

查询视频审核任务结果

功能说明

用于查询视频审核任务执行情况和结果。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function getVideoAuditingResult() {
  var jobId = 'av14d9ca15af3a11eca0d6525400d88xxx'; // jobId可以通过提交视频审核任务返回
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/video/auditing/' + jobId;
  cos.request({
      Bucket: config.Bucket,
```

```
        Region: config.Region,
        Method: 'GET',
        Key: '/video/auditing/' + jobId,
        Url: url,
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

参数名称	描述	类型	是否必选
jobId	需要查询的任务 ID	String	是

返回结果说明

详情请参见 查询视频审核任务结果。

音频审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于音频审核的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

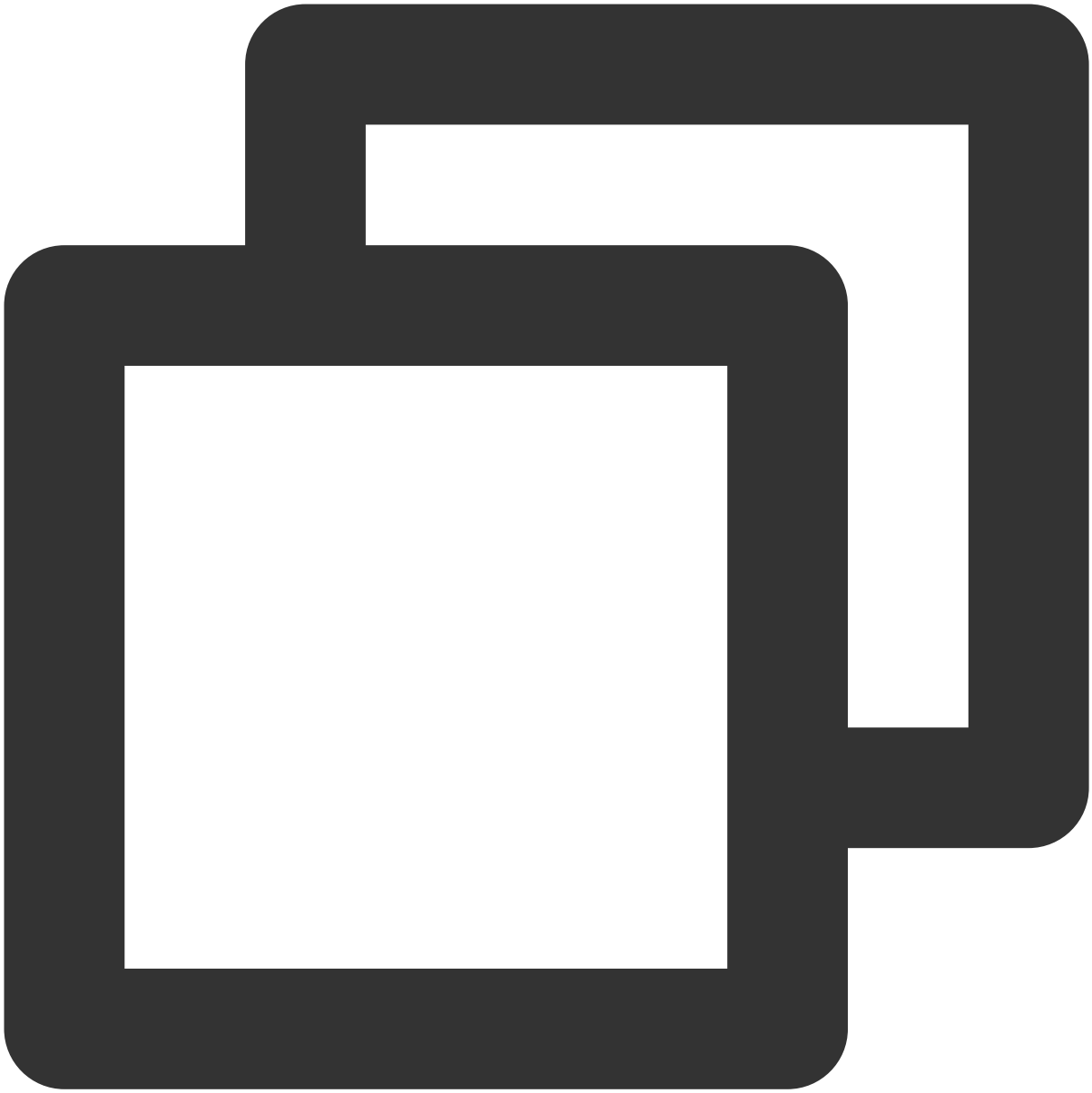
API	操作描述
提交音频审核任务	用于提交一个音频审核任务
查询音频审核任务结果	用于查询指定的音频审核任务

提交音频审核任务

功能说明

用于提交一个音频审核任务。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function postAudioAuditing() {
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/audio/auditing';
  var body = COS.util.json2xml({
    Request: {
      Input: {
```

```
          Object: '1.mp3', /* 需要审核的音频文件，存储桶里的路径 */
        },
        Conf: {
          BizType: '',
          DetectType: 'Porn',
        }
      }
    });
    cos.request({
        Bucket: config.Bucket,
        Region: config.Region,
        Method: 'POST',
        Url: url,
        Key: '/audio/auditing', /** 固定值，必须 */
        ContentType: 'application/xml', /** 固定值，必须 */
        Body: body
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	存储在 COS 存储桶中的音频文件名称，例如在目录 test 中的文件 audio.mp3，则文件名称为 test/audio.mp3。Object 和 Url 只能选择其中一种。	String	否
Url	Request.Input	音频文件的链接地址，例如 http://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/audio.mp3。Object 和 Url 只能选择其中一种。	String	否

版权所有：腾讯云计算（北京）有限责任公司

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告）、Illegal（违法）、Abuse（谩骂），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	否
Callback	Request.Conf	审核结果以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否
CallbackVersion	Request.Conf	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	string	否
BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否

返回结果说明

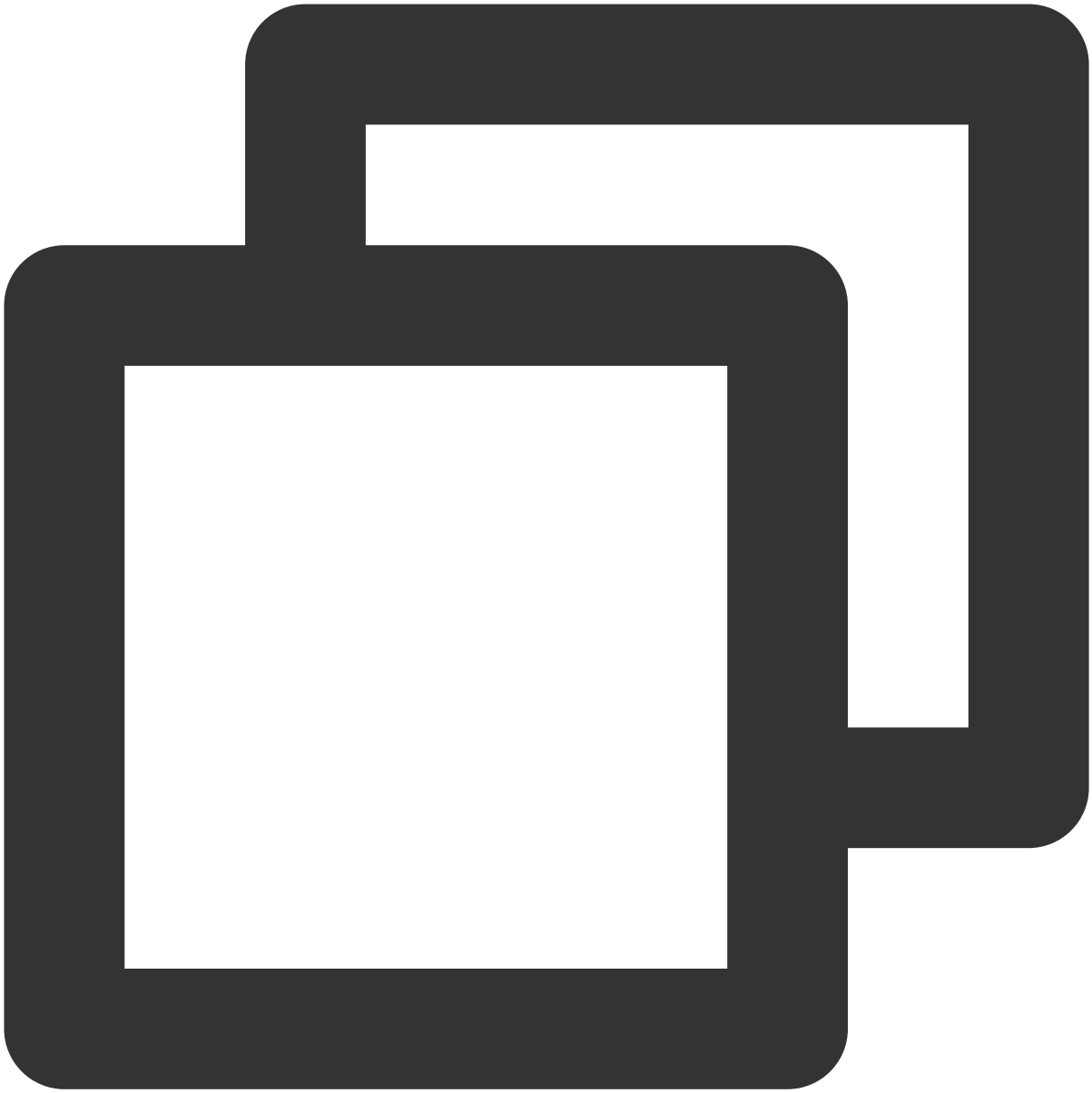
详情请参见 提交音频审核任务。

查询音频审核任务

功能说明

用于查询音频审核任务执行情况和结果。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function getAudioAuditingResult() {
  var jobId = 'sa0c28d41daff411ecb23352540078cxxx'; // jobId可以通过提交音频审核任务返回
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/audio/auditing/' + jobId;
  cos.request({
      Bucket: config.Bucket,
```

```
        Region: config.Region,
        Method: 'GET',
        Key: '/audio/auditing/' + jobId,
        Url: url,
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

参数名称	描述	类型	是否必选
jobId	需要查询的任务 ID	String	是

返回结果说明

详情请参见 查询音频审核任务结果。

文本审核

最近更新时间：2024-03-13 10:07:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于文本审核的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

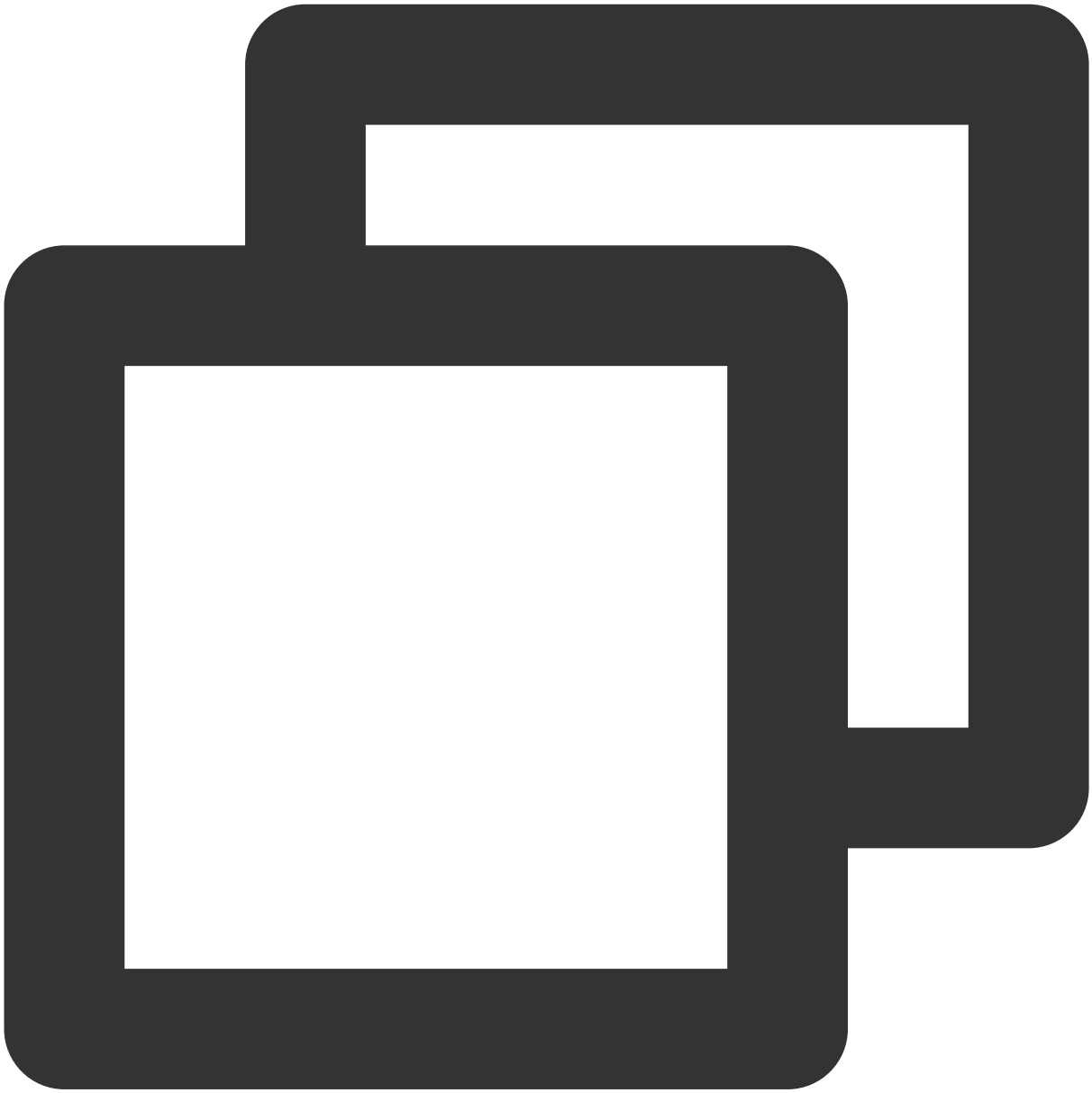
API	操作描述
提交文本审核任务	用于提交一个文本审核任务
查询文本审核任务结果	用于查询指定的文本审核任务

提交文本审核任务

功能说明

用于提交一个文本审核任务。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function postTextAuditing() {
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/text/auditing';
  var body = COS.util.json2xml({
    Request: {
      Input: {
```

```
          Object: 'hello.txt', /* 需要审核的文本文件，存储桶里的路径 */
        },
        Conf: {
          BizType: '',
          DetectType: 'Porn',
        }
      }
    });
    cos.request({
        Bucket: config.Bucket,
        Region: config.Region,
        Method: 'POST',
        Url: url,
        Key: '/text/auditing', /** 固定值，必须 */
        ContentType: 'application/xml', /** 固定值，必须 */
        Body: body
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	文本审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的内容。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	当前 COS 存储桶中的文本文件名称，例如在目录 test 中的文件 img.jpg，则文件名称为 test/img.jpg，文本文件仅支持UTF8编码	String	否

		和 GBK 编码的内容，且文件大小不得超过1MB。		
Content	Request.Input	当传入的内容为纯文本信息，需要先经过 base64 编码，文本编码前的原文长度不能超过10000个 utf8 编码字符。若超出长度限制，接口将会报错。	String	否

注意：

不可同时输入 Object 和 Content。

当选择 Object 时，审核结果为异步返回，可通过 查询文本审核任务结果 API 接口获取返回结果。

当选择 Content 时，审核结果为同步返回，可通过查看响应体 TextAuditingResponse 获取返回结果。

目前仅支持中文、英文、阿拉伯数字的检测与审核。

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告）、Illegal（违法）、Abuse（谩骂），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	否
Callback	Request.Conf	审核结果可以回调形式发送至您的回调地址，支持以 `http://` 或者 `https://` 开头的地址，例如： `http://www.callback.com` 。	String	否
BizType	Request.Conf	审核策略，不填写则使用默认策略。可在控制台进行配置，详情请参见 设置公共审核策略。	String	否

返回结果说明

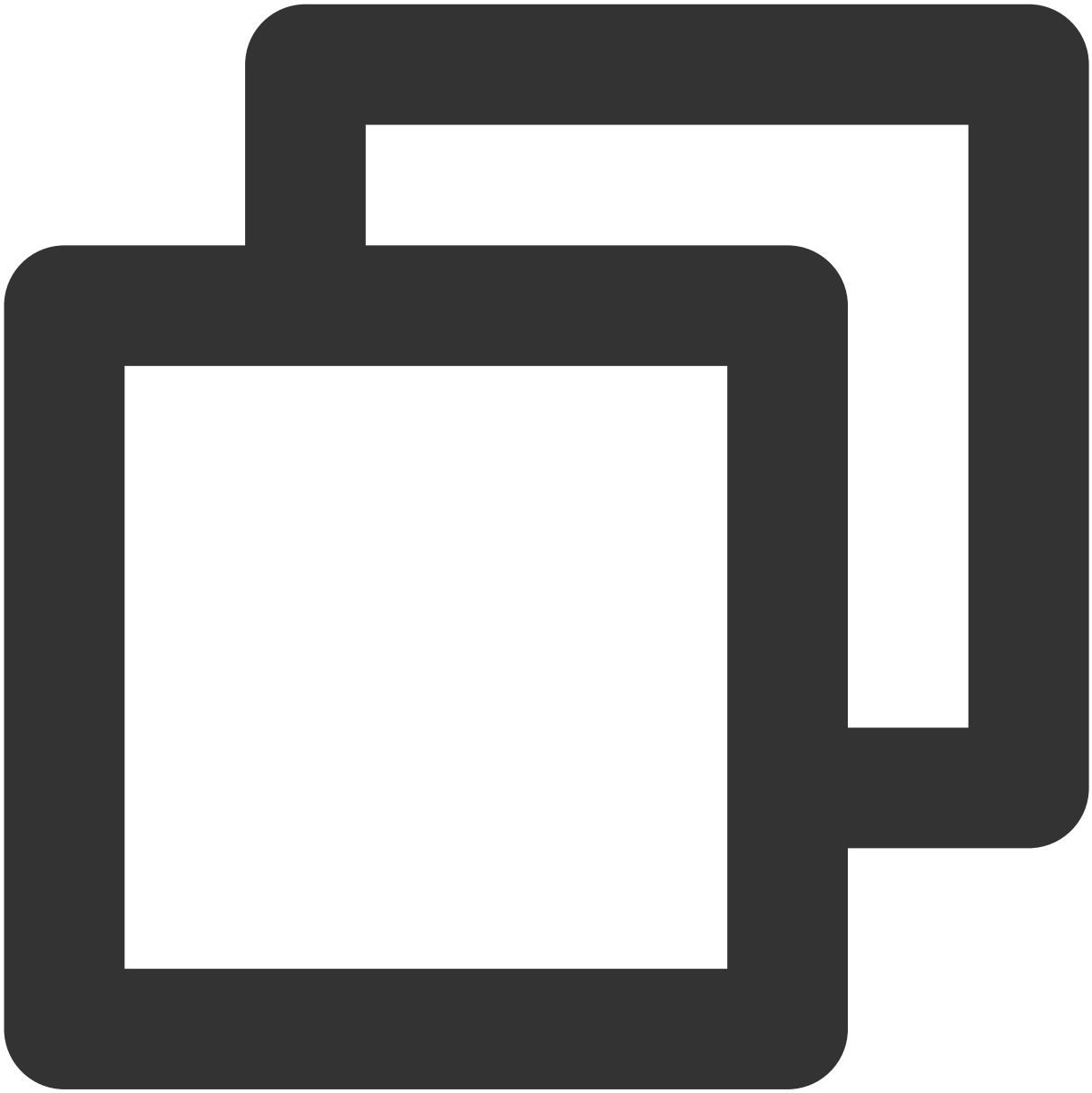
详情请参见 提交文本审核任务。

查询文本审核任务

功能说明

用于查询文本审核任务执行情况和结果。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function getTextAuditingResult() {
  var jobId = 'st8d88c664aff511ecb23352540078cxxx'; // jobId可以通过提交文本审核任务返回
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/text/auditing/' + jobId;
  cos.request({
      Bucket: config.Bucket,
```

```
        Region: config.Region,
        Method: 'GET',
        Key: '/text/auditing/' + jobId,
        Url: url,
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

参数名称	描述	类型	是否必选
jobId	需要查询的任务 ID	String	是

返回结果说明

详情请参见 查询文本审核任务结果。

网页审核

最近更新时间：2024-03-13 10:07:10

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于网页审核的 API 概览和 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

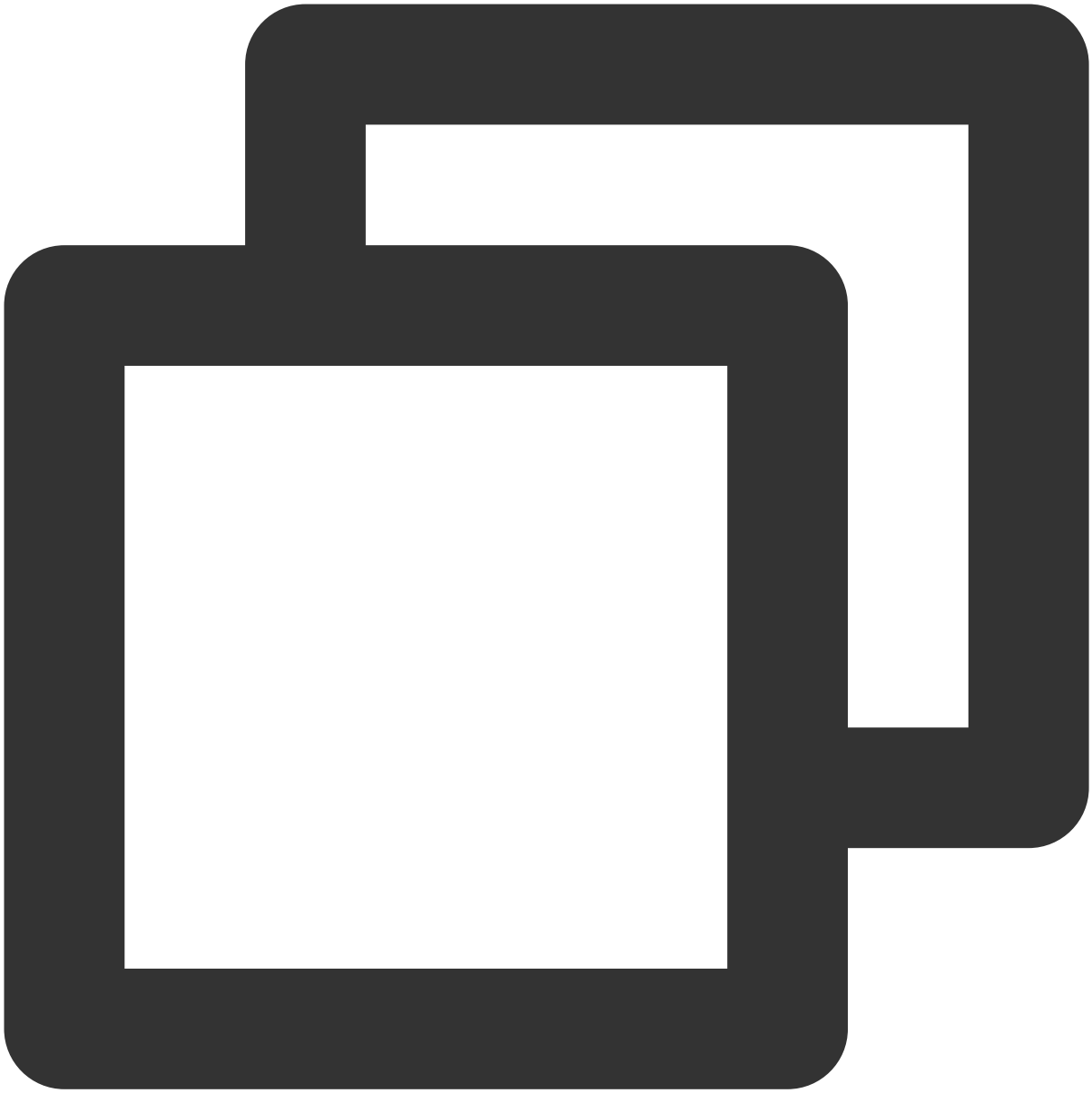
API	操作描述
提交网页审核任务	用于提交一个网页审核任务
查询网页审核任务结果	用于查询指定的网页审核任务

提交网页审核任务

功能说明

用于提交一个网页审核任务。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function postWebpageAuditing() {
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/webpage/auditing';
  var body = COS.util.json2xml({
    Request: {
      Input: {
```

```
        Url: 'https://cloud.tencent.com/', // 存在cos里的资源，审核结果异步返回，可以调用查
      },
      Conf: {
        BizType: '',
        DetectType: 'Porn,Ads',
      }
    }
  });
  cos.request({
      Bucket: config.Bucket,
      Region: config.Region,
      Method: 'POST',
      Url: url,
      Key: '/webpage/auditing', /** 固定值，必须 */
      ContentType: 'application/xml', /** 固定值，必须 */
      Body: body
  },
  function(err, data){
      console.log(err || data);
  });
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	网页审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	需要审核的网页信息。	Container	是
Conf	Request	审核规则配置。	Container	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Url	Request.Input	网页文件的链接地址，例如 `http://www.test.com` 。	String	是

DataId	Request.Input	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否
UserInfo	Request.Input	用户业务字段。	Container	否

Container 节点 UserInfo 的内容：

节点名称（关键字）	父节点	描述	类型	是否必选
TokenId	Request.Input.UserInfo	用户业务 TokenId，长度不超过128字节。	String	否
Nickname	Request.Input.UserInfo	用户业务 Nickname，长度不超过128字节。	String	否
DeviceId	Request.Input.UserInfo	用户业务 DeviceId，长度不超过128字节。	String	否
AppId	Request.Input.UserInfo	用户业务 AppId，长度不超过128字节。	String	否
Room	Request.Input.UserInfo	用户业务 Room，长度不超过128字节。	String	否
IP	Request.Input.UserInfo	用户业务 IP，长度不超过128字节。	String	否
Type	Request.Input.UserInfo	用户业务 Type，长度不超过128字节。	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
DetectType	Request.Conf	审核的场景类型，有效值：Porn（涉黄）、Ads（广告），可以传入多种类型，不同类型以逗号分隔，例如：Porn,Ads。	String	是
Callback	Request.Conf	回调地址，以 `http://` 或者 `https://` 开头的地址。	String	否
ReturnHighlightHtml	Request.Conf	指定是否需要高亮展示网页内的违规文本，查询及回调结果时会根据此参数决定是否返回高亮展	Boolean	否

版权所有：腾讯云计算（北京）有限责任公司

		示的 html 内容。取值为 true 或者 false，默认为 false。		

返回结果说明

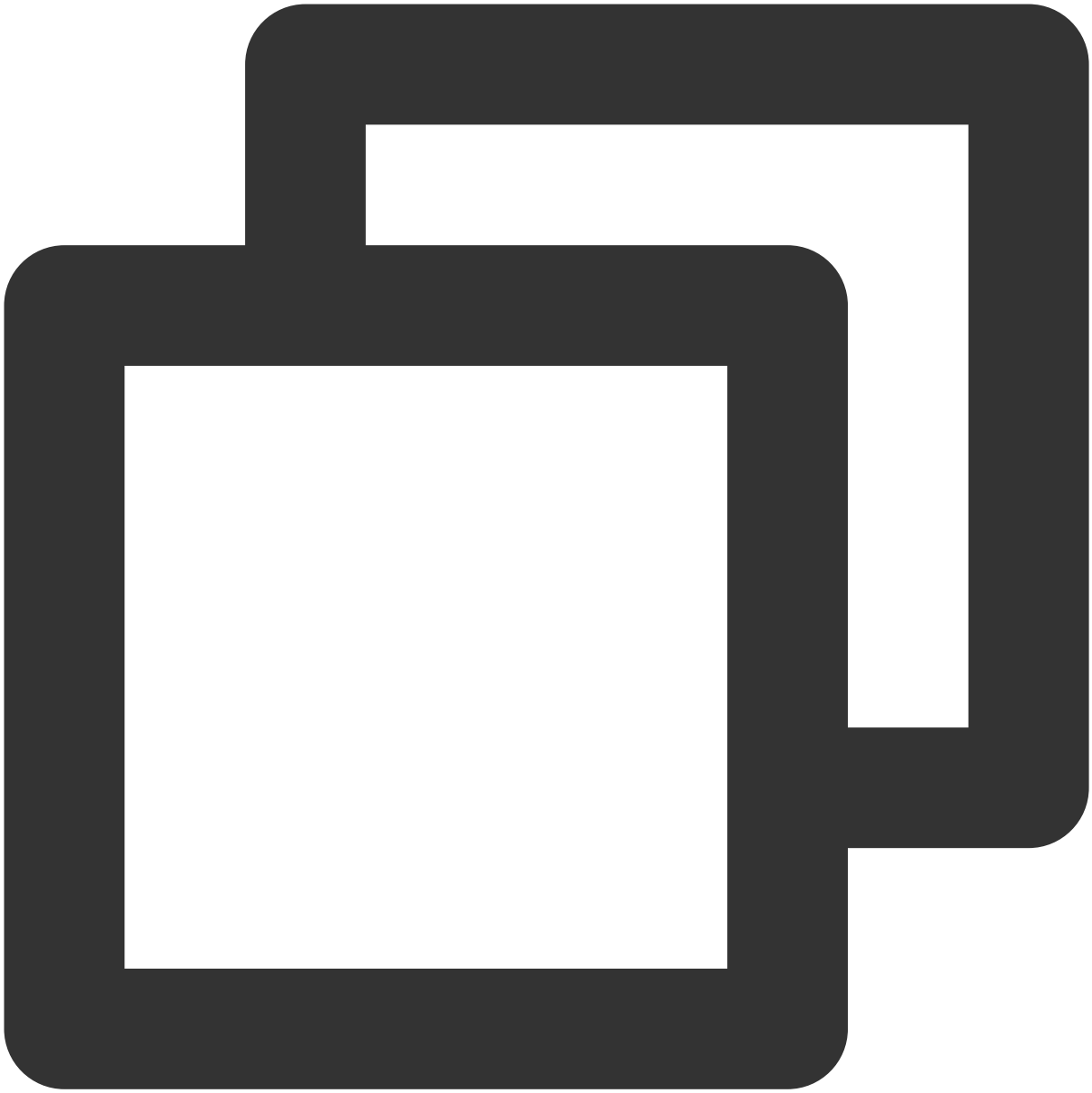
详情请参见 提交网页审核任务。

查询网页审核任务结果

功能说明

用于查询网页审核任务执行情况和结果。

请求示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
function getWebpageAuditingResult() {
  var jobId = 'shce868019aff611ecb1155254009a4xxx'; // jobId可以通过提交网页审核任务返回
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/webpage/auditing/' + jobId;
  cos.request({
      Bucket: config.Bucket,
```

```
        Region: config.Region,
        Method: 'GET',
        Key: '/webpage/auditing/' + jobId,
        Url: url,
    },
    function(err, data){
        console.log(err || data);
    });
}
```

参数说明

参数名称	描述	类型	是否必选
jobId	需要查询的任务 ID	String	是

返回结果说明

详情请参见 查询网页审核任务结果。

文件处理
文件解压

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象文件解压的 API 概览以及 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

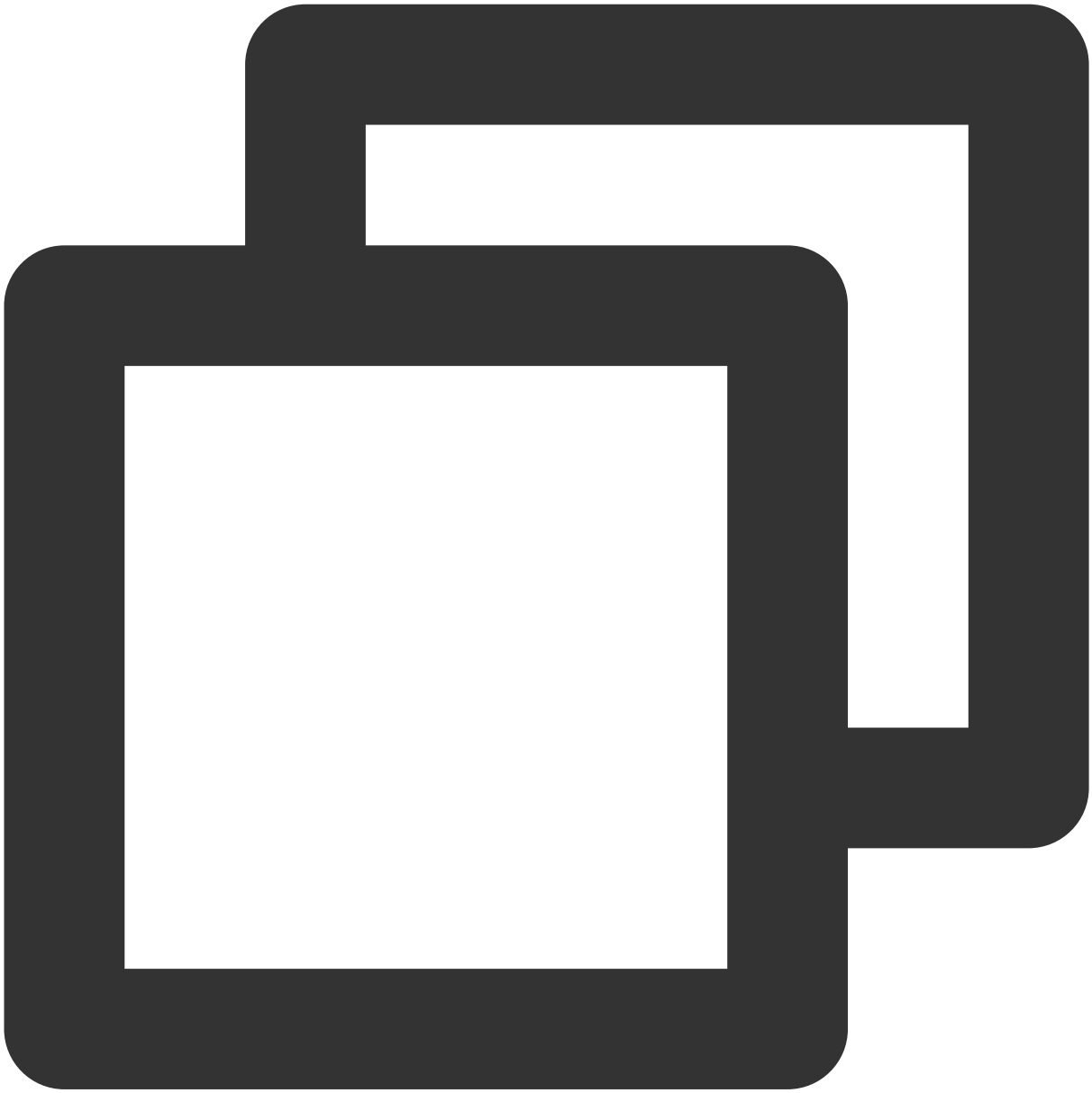
API	操作描述
提交文件解压任务	创建一个文件处理解压任务
查询文件解压结果	查询一个文件处理解压任务

提交文件解压任务

功能说明

以提交任务的方式进行文件解压缩，异步返回压缩包内被解压出来的文件。

示例代码



```
function postFileUnCompressTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs';
  var url = 'https://' + host;
  var body = COS.util.json2xml({
    Request: {
      Tag: 'FileUncompress', // 必须
```

```
        Input: {
          Object: 'compressed.zip', // 文件名，取值为文件在当前存储桶中的完整名称
        },
        Operation: {
          FileUncompressConfig: {
            Prefix: '', // 指定解压后输出文件的前缀，不填则默认保存在存储桶根路径
            PrefixReplaced: '0', // 指定解压后的文件路径是否需要替换前缀,默认0
          },
          Output: {
            Bucket: config.Bucket, // 保存解压后文件的存储桶
            Region: config.Region, // 保存解压后文件的存储桶地域
          },
        },
        // QueueId: '', // 任务所在的队列 ID
        // CallBack: 'http://callback.demo.com', // 任务回调的地址
        // CallBackFormat: 'JSON', // 任务回调格式
        // CallBackType: 'Url', // 任务回调类型，Url 或 TDMQ，默认 Url
      }
    });
    cos.request({
        Method: 'POST',
        Key: 'file_jobs',
        Url: url,
        Body: body,
        ContentType: 'application/xml',
    },
    function(err, data){
        console.log(err || data);
    });
}
postFileUnCompressTask();
```

参数说明

Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，文件解压默认为：FileUncompress。	String	是
Input	Request	包含待操作的文件信息。	Container	是
Operation	Request	包含文件解压的处理规则。	Container	是
QueueId	Request	任务所在的队列 ID。	String	是

CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	文件名，取值为文件在当前存储桶中的完整名称。	String	是

Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileUncompressConfig	Request.Operation	指定文件解压的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
Output	Request.Operation	指定解压后的文件保存的存储桶信息。	Container	是

FileUncompressConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Prefix	Request.Operation.FileUncompressConfig	指定解压后输出文件的前缀，不填则默认保存在存储桶根路径。	String	否
PrefixReplaced	Request.Operation.FileUncompressConfig	指定解压后的文件路径是否需要替换前缀，有效值：	String	否

版权所有：腾讯云计算（北京）有限责任公司

		0：不添加额外的前缀，解压缩将保存在Prefix指定的路径下（不会保留压缩包的名称，仅将压缩包内的文件保存至指定的路径）。 1：以压缩包本身的名称作为前缀，解压缩将保存在Prefix指定的路径下。 2：以压缩包完整路径作为前缀，此时如果不指定Prefix，就是解压到压缩包所在的当前路径（包含压缩包本身名称）。 默认值为0。		

说明：

假设压缩包名为 test.zip，压缩包内的文件为“图片.jpg”，存储在了存储桶 A 的 123 目录下，则压缩包的完整路径为：123/test.zip。

对该压缩包进行解压缩，指定解压缩到存储桶A，并指定 `Prefix` 为 456，则不同的 `PrefixReplaced` 值对应解压后的文件保存情况如下：

0：“图片.jpg”会保存在 456 目录下，“图片.jpg”的完整路径为 456/图片.jpg；

1：“图片.jpg”会以 test 为前缀保存在 456 目录下，“图片.jpg”的完整路径为 456/test/图片.jpg；

2：“图片.jpg”会以 123/test 为前缀保存在 456 目录下，“图片.jpg”的完整路径为 456/123/test/图片.jpg。

Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域。	String	是
Bucket	Request.Operation.Output	保存解压后文件的存储桶。	String	是

返回结果说明

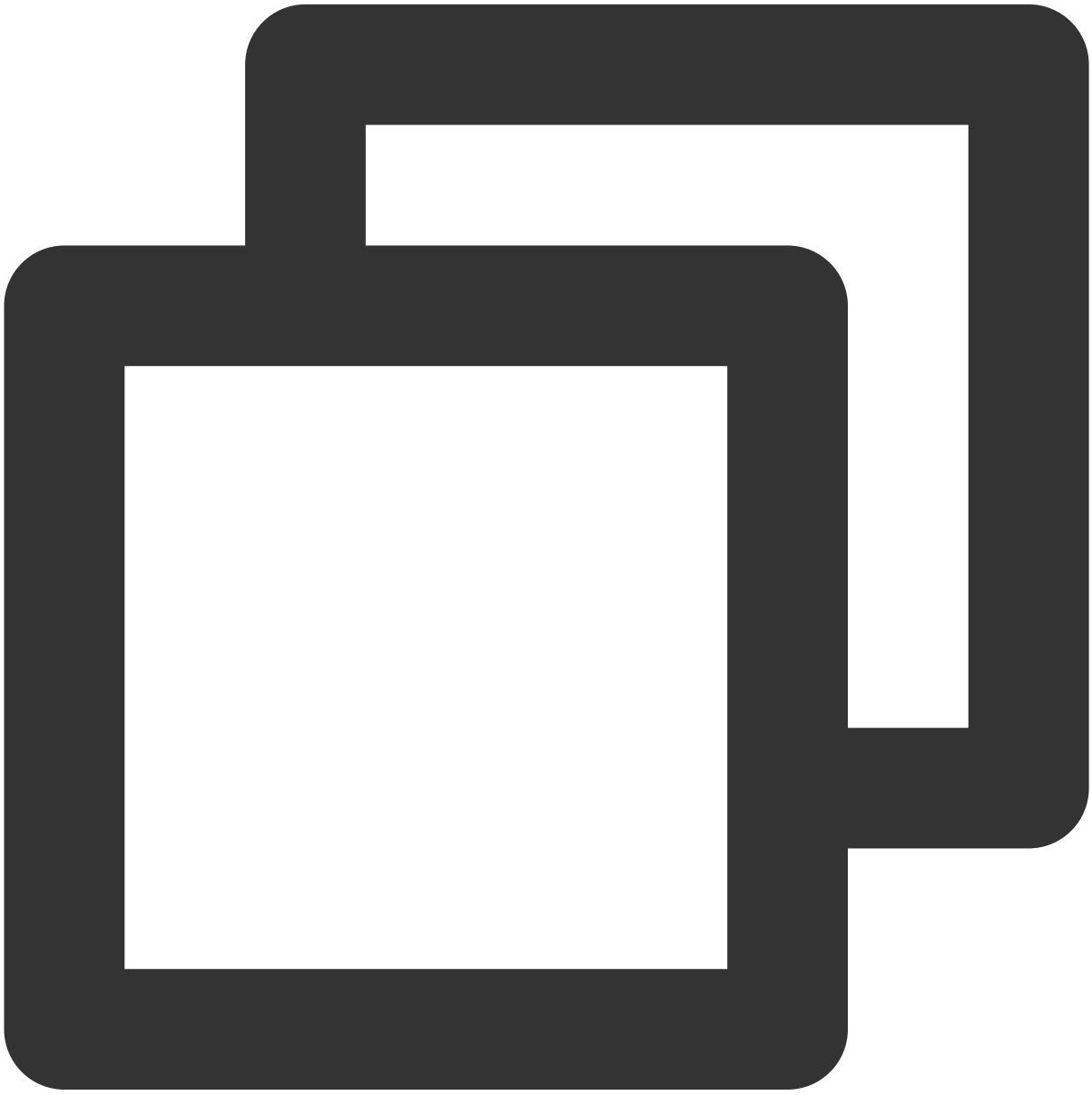
详情请参见 提交文件解压任务。

查询文件解压结果

功能说明

查询一个文件处理任务,根据任务 ID 查询任务详情。

示例代码



```
function getFileUnCompressTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
```

```
  var jobId = 'xxxxxx'; // 提交文件解压任务后会返回当前任务的 jobId
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs/' +
  var url = 'https://' + host;
  cos.request({
    Method: 'GET',
    Key: 'file_jobs/' + jobId,
    Url: url,
  },
  function(err, data){
      console.log(err || data);
  });
}
getFileUnCompressTask();
```

参数说明

参数名称	描述	类型	是否必选
jobId	要查询的任务 ID	String	是

返回结果说明

详情请参见 查询文件解压结果。

文件压缩

最近更新时间：2024-03-13 10:07:10

简介

本文档提供关于数据万象文件压缩的 API 概览以及 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

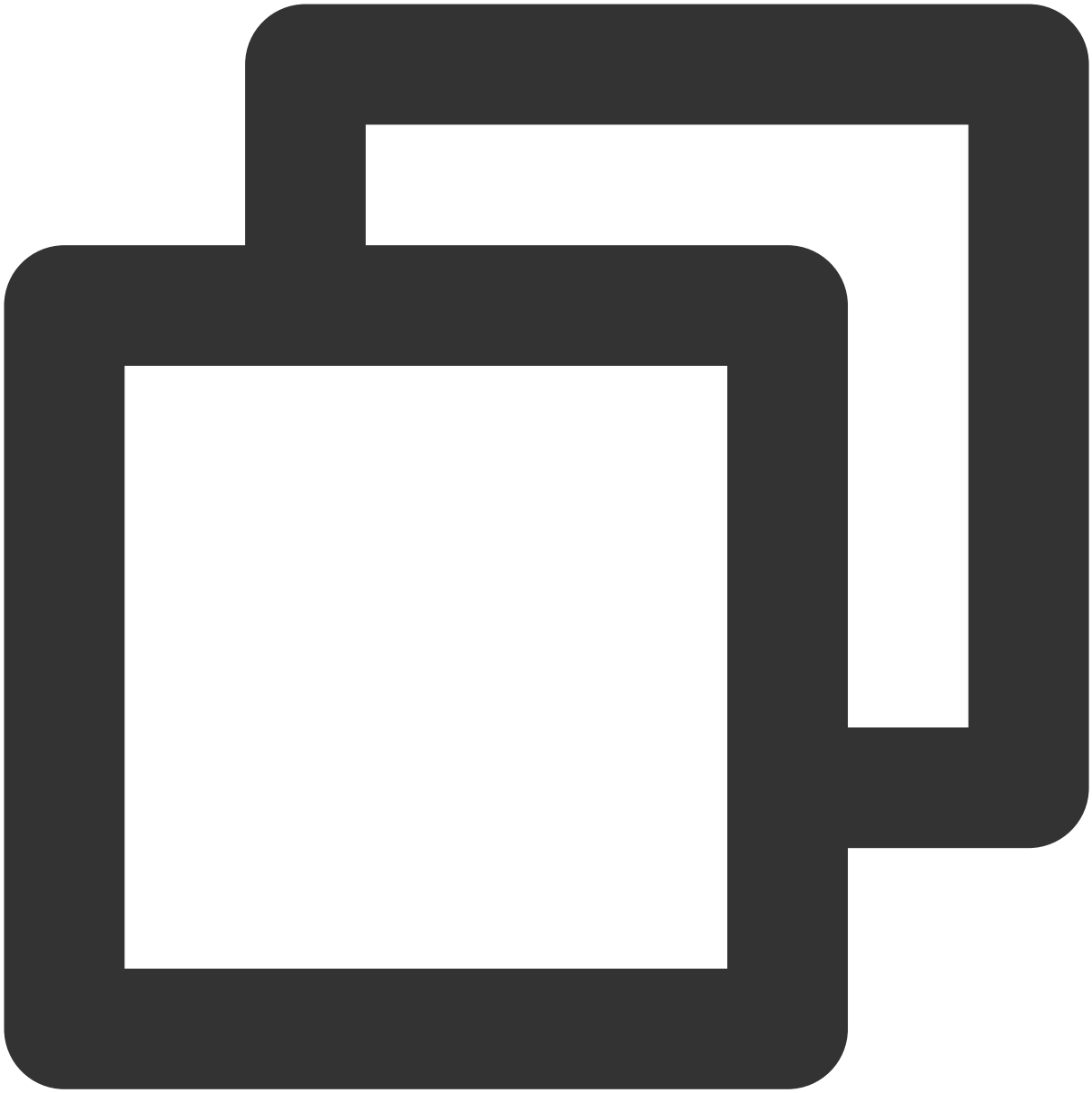
API	操作描述
提交多文件打包压缩任务	创建一个多文件打包压缩任务
查询多文件打包压缩结果	查询指定的多文件打包压缩任务结果

提交多文件打包压缩任务

功能说明

多文件打包压缩功能可以将您的多个文件，打包为 zip 等压缩包格式，以提交任务的方式进行多文件打包压缩，异步返回打包后的文件。

示例代码



```
function postFileCompressTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs';
  var url = 'https://' + host;
  var body = COS.util.json2xml({
    Request: {
      Tag: 'FileCompress', // 必须
```

```
        Operation: {
          FileCompressConfig: {
            Flatten: '0', // 文件打包时，是否需要去除源文件已有的目录结构.0:不需要;1:需要
            Format: 'zip', // 打包压缩的类型，有效值：zip、tar、tar.gz
            // UrlList、Prefix、Key 三者仅能选择一个，不能都为空，也不会同时生效
            // UrlList: '', // 索引文件的对象地址
            Prefix: 'testCompress/', // 目录前缀
            // Key: [], // 支持对存储桶中的多个文件进行打包，个数不能超过 1000，总大小不超过50G
          },
          Output: {
            Bucket: config.Bucket, // 保存压缩后文件的存储桶
            Region: config.Region, // 保存压缩后文件的存储桶地域
            Object: 'testCompress/compressed.zip', // 压缩后文件的文件名
          },
          UserData: '',
        },
        // QueueId: '', // 任务所在的队列 ID
        // CallBack: 'http://callback.demo.com', // 任务回调的地址
        // CallBackFormat: 'JSON', // 任务回调格式
        // CallBackType: 'Url', // 任务回调类型，Url 或 TDMQ，默认 Url
      }
    });
    cos.request({
        Method: 'POST',
        Key: 'file_jobs',
        Url: url,
        Body: body,
        ContentType: 'application/xml',
    },
    function(err, data){
        console.log(err || data);
    });
}
postFileCompressTask();
```

参数说明

Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，多文件打包压缩默认为：FileCompress。	String	是
Operation	Request	包含文件打包压缩的处理规则。	Container	是

QueueId	Request	任务所在的队列 ID。	String	否
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情请参见 CallBackMqConfig	Container	否

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileCompressConfig	Request.Operation	指定文件打包压缩的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
Output	Request.Operation	指定打包压缩后的文件保存的地址信息。	Container	是

Container 类型 FileCompressConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Flatten	Request.Operation.FileCompressConfig	文件打包时，是否需要去除源文件已有的目录结构，有效值： 0：不需要去除目录结构，打包后压缩包中的文件会保留原有的目录结构； 1：需要，打包后压缩包内的文件会去除原有的目录结构，所有文件都在同一层级。 例如：源文件 URL 为 https://domain/source/test.mp4, 则源文件路径为 source/test.mp4，如	String	是

		果为 1，则 ZIP 包中该文件路径为 test.mp4；如果为0， ZIP 包中该文件路径为 source/test.mp4。		
Format	Request.Operation.FileCompressConfig	打包压缩的类型，有效值：zip、tar、tar.gz。	String	是
UrlList	Request.Operation.FileCompressConfig	支持将需要打包的文件整理成索引文件，后台将根据索引文件内提供的文件url，打包为一个压缩包文件。索引文件需要保存在当前存储桶中，本字段需要提供索引文件的对象地址，例如：/test/index.csv。索引文件格式：仅支持 CVS文件，一行一条URL（仅支持本存储桶文件），如有多列字段，默认取第一列作为URL。最多不超过10000个文件, 总大小不超过50G, 否则会导致任务失败。	String	否
Prefix	Request.Operation.FileCompressConfig	支持对存储桶中的某个前缀进行打包，如果需要对某个目录进行打包，需要加/，例如test目录打包，则值为：test/。最多不超过10000个文件，总大小不超过50G，否则会导致任务失败。	String	否
Key	Request.Operation.FileCompressConfig	支持对存储桶中的多个文件进行打包，个数不能超过 1000, 总大小不超过50G，否则会导致任务失败。	String 数组	否

注意：

UrlList、Prefix、Key 三者仅能选择一个，不能都为空，也不会同时生效。如果填了多个，会按优先级 UrlList > Prefix > Key 取最高优先级执行。

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域。	String	是
Bucket	Request.Operation.Output	保存压缩后文件的存储桶。	String	是
Object	Request.Operation.Output	压缩后文件的文件名	String	是

版权所有：腾讯云计算（北京）有限责任公司

返回结果说明

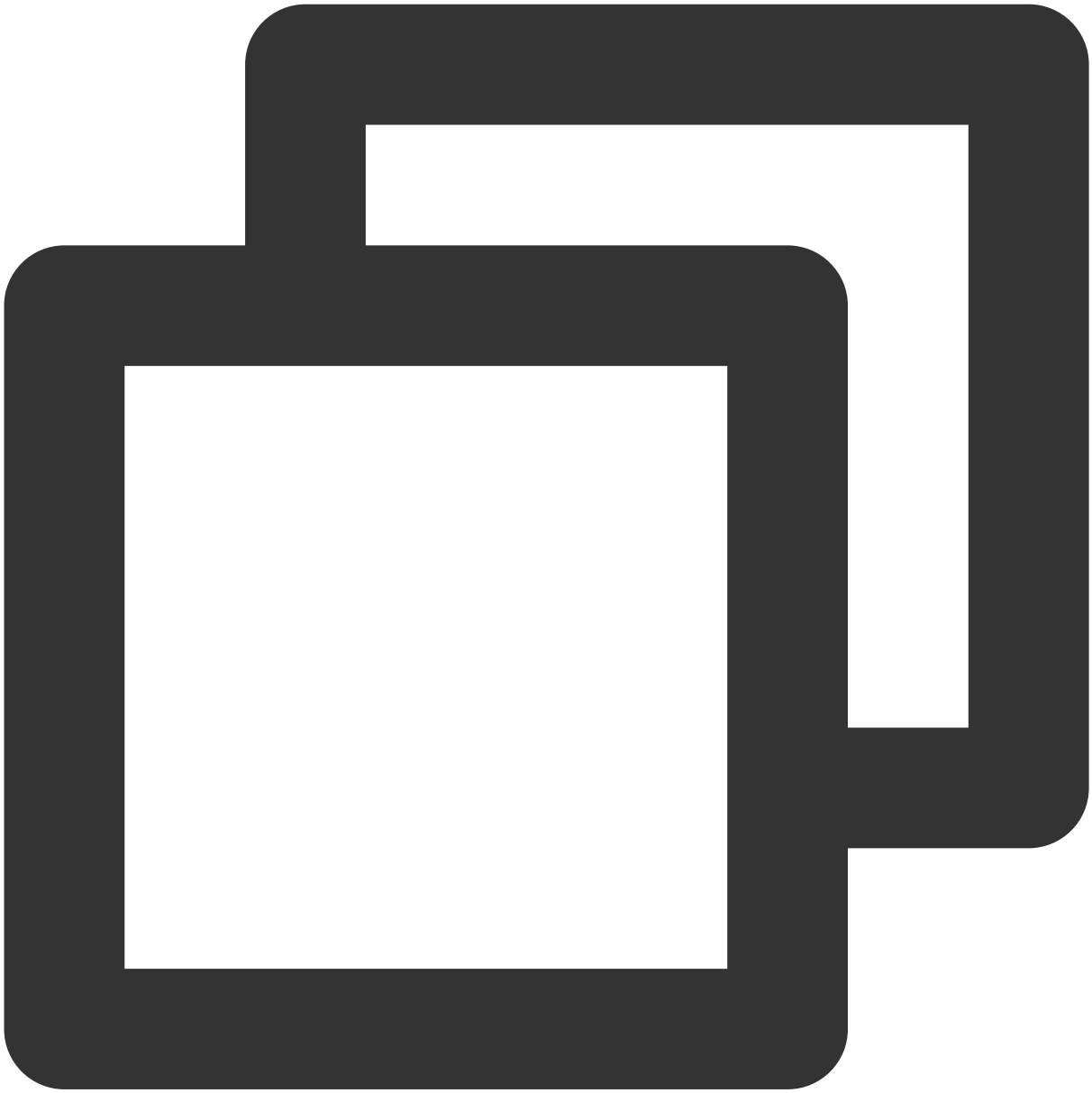
详情请参见 提交多文件打包压缩任务。

查询多文件打包压缩结果

功能说明

查询一个文件处理任务,根据任务 ID 查询任务详情。

示例代码



```
function getFileCompressTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var jobId = 'xxxxxx'; // 提交文件压缩任务后会返回当前任务的 jobId
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs/' +
  var url = 'https://' + host;
  cos.request({
    Method: 'GET',
```

```
      Key: 'file_jobs/' + jobId,
      Url: url,
    },
    function(err, data){
        console.log(err || data);
    });
}
getFileCompressTask();
```

参数说明

参数名称	描述	类型	是否必选
jobId	要查询的任务 ID	String	是

返回结果说明

详情请参见 查询多文件打包压缩结果。

文件哈希值计算

最近更新时间：2024-03-13 10:07:11

简介

本文档提供关于数据万象文件哈希值计算的 API 概览以及 SDK 示例代码。

注意：

COS Javascript SDK 版本需要大于等于 v1.3.1。

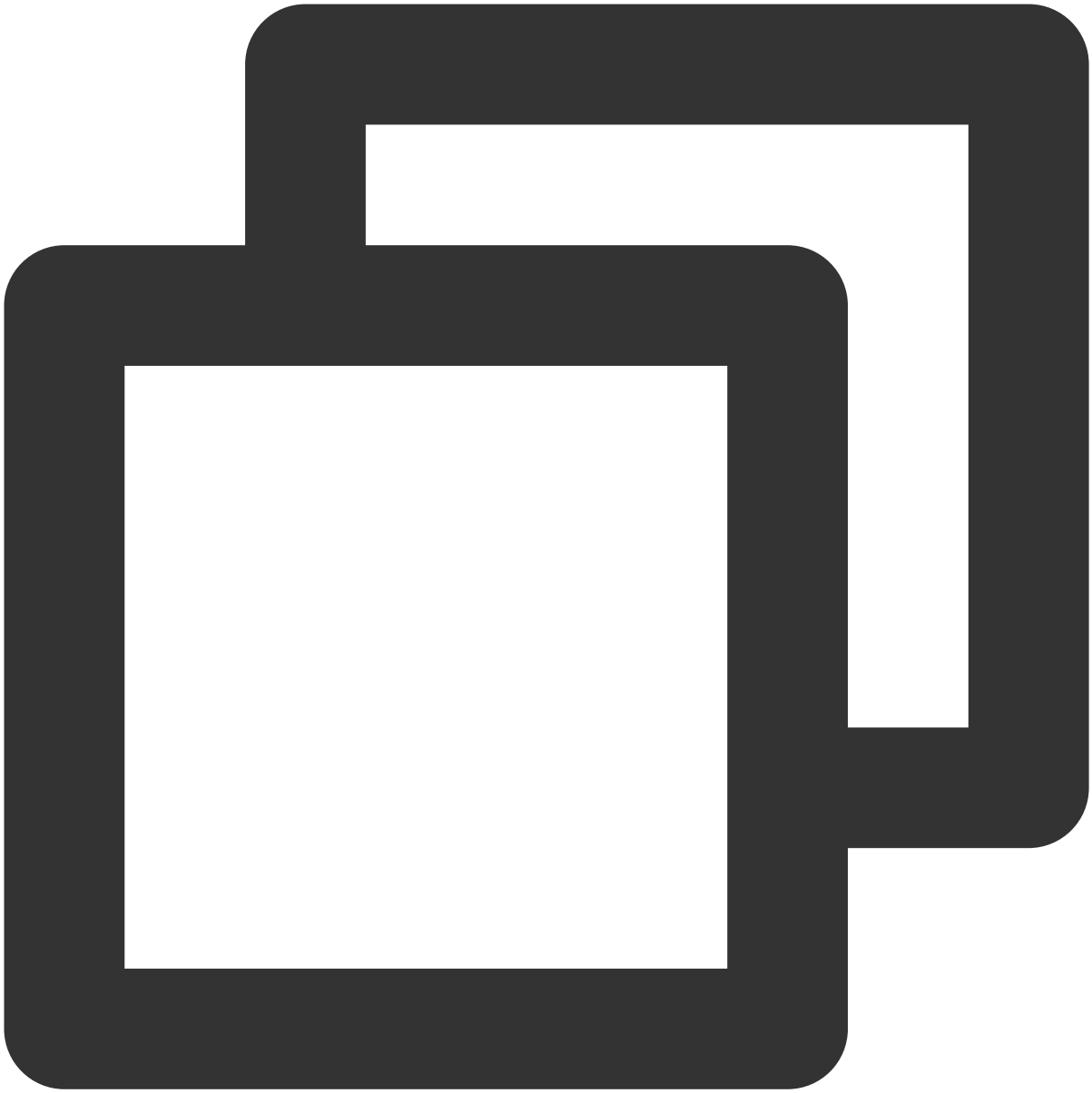
API	操作描述
提交哈希值计算任务	创建一个文件处理哈希值计算任务
查询哈希值计算结果	查询一个文件处理哈希值计算结果任务

提交哈希值计算任务

功能说明

以提交任务的方式进行文件哈希值计算，异步返回计算得到的哈希值。

示例代码



```
function postFileHashTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs';
  var url = 'https://' + host;
  var body = COS.util.json2xml({
    Request: {
      Tag: 'FileHashCode', // 必须
```

```
        Input: {
          Object: 'test/1.pdf', // 文件名，取值为文件在当前存储桶中的完整名称
        },
        Operation: {
          FileHashCodeConfig: {
            Type: 'MD5', // 哈希值的算法类型，有效值：MD5、SHA1、SHA256
            AddToHeader: 'false', // 是否将计算得到的哈希值添加至文件自定义header，有效值：tr
          },
          // UserData: '', // 透传用户信息，可打印的 ASCII 码，长度不超过1024
        },
        // QueueId: '', // 任务所在的队列 ID
        // CallBack: 'http://callback.demo.com', // 任务回调的地址
        // CallBackFormat: 'JSON', // 任务回调格式
        // CallBackType: 'Url', // 任务回调类型，Url 或 TDMQ，默认 Url
      }
    });
    cos.request({
        Method: 'POST',
        Key: 'file_jobs',
        Url: url,
        Body: body,
        ContentType: 'application/xml',
    },
    function(err, data){
        console.log(err || data);
    });
}
postFileHashTask();
```

参数说明

Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，哈希值计算任务类型为：FileHashCode。	String	是
Operation	Request	包含哈希值计算的处理规则。	Container	是
QueueId	Request	任务所在的队列 ID。	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否

CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情请参见 CallBackMqConfig	Container	否

Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	文件名，取值为文件在当前存储桶中的完整名称。	String	是

Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileHashCodeConfig	Request.Operation	指定哈希值计算的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否

FileHashCodeConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Type	Request.Operation.FileHashCodeConfig	哈希值的算法类型，有效值：MD5、SHA1、SHA256。	String	是
AddToHeader	Request.Operation.FileHashCodeConfig	是否将计算得到的哈希值添加至文件自定义 header, 有效值：true、false，默认值为 false。 自定义header根据 `Type` 的值变化，例如 `Type` 值为MD5时，自定义 heade r为 x-cos-meta-md5。	String	否

返回结果说明

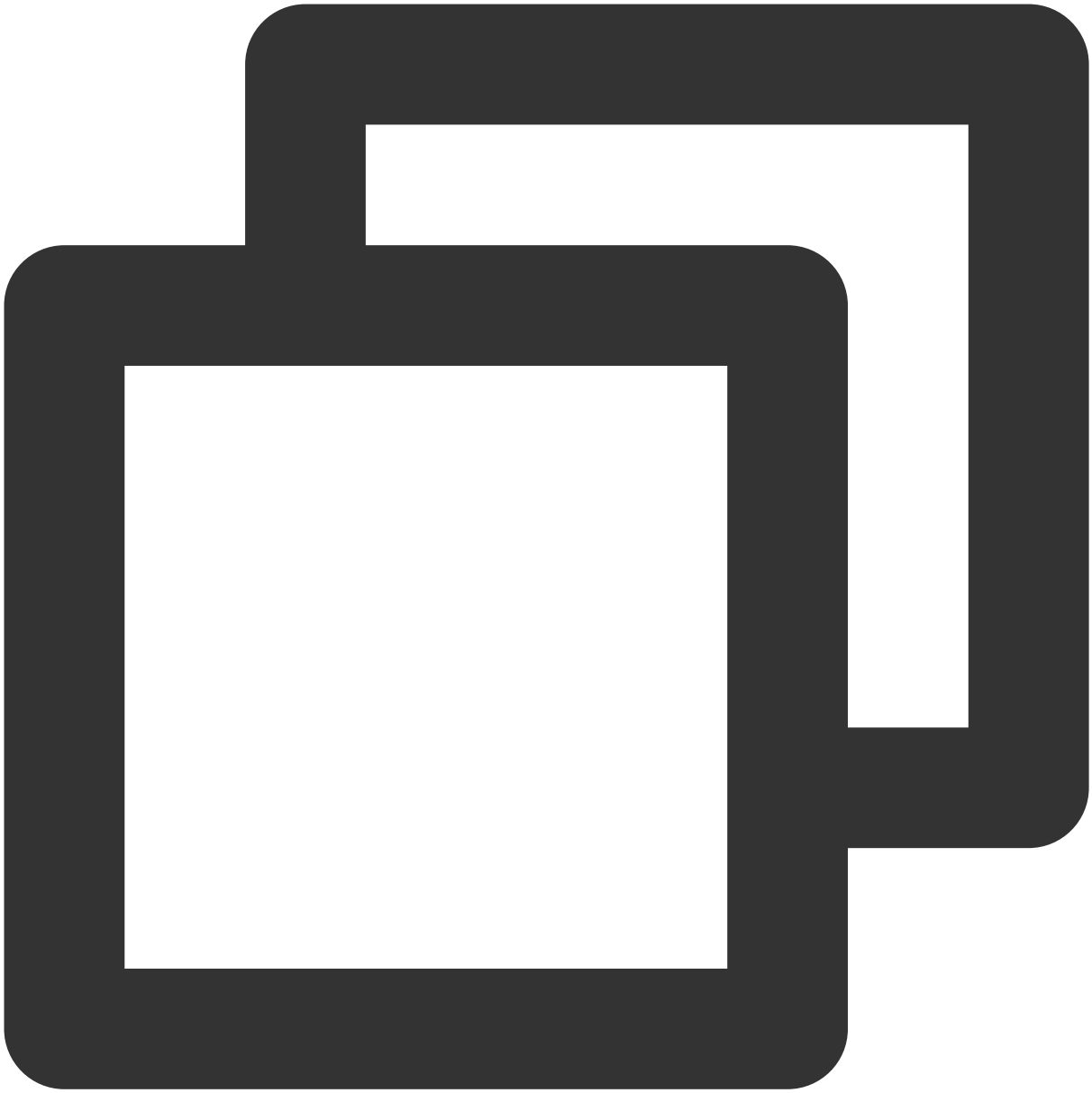
详情请参见 提交哈希值计算任务。

查询哈希值计算结果

功能说明

查询一个文件处理任务，根据任务 ID 查询任务详情。

示例代码



```
function getFileHashTask() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var jobId = 'xxxxxx'; // 提交文件哈希值计算任务后会返回当前任务的 jobId
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/file_jobs/' +
  var url = 'https://' + host;
  cos.request({
    Method: 'GET',
```

```
    Key: 'file_jobs/' + jobId,
    Url: url,
  },
  function(err, data){
      console.log(err || data);
  });
}
getFileHashTask();
```

参数说明

参数名称	描述	类型	是否必选
jobId	要查询的任务 ID	String	是

返回结果说明

详情请参见 查询哈希值计算结果。

Node.js SDK
快速入门

最近更新时间：2024-03-13 10:15:16

下载与安装

相关资源

对象存储服务的 XML JS SDK 资源 github 地址：XML Node.js SDK。
SDK 快速下载地址：XML Node.js SDK。
演示示例 Demo 下载地址：XML Node.js SDK Demo。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
说明：
如果您在使用 SDK 时遇到函数或方法不存在等错误，请先将 SDK 升级到最新版再重试。

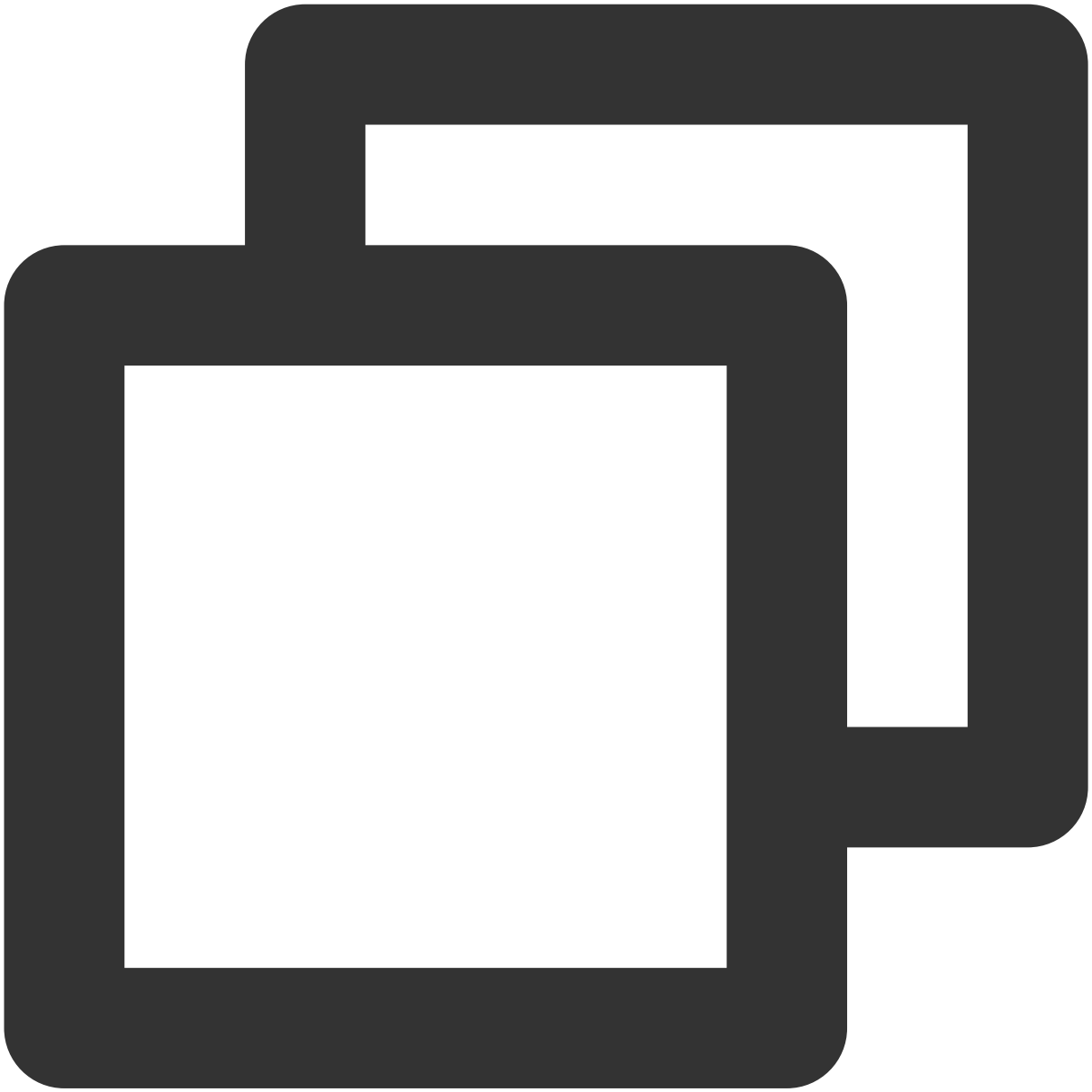
环境依赖

1. 使用 SDK 需要您的运行环境包含 nodejs 以及 npm，其中 nodejs 版本要求 ≥ 6。
2. 登录 对象存储控制台 创建存储桶后，获取存储桶名称和 地域名称。
3. 登录 访问管理控制台 获取您的项目 SecretId 和 SecretKey。

说明：
关于本文中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。

安装 SDK

通过 npm 安装环境 SDK： npm 地址。



```
npm i cos-nodejs-sdk-v5 --save
```

开始使用

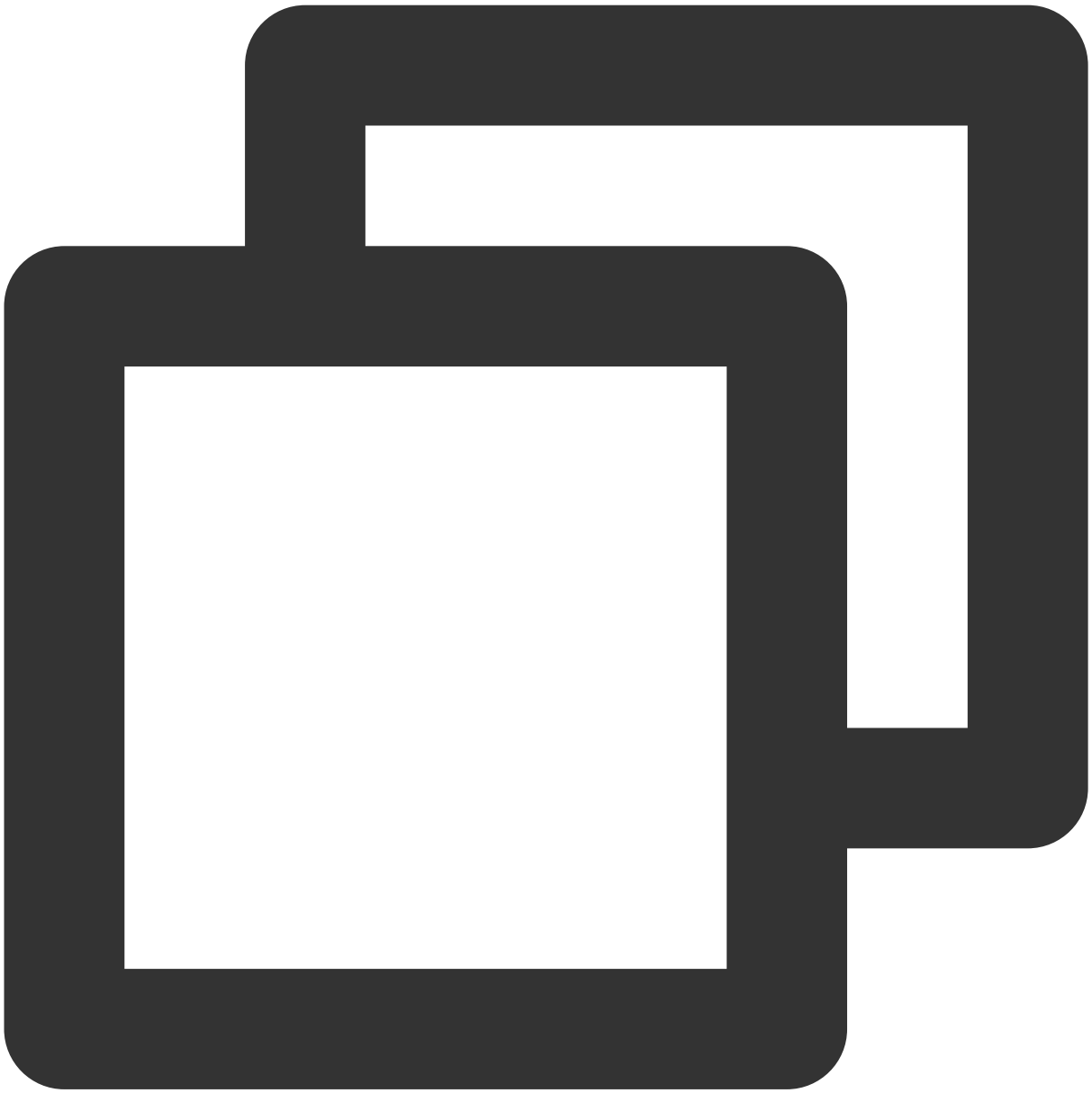
初始化

注意：

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。
如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

使用临时密钥初始化(推荐)

临时密钥生成和使用请参见 临时密钥生成及使用指引。Node.js SDK 支持通过传入临时密钥进行初始化，请参考以下示例代码：



```
var request = require('request');
var COS = require('cos-nodejs-sdk-v5');
```

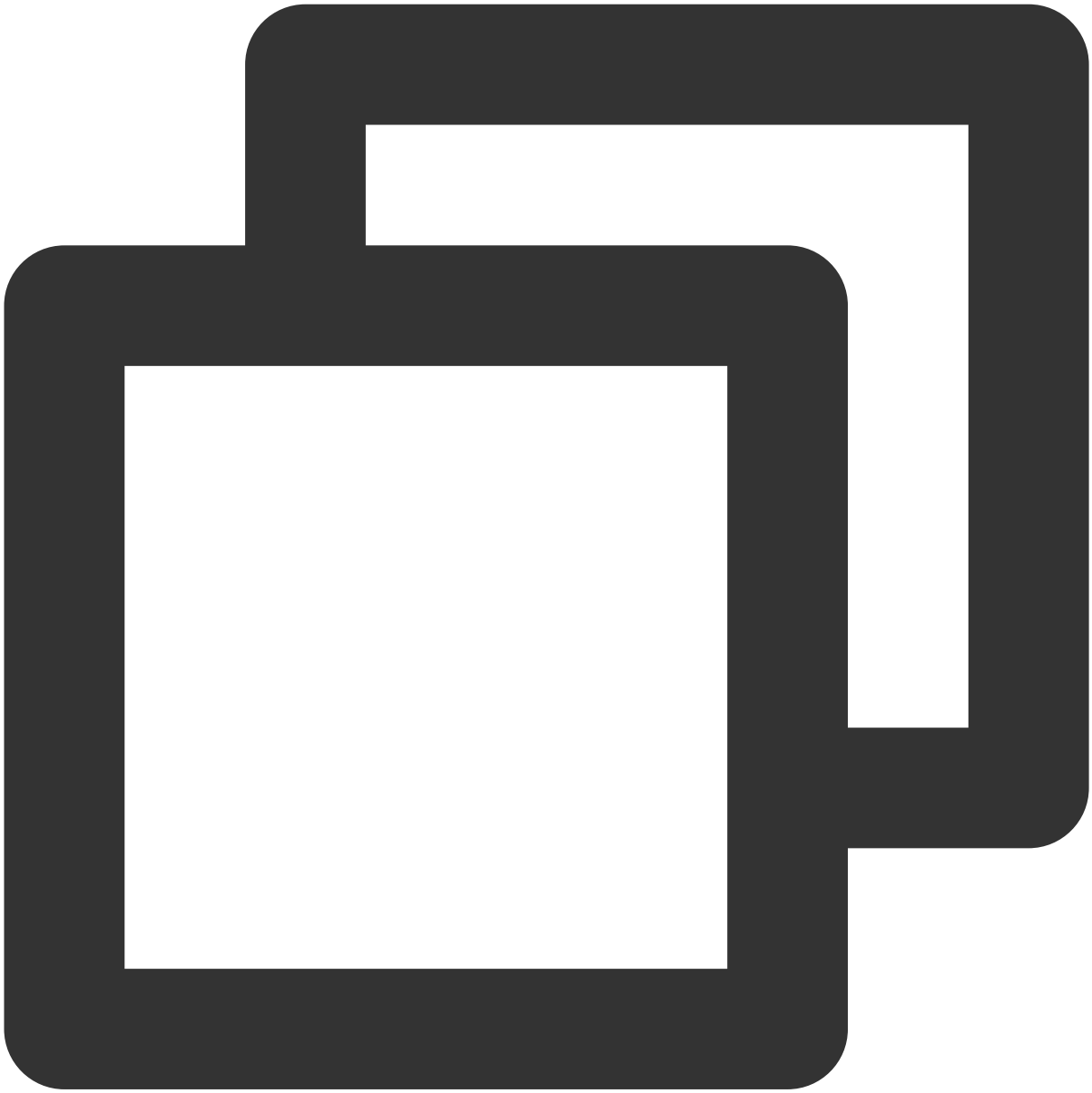
```
var cos = new COS({
    getAuthorization: function (options, callback) {
        // 初始化时不会调用，只有调用 cos 方法（例如 cos.putObject）时才会进入
        // 异步获取临时密钥
        request({
            url: 'https://example.com/sts',
            data: {
                // 可从 options 取需要的参数
            }
        }, function (err, response, body) {
            try {
                var data = JSON.parse(body);
                var credentials = data.credentials;
            } catch(e) {}
            if (!data || !credentials) return console.error('credentials invalid');
            callback({
                TmpSecretId: credentials.tmpSecretId,         // 临时密钥的 tmpSecretI
                TmpSecretKey: credentials.tmpSecretKey,       // 临时密钥的 tmpSecretK
                SecurityToken: credentials.sessionToken, // 临时密钥的 sessionToken
                ExpiredTime: data.expiredTime,                // 临时密钥失效时间戳，是
            });
        });
    }
});
```

使用永久密钥初始化(不推荐)

使用永久密钥初始化时，请注意保管好密钥，防止泄露。

请先在访问管理控制台中的 API 密钥管理 页面获取 SecretId、SecretKey。

将 SecretId、SecretKey、Bucket 和 Region 修改为您实际开发环境下的值，测试上传文件，请参考以下示例代码：



```
// SECRETID 和 SECRETKEY 请登录 https://console.tencentcloud.com/cam/capi 进行查看和管
var COS = require('cos-nodejs-sdk-v5');
var cos = new COS({
    SecretId: process.env.SecretId, // 推荐使用环境变量获取；用户的 SecretId，建议使用子账
    SecretKey: process.env.SecretKey, // 推荐使用环境变量获取；用户的 SecretKey，建议使用
});
```

以下是部分常用接口例子，更详细的初始化方法请参见 demo 示例。

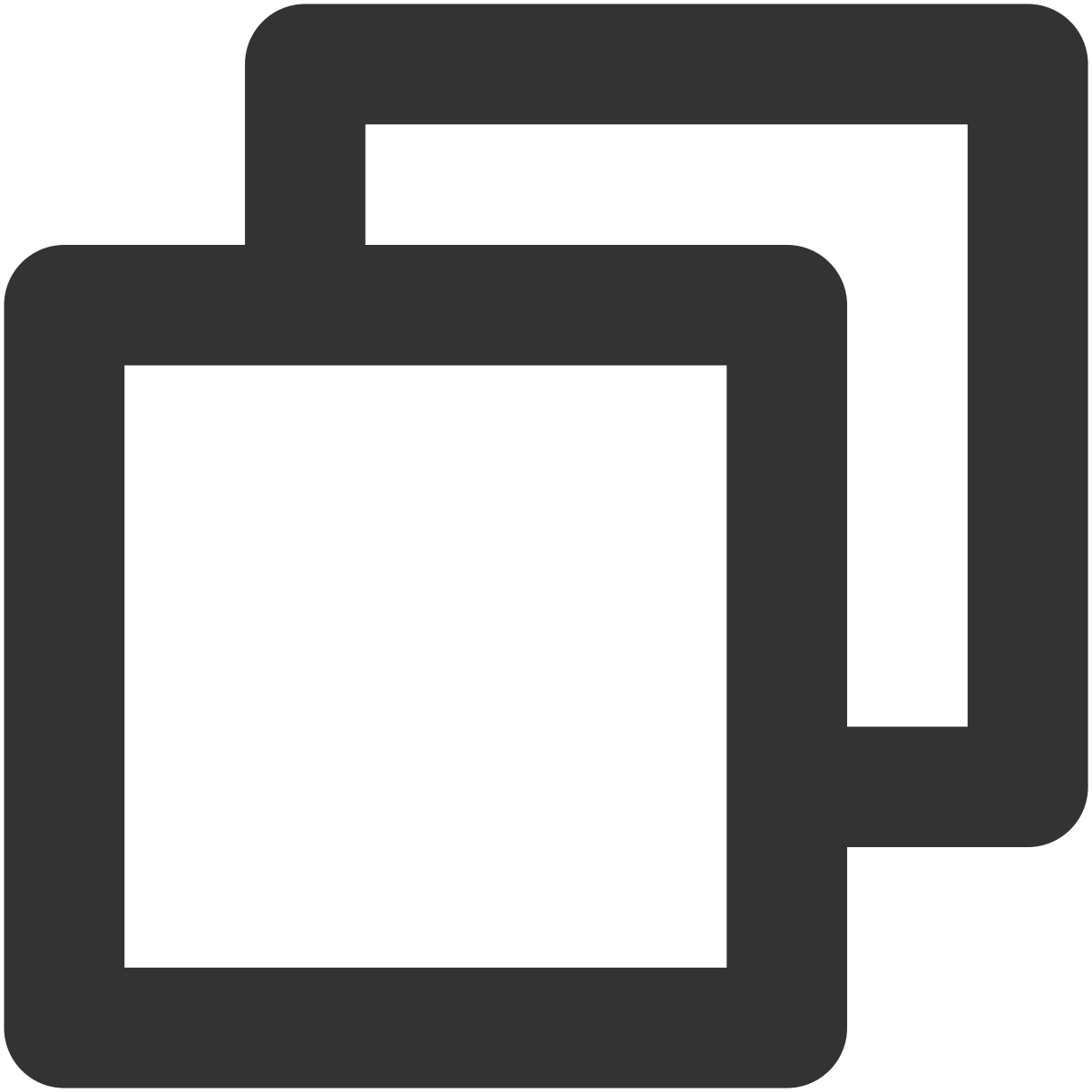
配置项

构造函数参数说明

参数名	参数描述	类型	是否必填
SecretId	用户的 SecretId	String	是
SecretKey	用户的 SecretKey	String	是
FileParallelLimit	同一个实例下上传的文件并发数，默认值3	Number	否
ChunkParallelLimit	同一个上传文件的分块并发数，默认值3	Number	否
ChunkRetryTimes	分块上传及分块复制时，出错重试次数，默认值2（加第一次，请求共3次）	Number	否
ChunkSize	分块上传时，每块的字节数大小，单位为 Byte，默认值1048576（1MB）	Number	否
SliceSize	使用 uploadFiles 批量上传时，文件大小大于该数值将使用按分片上传，否则将调用简单上传，单位 Byte，默认值1048576（1MB）	Number	否
CopyChunkParallelLimit	进行分块复制操作中复制分块上传的并发数，默认值20	Number	否
CopyChunkSize	使用 sliceCopyFile 分块复制文件时，每片的大小字节数，单位为 Byte，默认值10485760（10MB）	Number	否
CopySliceSize	使用 sliceCopyFile 分片复制文件时，文件大小大于该数值将使用分片复制，否则将调用简单复制，单位为 Byte，默认值10485760（10MB）	Number	否
ProgressInterval	上传进度的回调方法 onProgress 的回调频率，单位 ms，默认值1000	Number	否
Protocol	发请求时用的协议，可选项 `https:` 、 `http:` ，默认判断当前页面是 `http:` 时使用 `http:` ，否则使用 `https:`	String	否
ServiceDomain	调用 getService 方法时，请求的域名，例如 `service.cos.myqcloud.com`	String	否
Domain	调用操作存储桶和对象的 API 时自定义请求域名。可以使用模板， 例如 `"{Bucket}.cos.{Region}.myqcloud.com"` ，即在调用 API 时会使用参数中传入的 Bucket 和 Region 进行替换	String	否
UploadQueueSize	上传队列最长大小，超出队列大小并失败/已完成/已取消	Number	否

	状态的任务会被清理，默认1000		
ForcePathStyle	强制使用后缀式模式发请求。后缀式模式中 Bucket 会放在域名后的 pathname 里，并且 Bucket 会加入签名 pathname 计算，默认 false	Boolean	否
UploadCheckContentMd5	强制上传文件也校验 Content-MD5，会对文件请求 Body 计算 md5 放在 header 的 Content-MD5 字段里，默认 false	Boolean	否
Timeout	超时时间，单位毫秒，默认为0，即不设置超时时间	Number	否
KeepAlive	多个请求同用 TCP 连接，默认 true，若请求并发量大建议 打开	Boolean	否
StrictSsl	严格校验 HTTPS 证书，默认 true	Boolean	否
Proxy	请求时使用 HTTP 代理，例如：`http://127.0.0.1:8080`	String	否
getAuthorization	获取签名的回调方法，如果没有 SecretId、SecretKey 时，这个参数必选	Function	否
UseAccelerate	是否启用全球加速域名，默认为 false。若改为 true，需要存储桶开启全球加速功能，详情请参见 开启全球加速。	Boolean	否

getAuthorization 回调函数说明的函数说明（使用格式一）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的函数说明：

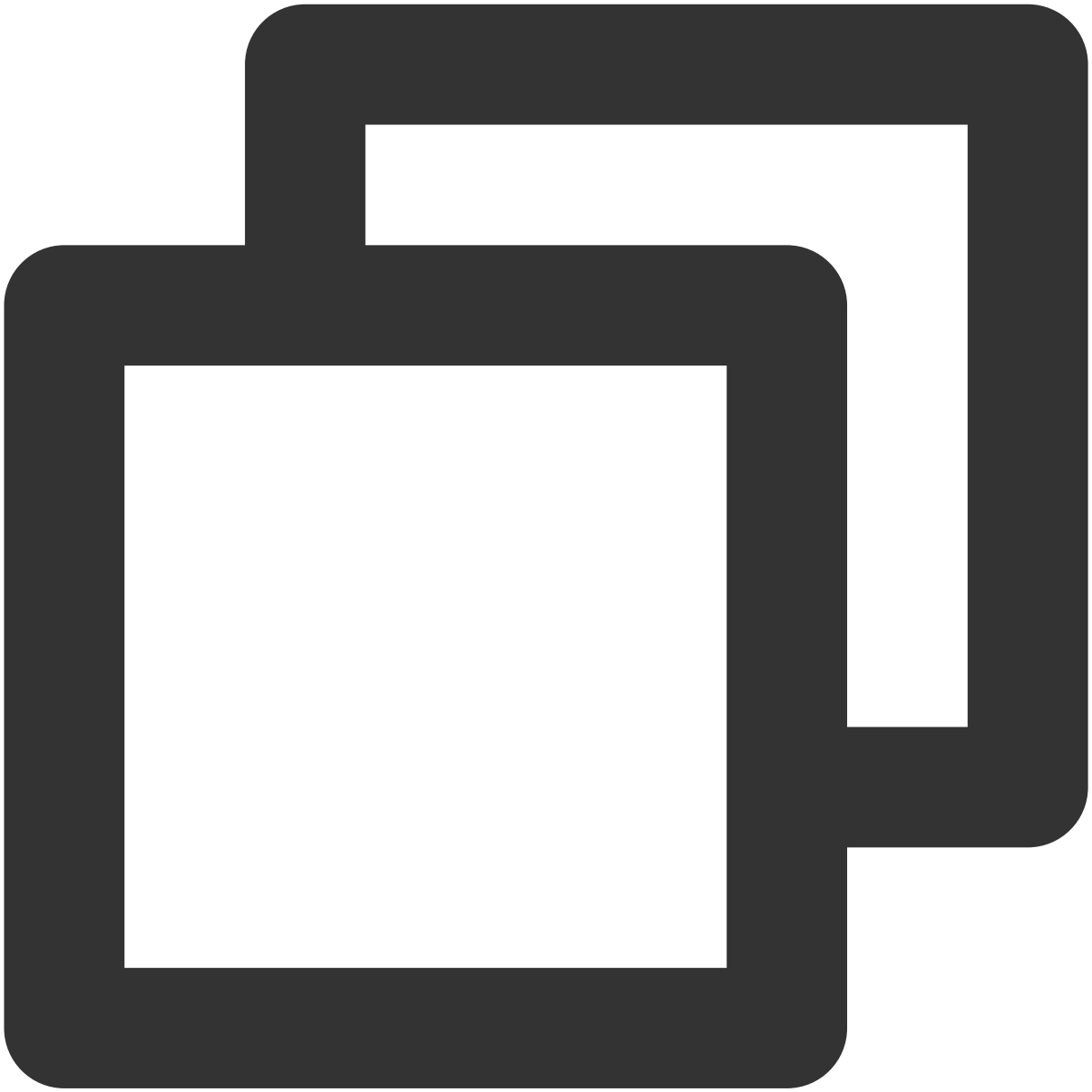
参数名	参数描述	类型
options	获取临时密钥需要的参数对象	Object
- Bucket	存储桶的名称，命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String

- Region	存储桶所在地域，枚举值请参见 地域和访问域名	String
callback	临时密钥获取完成后的回传方法	Function

获取完临时密钥后，callback 回传一个对象，回传对象的属性列表如下：

属性名	参数描述	类型	是否必填
TmpSecretId	获取回来的临时密钥的 tmpSecretId	String	是
TmpSecretKey	获取回来的临时密钥的 tmpSecretKey	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段	String	是
StartTime	密钥获取的开始时间，即获取时刻的时间戳，单位秒，startTime，如：1580000000，用于签名开始时间，传入该参数可避免前端时间偏差签名过期问题	String	否
ExpiredTime	获取回来的临时密钥的 expiredTime，超时时刻的时间戳，单位秒，如：1580000900	String	是

getAuthorization 回调函数说明（使用格式二）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的函数说明：

参数名	参数描述	类型
options	获取签名需要的参数对象	Object
- Method	当前请求的 Method	String
-	请求路径，用于签名计算	String

Pathname		
- Key	对象键（Object 的名称），对象在存储桶中的唯一标识，了解更多请参见 对象概述	String
- Query	当前请求的 query 参数对象，{key: 'val'} 的格式	Object
- Headers	当前请求的 header 参数对象，{key: 'val'} 的格式	Object
callback	临时密钥获取完成后的回调	Function

getAuthorization 计算完成后，callback 回传参数支持两种格式：
格式一：回传鉴权凭证字符串 Authorization。
格式二：回传一个对象，对象属性列表如下：

属性名	参数描述	类型	是否必填
Authorization	计算得到的签名字符串	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段	String	否

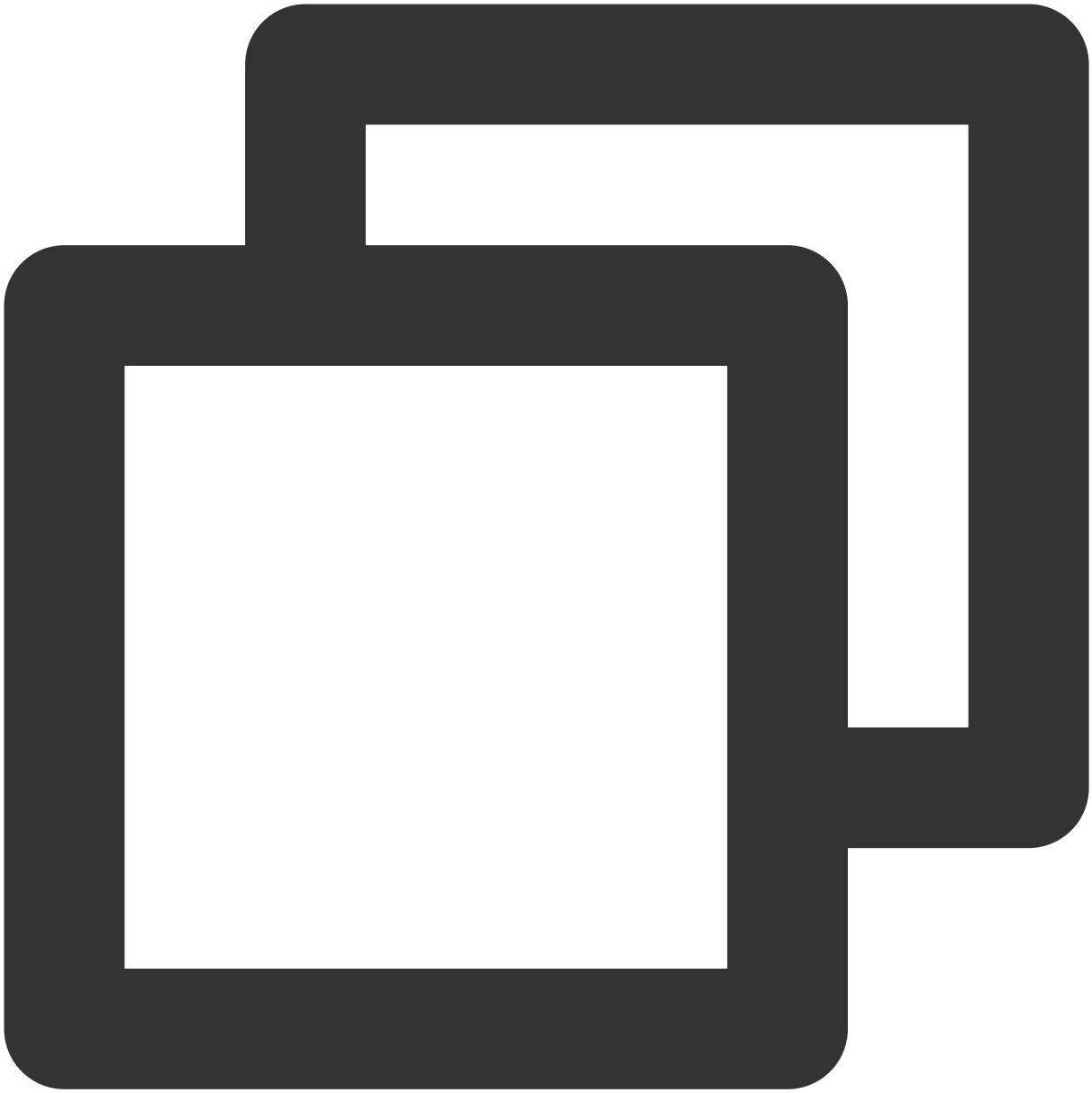
获取鉴权凭证

实例本身鉴权凭证可以通过实例化时传入的参数控制如何或获取，有三种获取方式：
1. 实例化时，传入 SecretId、SecretKey，每次需要签名都由实例内部计算。
2. 实例化时，传入 getAuthorization 回调，每次需要签名通过这个回调计算完返回签名给实例。
3. 实例化时，传入 getSTS 回调，每次需要临时密钥通过这个回调回去完返回给实例，在每次请求时实例内部使用临时密钥计算得到签名。

使用方式

回调方式

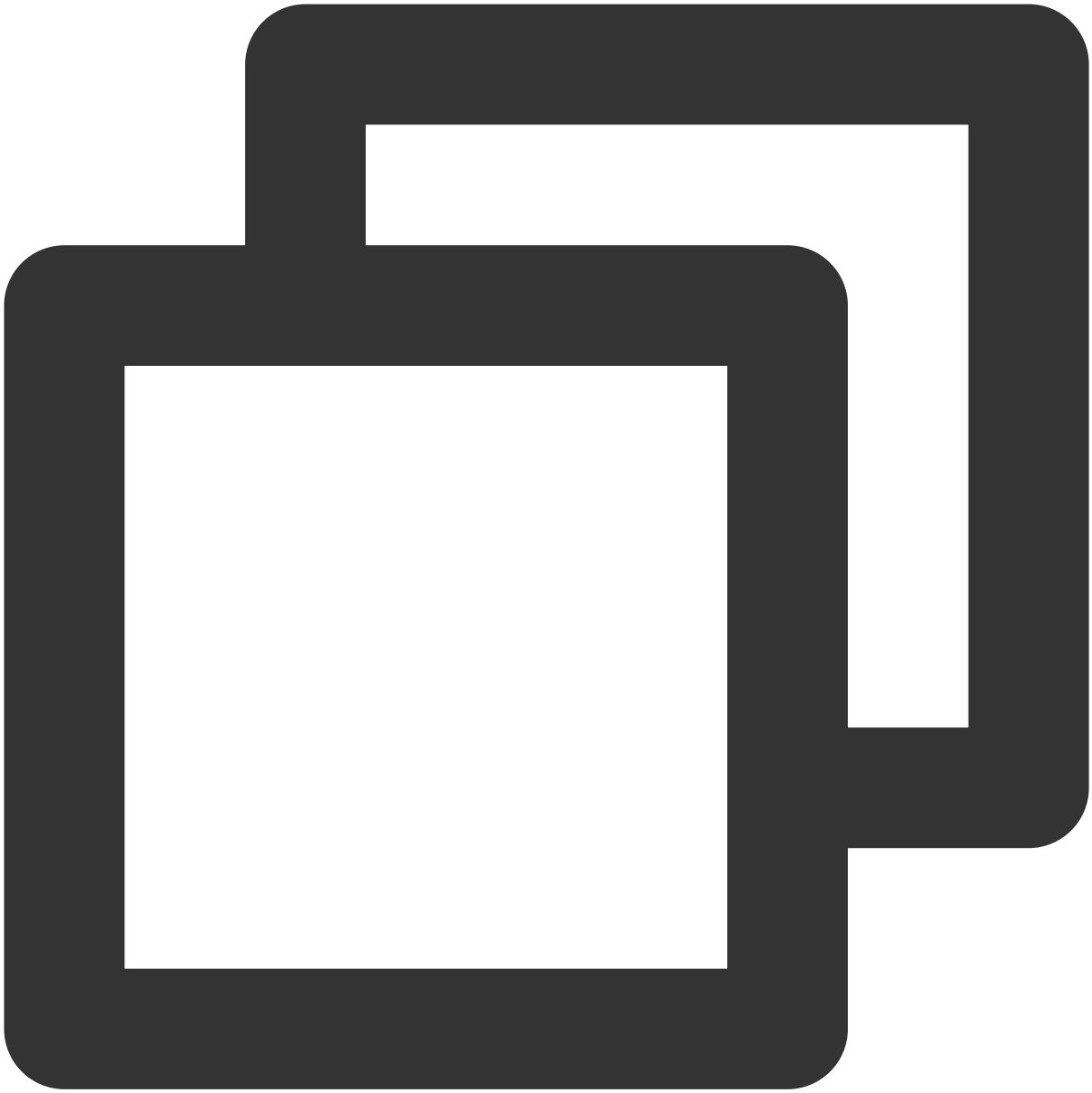
文档里默认使用回调方式，使用代码如下:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }, function(err, data) {
  if (err) {
    console.log('上传出错', err);
  } else {
    console.log('上传成功', data);
  }
});
```

Promise

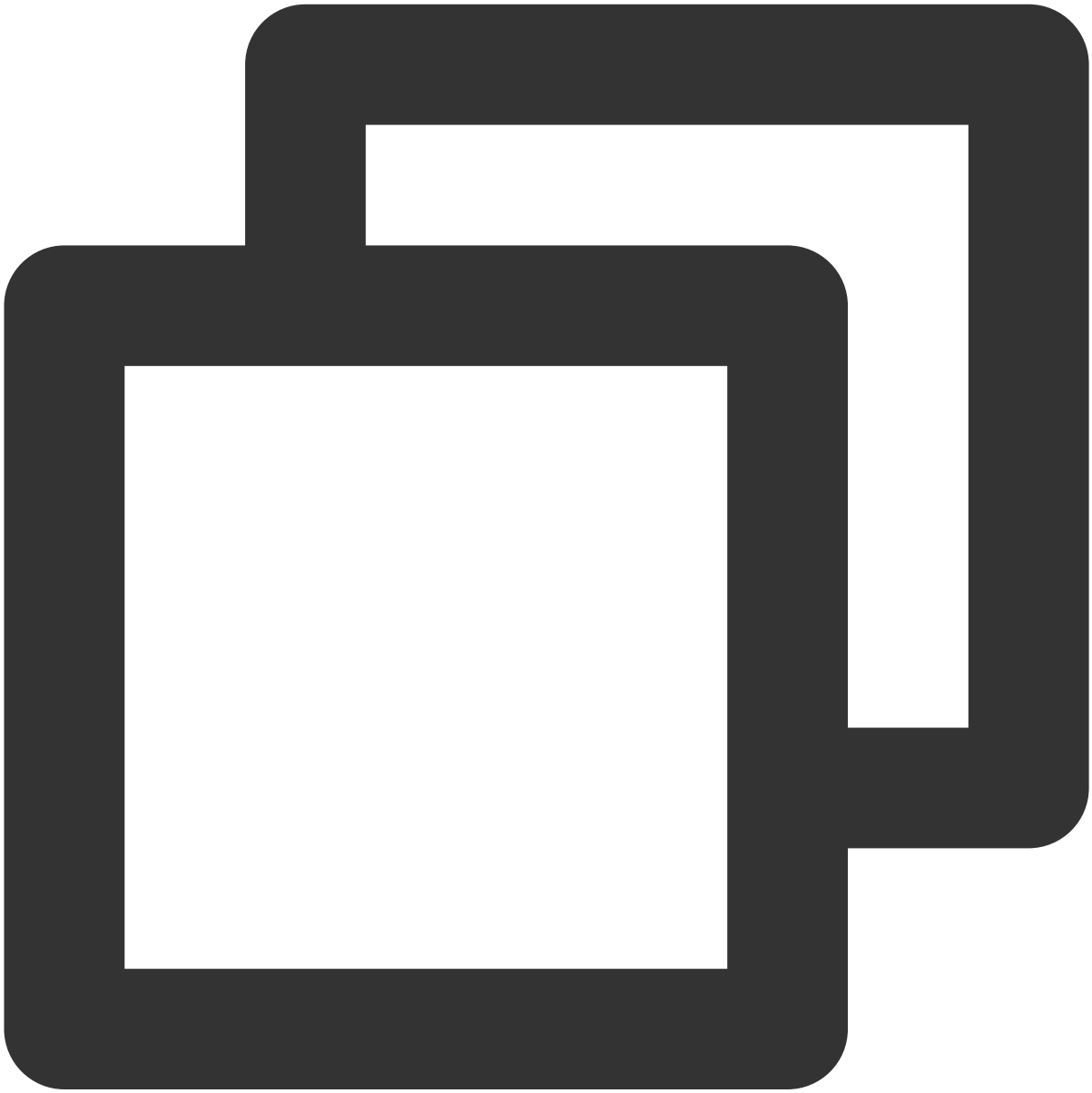
sdk同样支持 Promise 方式调用，例如上述回调方式的代码等同于以下代码:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }).then(data => {
  console.log('上传成功', data);
}).catch(err => {
  console.log('上传出错', err);
});
```

同步方式

同步方式基于 JavaScript 的 async 和 await，上述回调方式的代码等同于以下代码:



```
async function upload() {
  // 这里省略初始化过程和上传参数
  var cos = new COS({ ... });
  try {
    var data = await cos.uploadFile({ ... });
    return { err: null, data: data }
  } catch (err) {
    return { err: err, data: null };
```

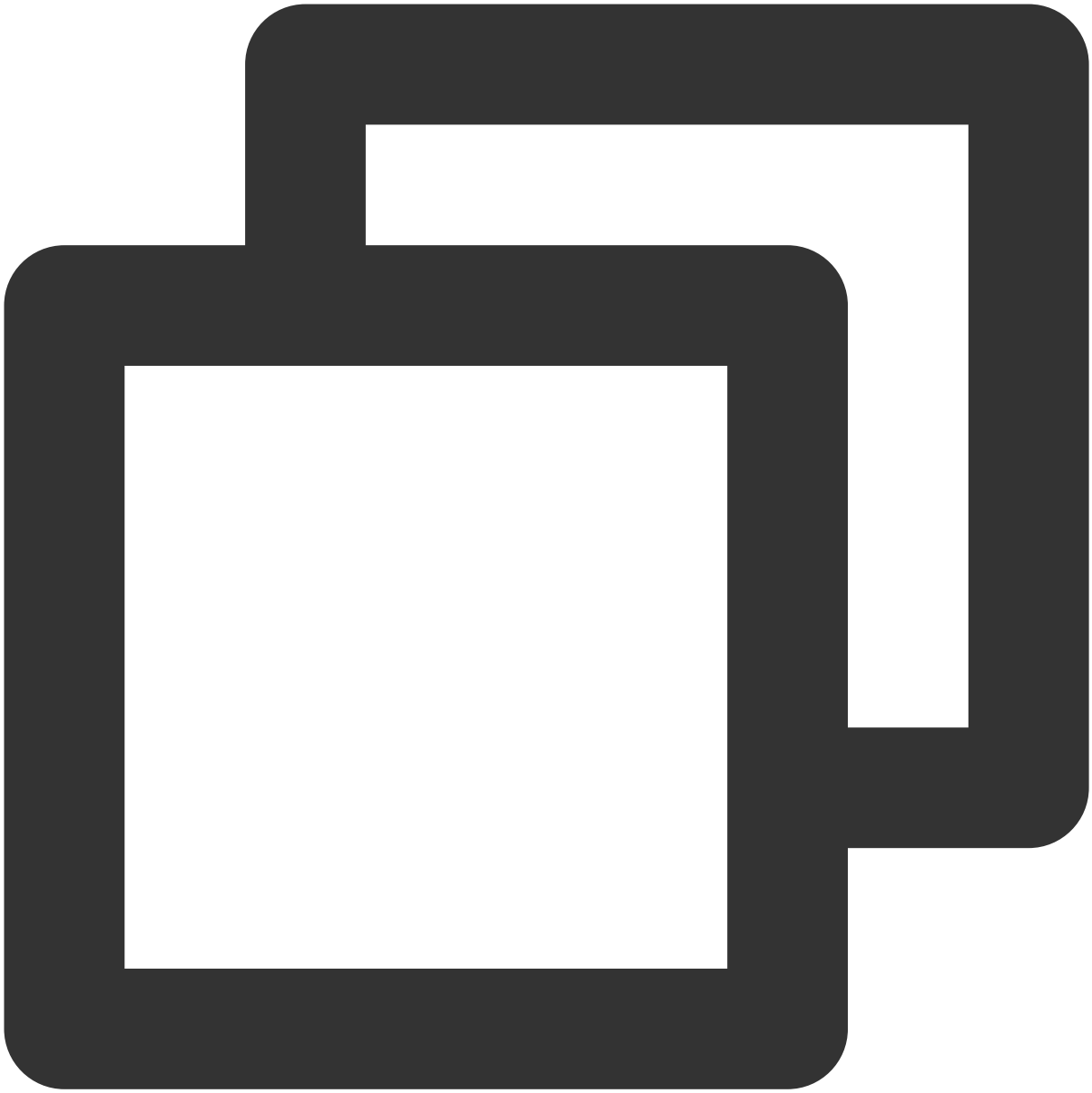
```
  }
}
// 可以同步拿到请求的返回值,这里举例说明,实际返回的数据格式可以自定义
var uploadResult = await upload();
if (uploadResult.err) {
  console.log('上传出错', uploadResult.err);
} else {
  console.log('上传成功', uploadResult.data);
}
```

注意：

cos.getObjectUrl 目前只支持回调方式。

使用技巧

通常情况下我们只需要创建一个 COS SDK 实例，然后在需要调用SDK方法的地方直接使用这个实例即可，示例代码如下：

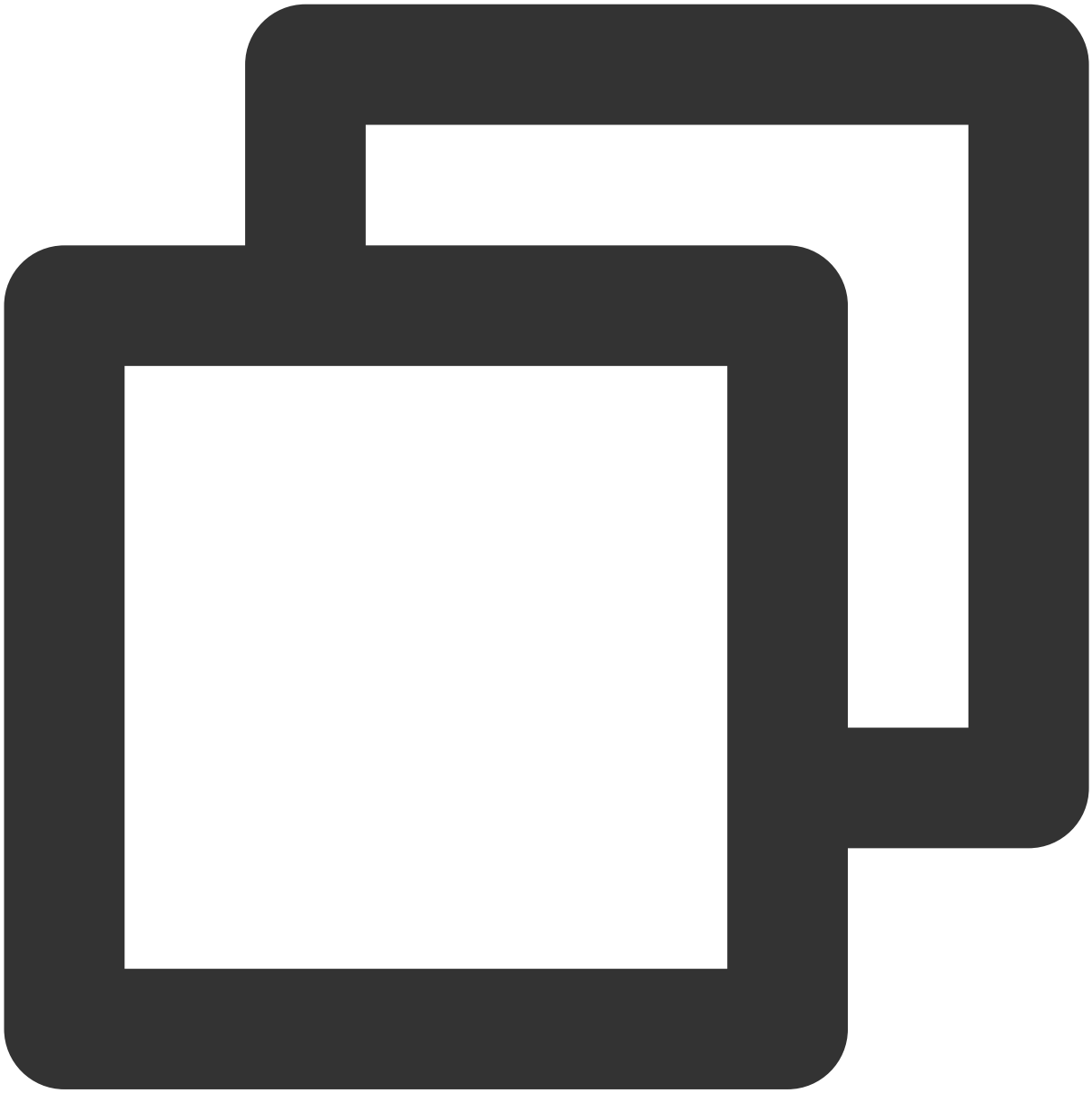


```
var cos = new COS({
  ....
});

/* 自己封装的上传方法 */
function myUpload() {
  // 不需要在每个方法里创建一个 COS SDK 实例
  // var cos = new COS({
  //   ...
  // });
  cos.putObject({
```

```
    ....
  });
}

/* 自己封装的删除方法 */
function myDelete() {
  // 不需要在每个方法里创建一个 COS SDK 实例
  // var cos = new COS({
  //   ...
  // });
  cos.deleteObject({
    ....
  });
}
```

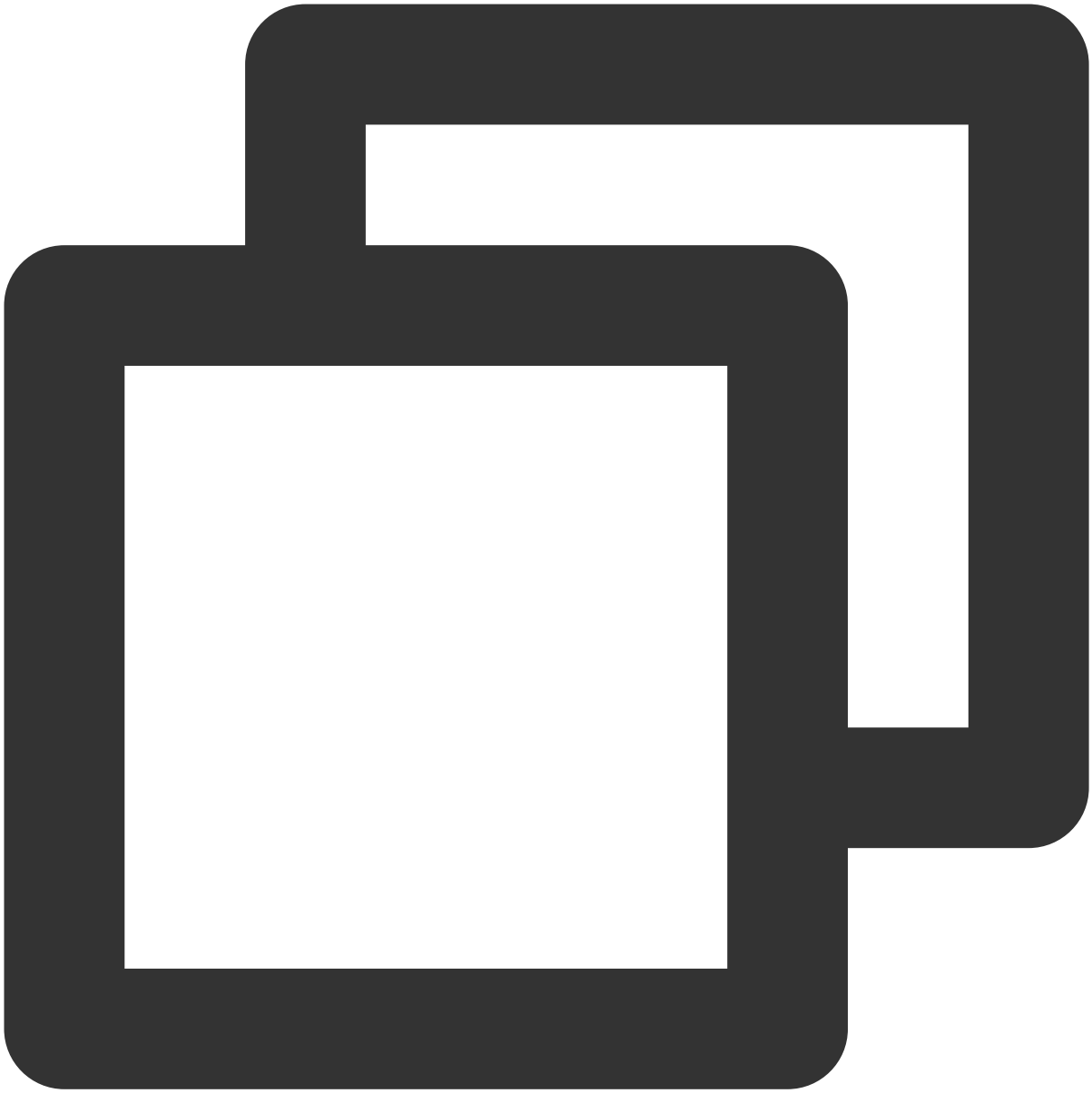
以下是部分常用接口入口，更详细的初始化方法请参见 demo 示例。

创建存储桶



```
cos.putBucket({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION'
}, function(err, data) {
    console.log(err || data);
});
```

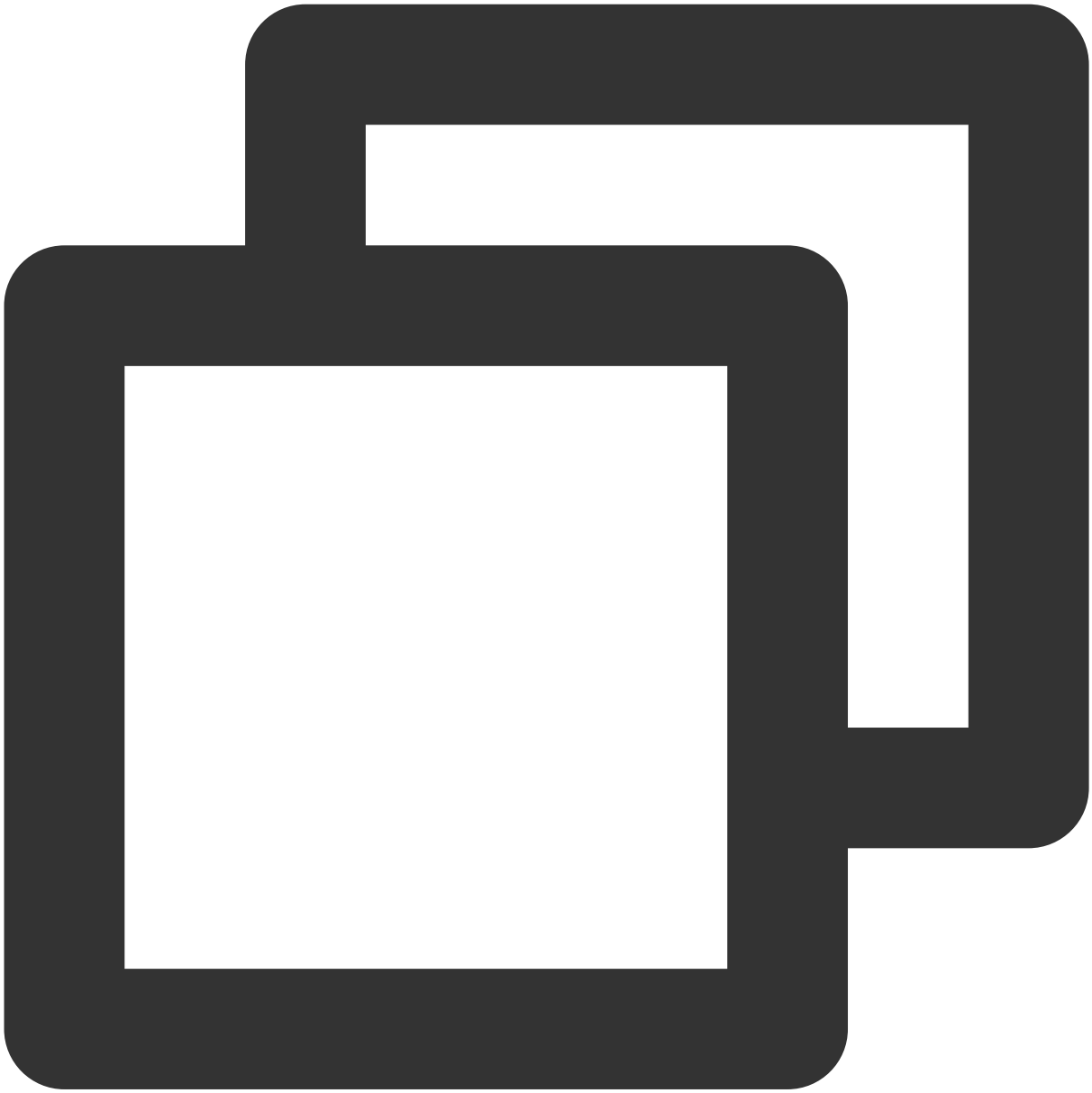
查询存储桶列表



```
cos.getService(function (err, data) {
    console.log(data && data.Buckets);
});
```

上传对象

该接口适用于小文件上传，大文件请使用分块上传接口，详情请参见 对象操作 文档。

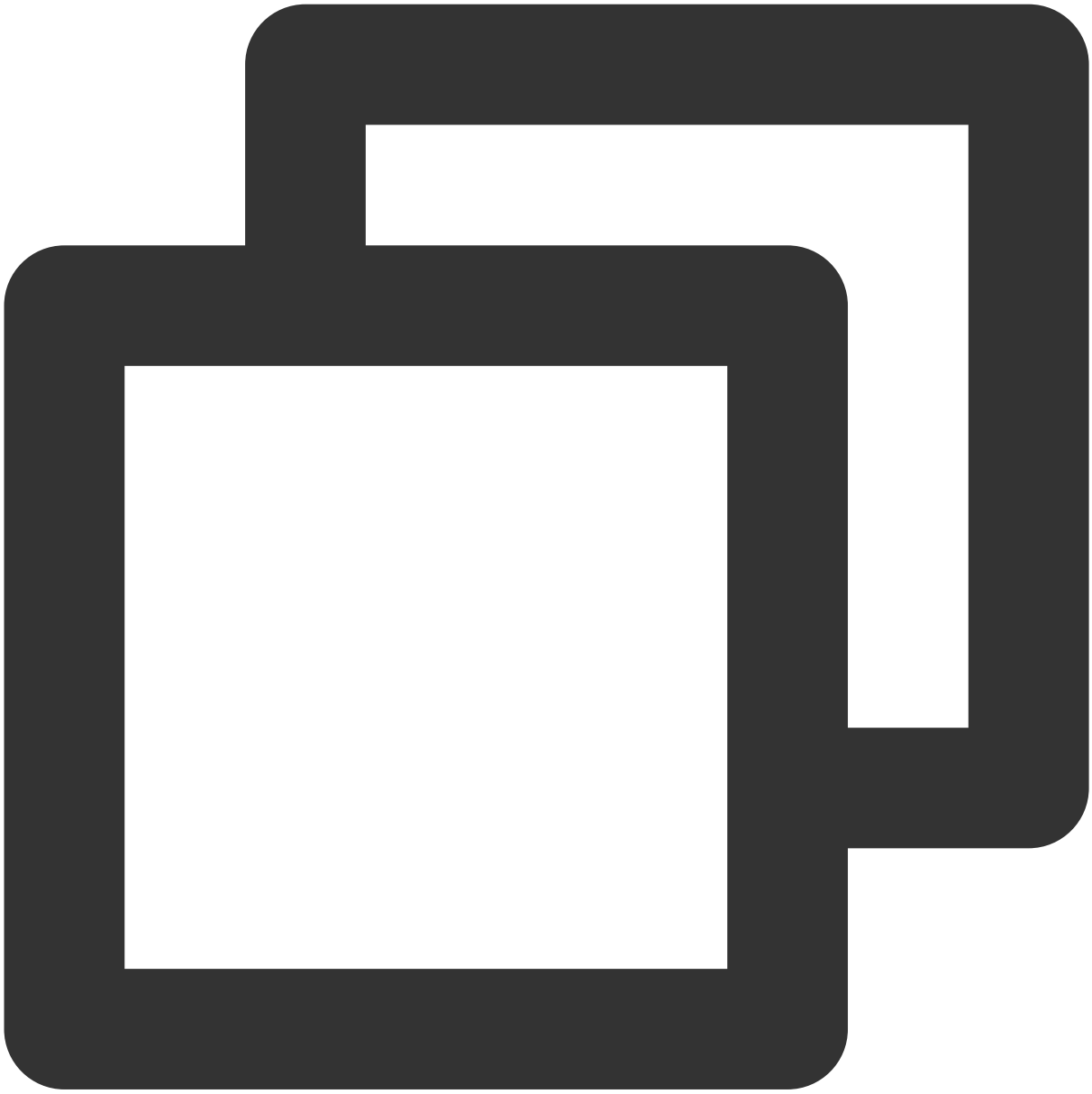


```
cos.putObject({
    Bucket: 'examplebucket-1250000000', /* 必须 */
    Region: 'COS_REGION',    /* 必须 */
    Key: 'exampleobject',              /* 必须 */
    StorageClass: 'STANDARD',
    Body: fs.createReadStream('./exampleobject'), // 上传文件对象
    onProgress: function(progressData) {
        console.log(JSON.stringify(progressData));
    }
}, function(err, data) {
    console.log(err || data);
```

版权所有：腾讯云计算（北京）有限责任公司

```
});
```

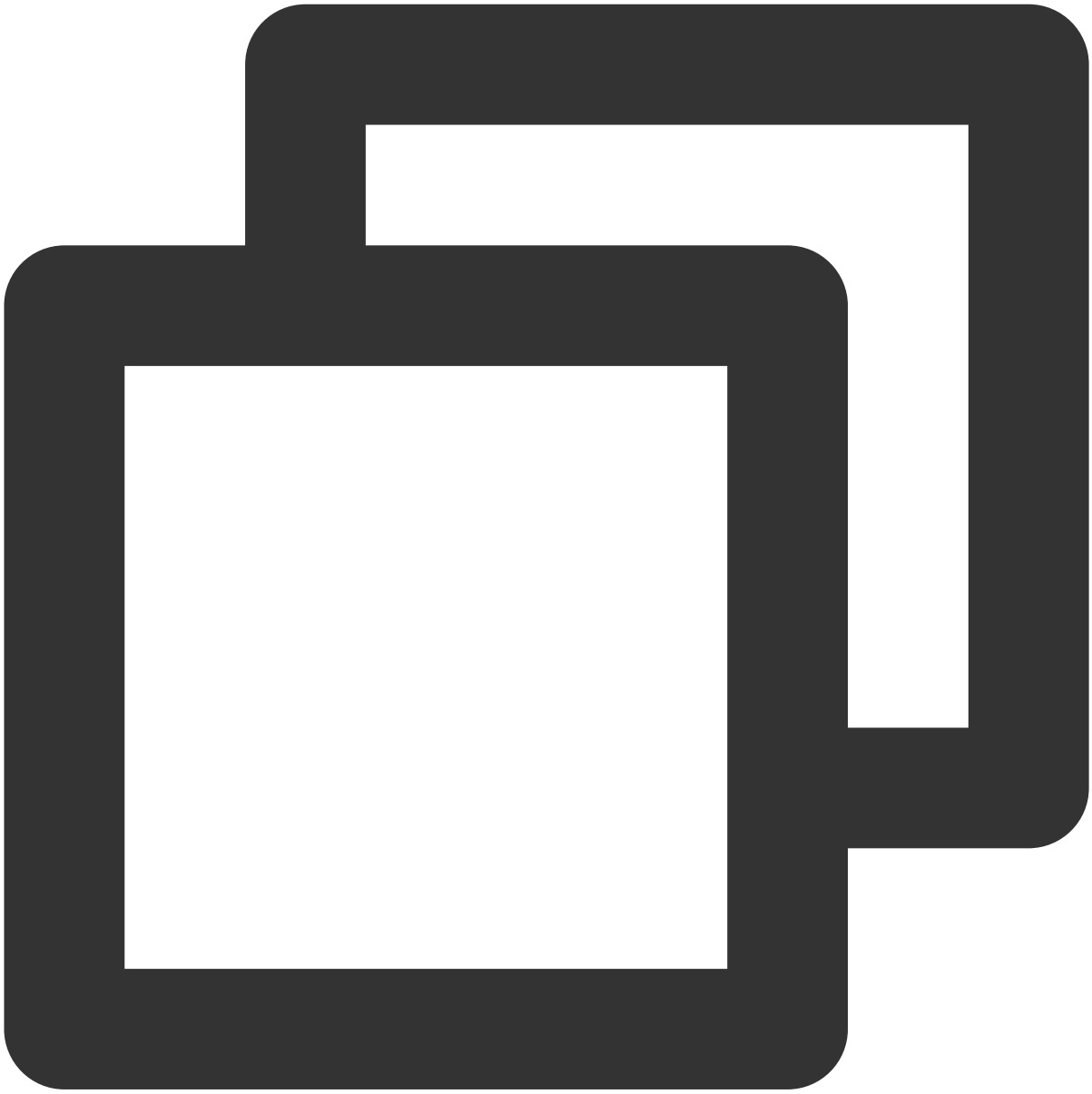
查询对象列表



```
cos.getBucket({
    Bucket: 'examplebucket-1250000000', /* 必须 */
    Region: 'COS_REGION',     /* 必须 */
    Prefix: 'a/',           /* 非必须 */
}, function(err, data) {
    console.log(err || data.Contents);
```

```
});
```

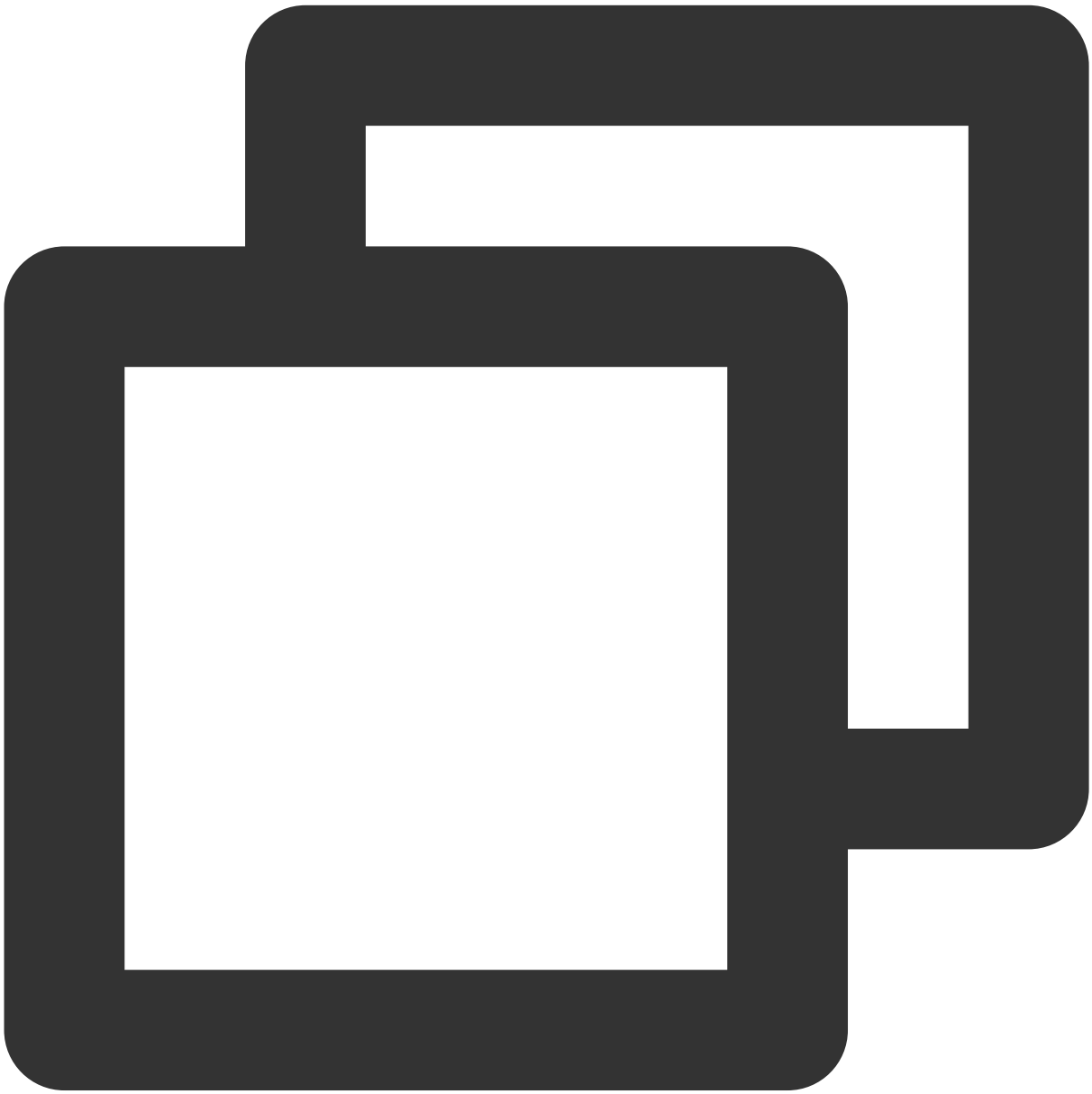
下载对象



```
cos.getObject({
    Bucket: 'examplebucket-1250000000', /* 必须 */
    Region: 'COS_REGION',     /* 必须 */
    Key: 'exampleobject',              /* 必须 */
    Output: fs.createWriteStream('./exampleobject'),
}, function(err, data) {
```

```
    console.log(err || data);
});
```

删除对象



```
cos.deleteObject({
    Bucket: 'examplebucket-1250000000', /* 必须 */
    Region: 'COS_REGION',    /* 必须 */
    Key: 'exampleobject'        /* 必须 */
}, function(err, data) {
```

```
    console.log(err || data);
});
```

图片处理
基础图片处理

最近更新时间：2024-03-13 10:15:17

简介

本文档提供关于基础图片处理的相关的 API 概览以及 SDK 示例代码。

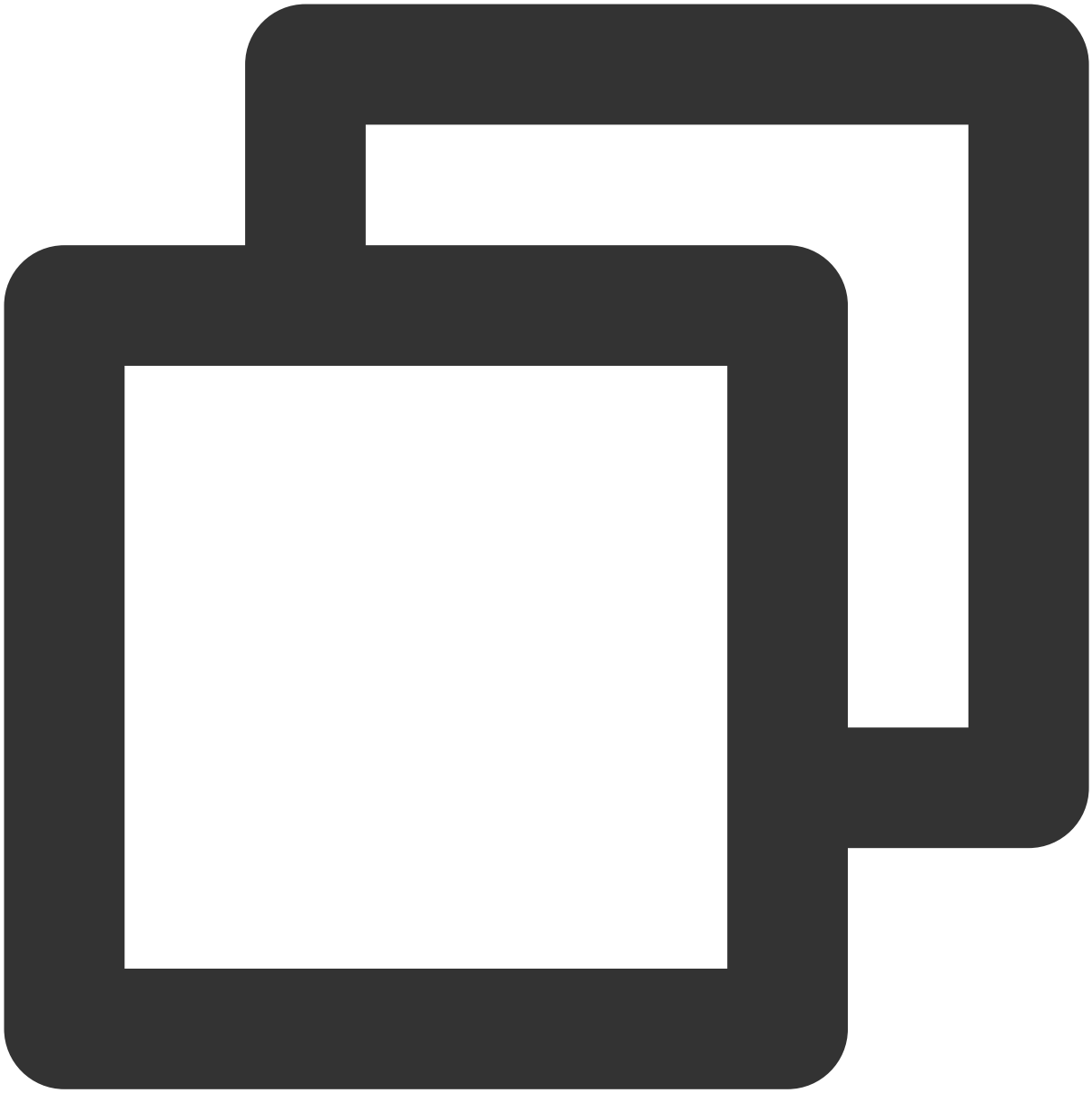
服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

上传时使用图片处理

下面示例展示了如何在上传图片时自动实现图片处理。
图片上传完成后，COS 会存储原始图片和已处理过的图片。后续用户可以通过普通的下载请求获取处理结果。

示例代码

版权所有：腾讯云计算（北京）有限责任公司



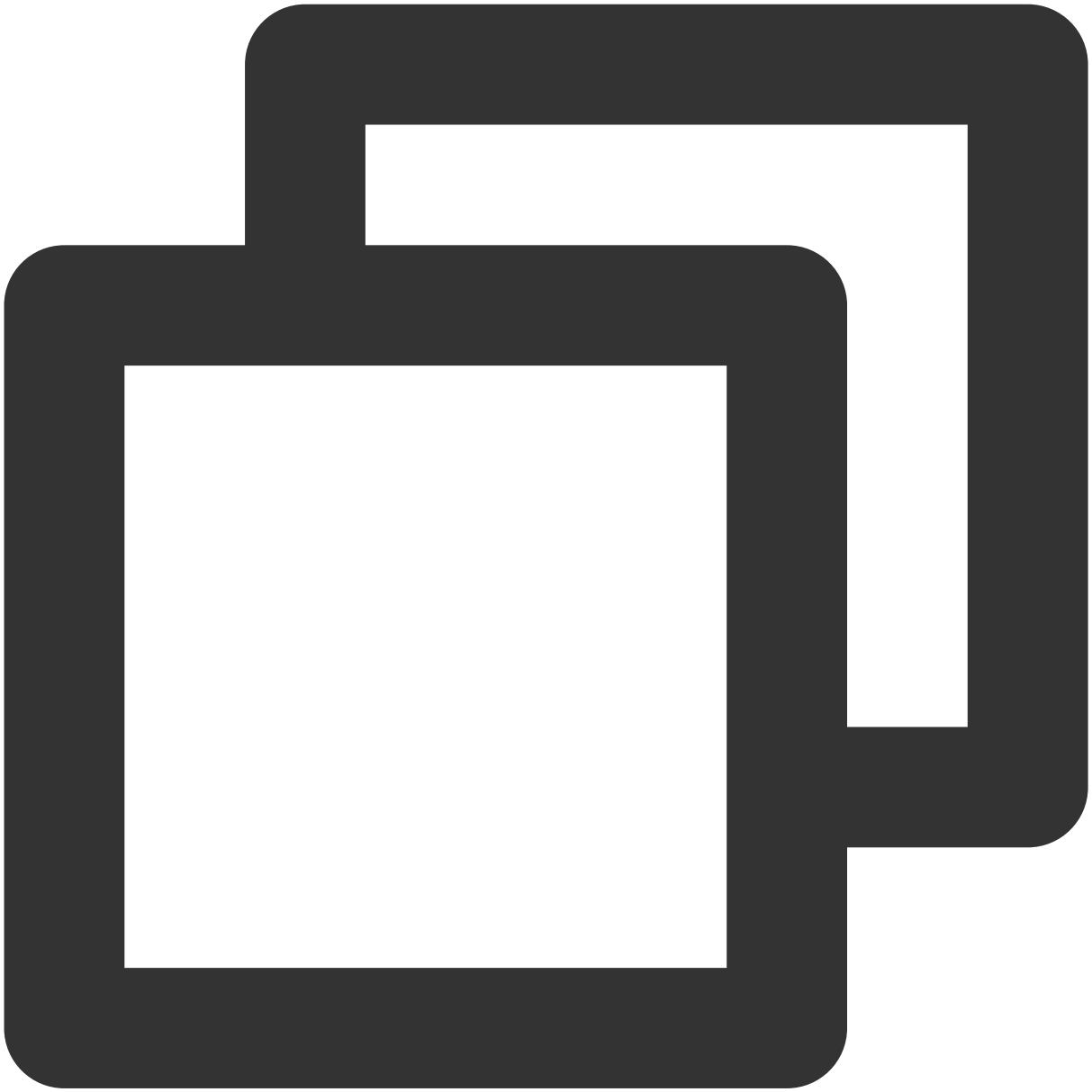
```
const filePath = "temp-file-to-upload" // 本地文件路径
cos.putObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Body: fs.createReadStream(filePath), // 上传文件对象
    onProgress: function(progressData) {
        console.log(JSON.stringify(progressData));
    },
    Headers: {
        // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
```

```
        'Pic-Operations': '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.j
    },
}, function(err, data) {
    console.log(err || data);
});
```

对云上数据进行图片处理

下面示例展示了如何在对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

示例代码



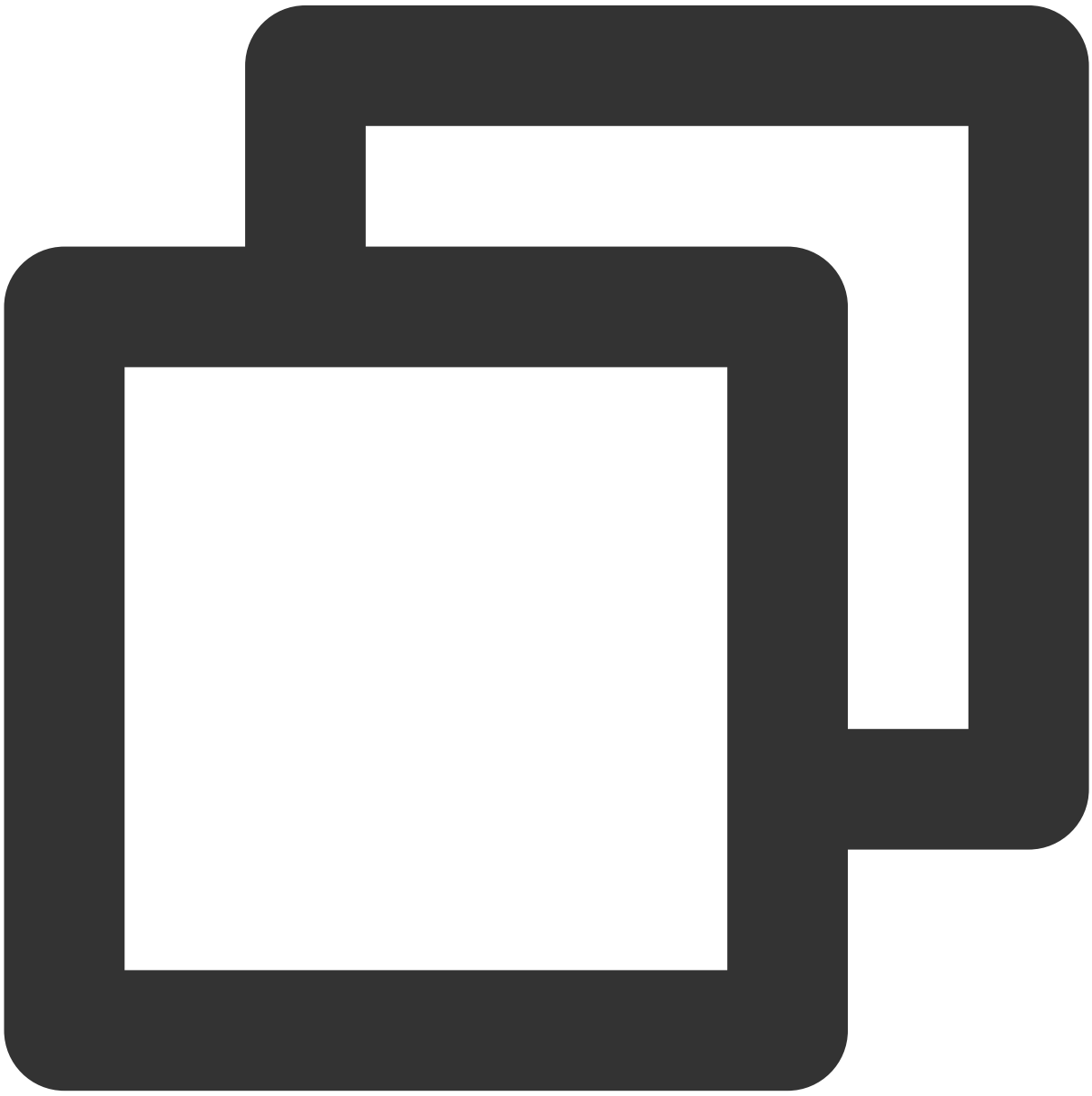
```
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Key: 'exampleobject',
    Method: 'POST',
    Action: 'image_process',
    Headers: {
        // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
      'Pic-Operations': '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jp
    },
}, function (err, data) {
```

```
    console.log(err || data);
});
```

下载时使用图片处理

下面示例展示了如何在下载图片时实现图片处理。

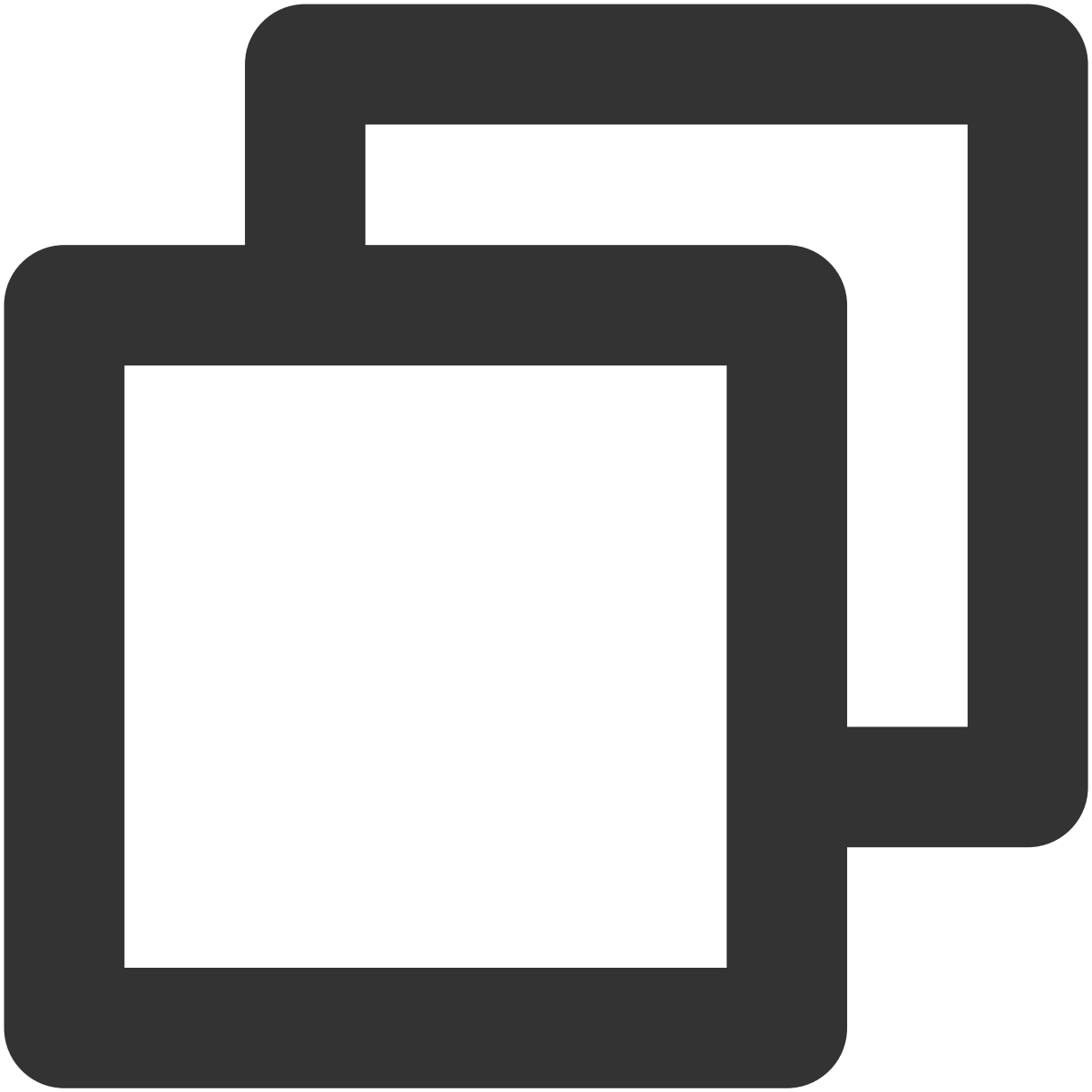
示例代码



```
cos.getObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    QueryString: `imageMogr2/thumbnail/200x/`,
}, function (err, data) {
   console.log(err || data);
   fs.writeFileSync('filepath', data.Body);  // 将图片内容保存本地
});
```

生成带图片处理参数的签名 URL

示例代码



```
// 生成带图片处理参数的文件签名 URL，过期时间设置为 30 分钟。
var url1 = cos.getObjectUrl({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Query: {
      `imageMogr2/thumbnail/200x/`: ''
    },
    Expires: 1800,
    Sign: true,
}, function (err, data) {
```

```
    if(data){
        console.log(data.Url);
    }
});

// 生成带图片处理参数的文件 URL，不带签名。
var url2 = cos.getObjectUrl({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Query: {
      `imageMogr2/thumbnail/200x/`: ''
    },
    Sign: false,
}, function (err, data) {
    if(data){
        console.log(data.Url);
    }
});
```

盲水印

最近更新时间：2024-03-13 10:15:16

简介

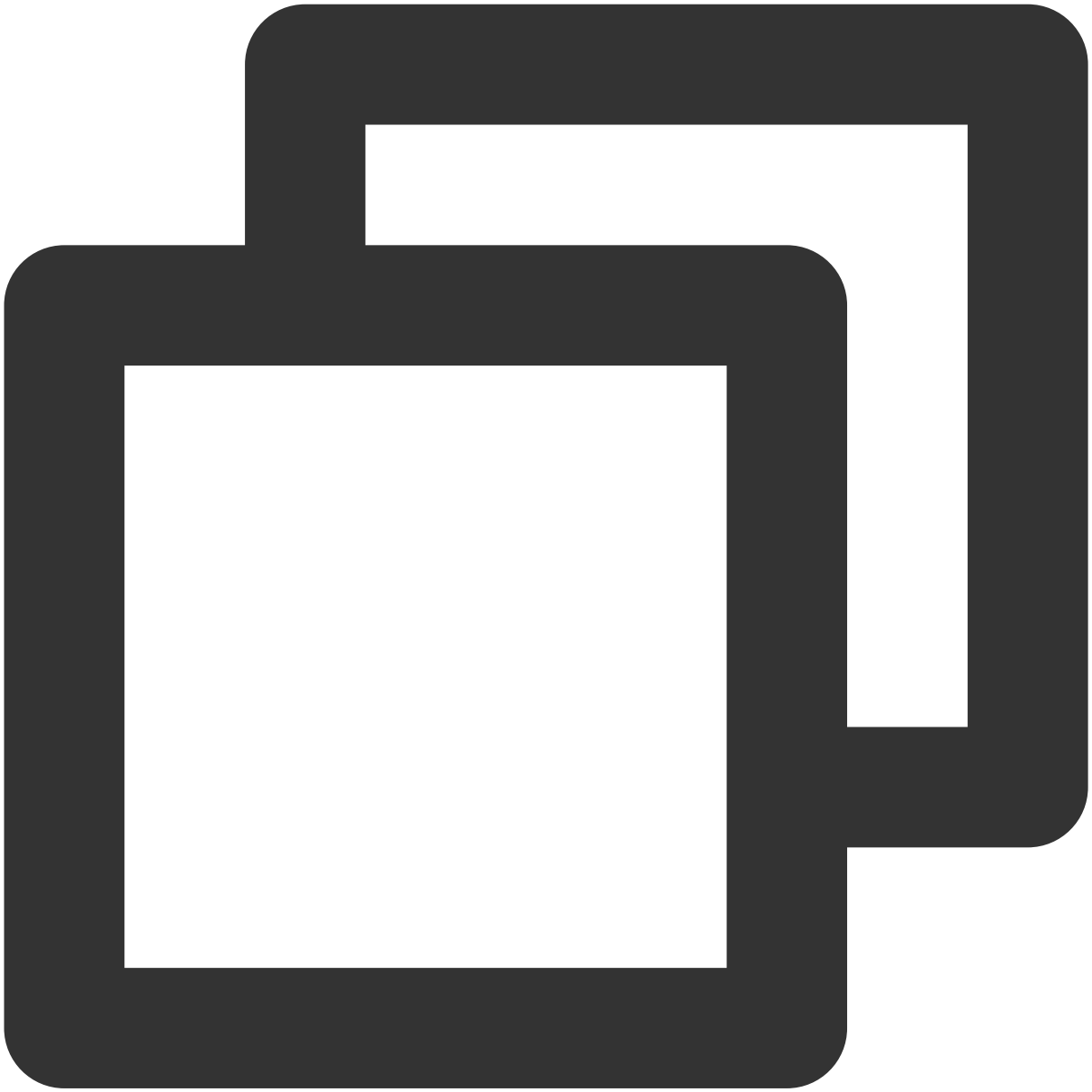
本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。
关于盲水印 API 文档请参见 盲水印。

添加盲水印

功能说明

盲水印支持在上传时添加和下载时添加。

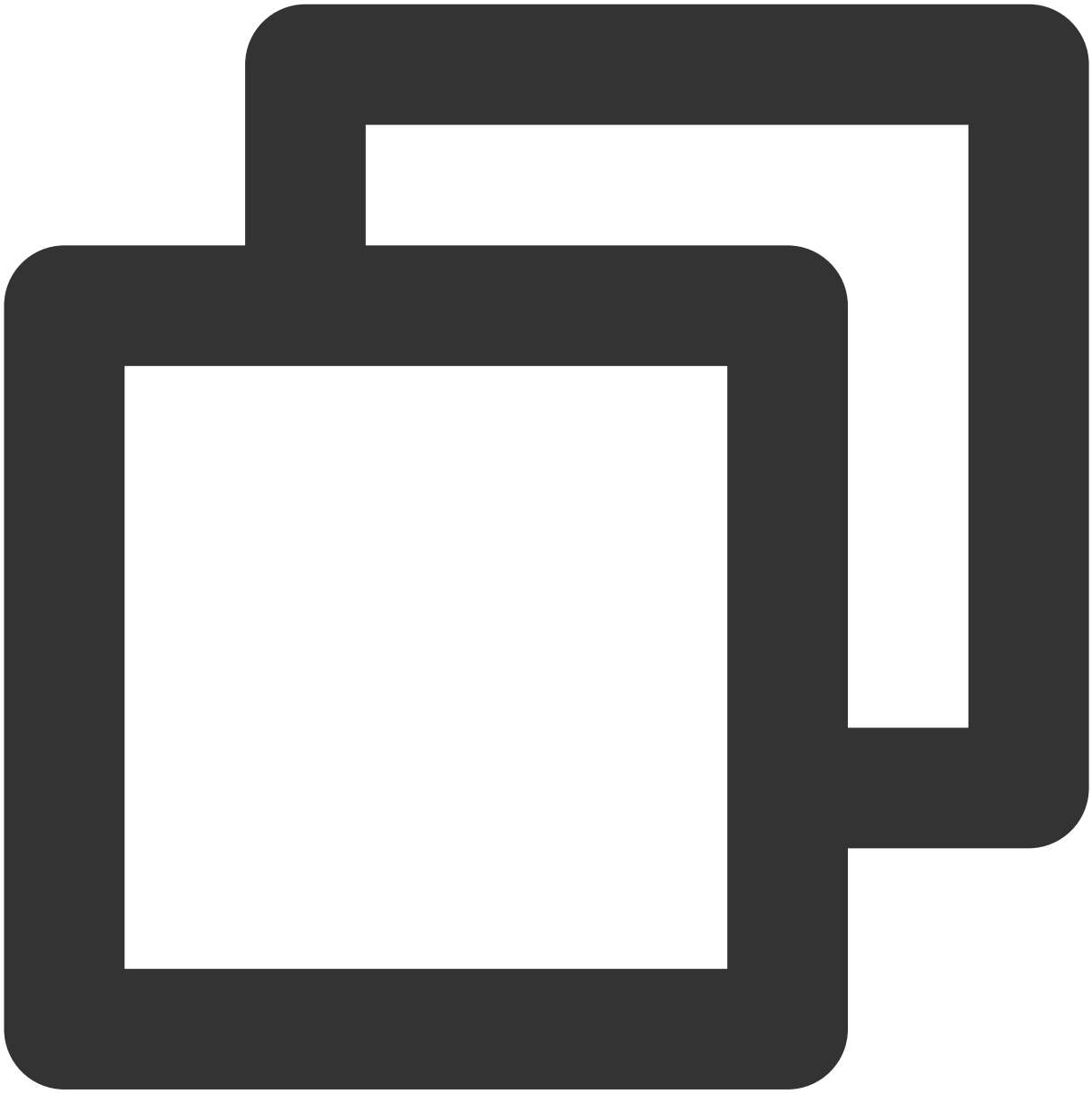
示例代码一：上传时添加盲水印



```
const filePath = "temp-file-to-upload" // 本地文件路径
cos.putObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    Body: fs.createReadStream(filePath), // 上传文件对象
    Headers: {
        // 万象持久化接口，上传时持久化
        'Pic-Operations': '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.j
  },
}, function(err, data) {
```

```
    console.log(err || data);
});
```

示例代码二：下载时添加盲水印



```
cos.getObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'COS_REGION',
    Key: 'exampleobject',
    QueryString: 'watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGlj
}, function (err, data) {
```

```
    console.log(err || data);
    fs.writeFileSync('filepath', data.Body);   // 将图片内容保存本地
});
```

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 10:15:16

简介

本文档提供关于媒体 bucket 接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

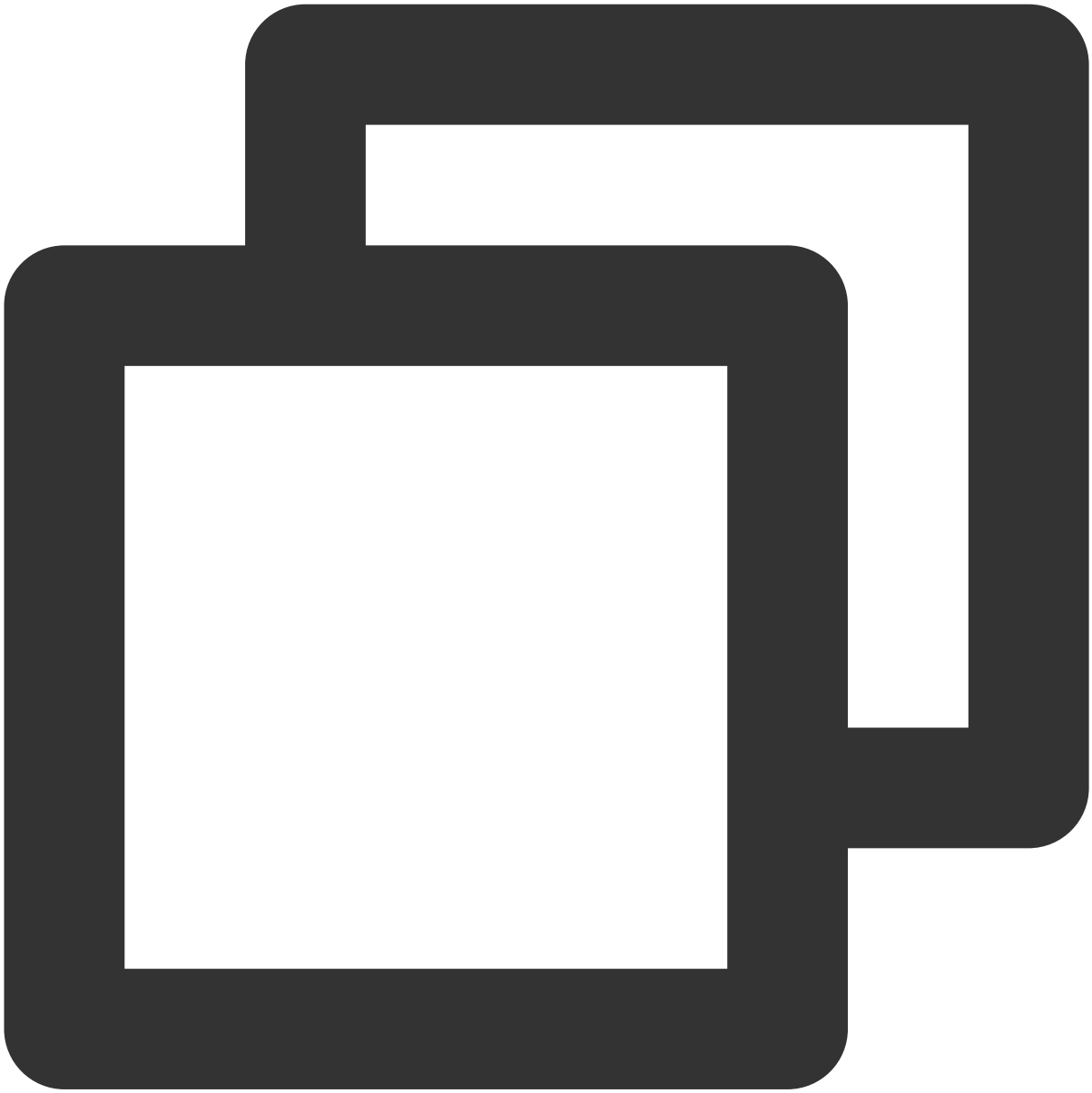
功能说明

用于查询已经开通媒体处理功能的存储桶。

注意：

COS Node.js SDK 版本需要大于等于 v2.11.2。

使用示例



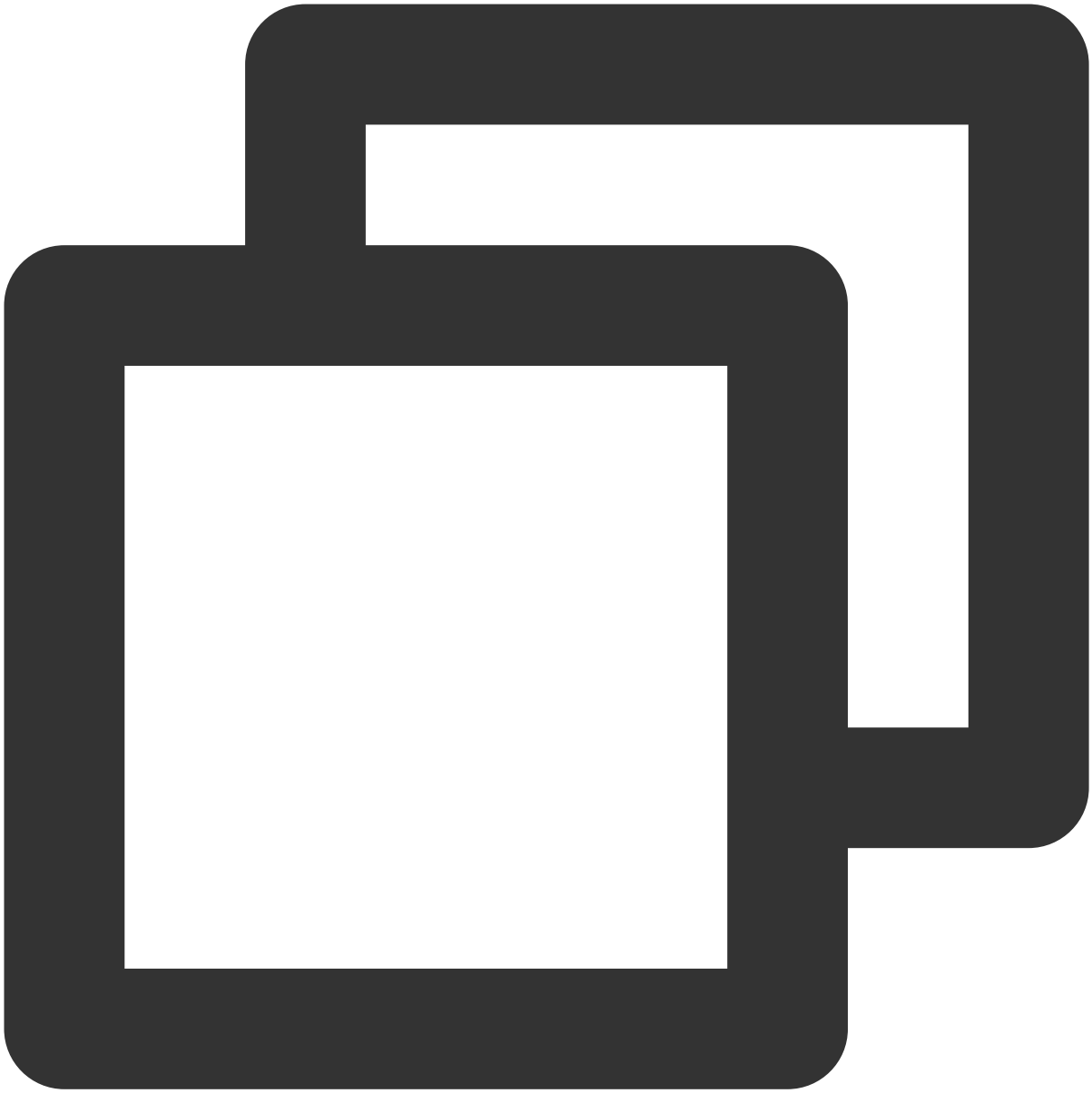
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
var host = 'ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/mediabucket';
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
```

```
    Key: 'mediabucket', /** 固定值，必须 */
    Url: url,
    Query: {
        pageNumber: '1', /** 第几页，非必须 */
        pageSize: '10', /** 每页个数，非必须 */
        // regions: 'ap-chengdu', /** 地域信息，例如'ap-beijing'，支持多个值用逗号分隔如'a
        // bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-125000000
        // bucketName: 'test', /** 存储桶名称前缀，前缀搜索，例如'test'，支持多个值用逗号分
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
PageNumber	第几页	String	否
PageSize	每页个数	String	否
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	String	否
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	String	否
BucketName	存储桶名称前缀，前缀搜索	String	否

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- Response	响应结果	Object
- - RequestId	请求的唯一 ID	String
- - TotalCount	媒体 Bucket 总数	String
- - PageNumber	当前页数，同请求中的 pageNumber	String
- - PageSize	每页个数，同请求中的 pageSize	String
- - MediaBucketList	媒体 Bucket 列表	Object
- - - BucketId	存储桶 ID	String
- - - Name	存储桶名称，同 BucketId	String
- - - Region	所在的地域	String
- - - CreateTime	创建时间	String

媒体信息接口

最近更新时间：2024-03-13 10:15:16

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

注意：

COS Node.js SDK 版本需要大于等于 v2.11.2。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

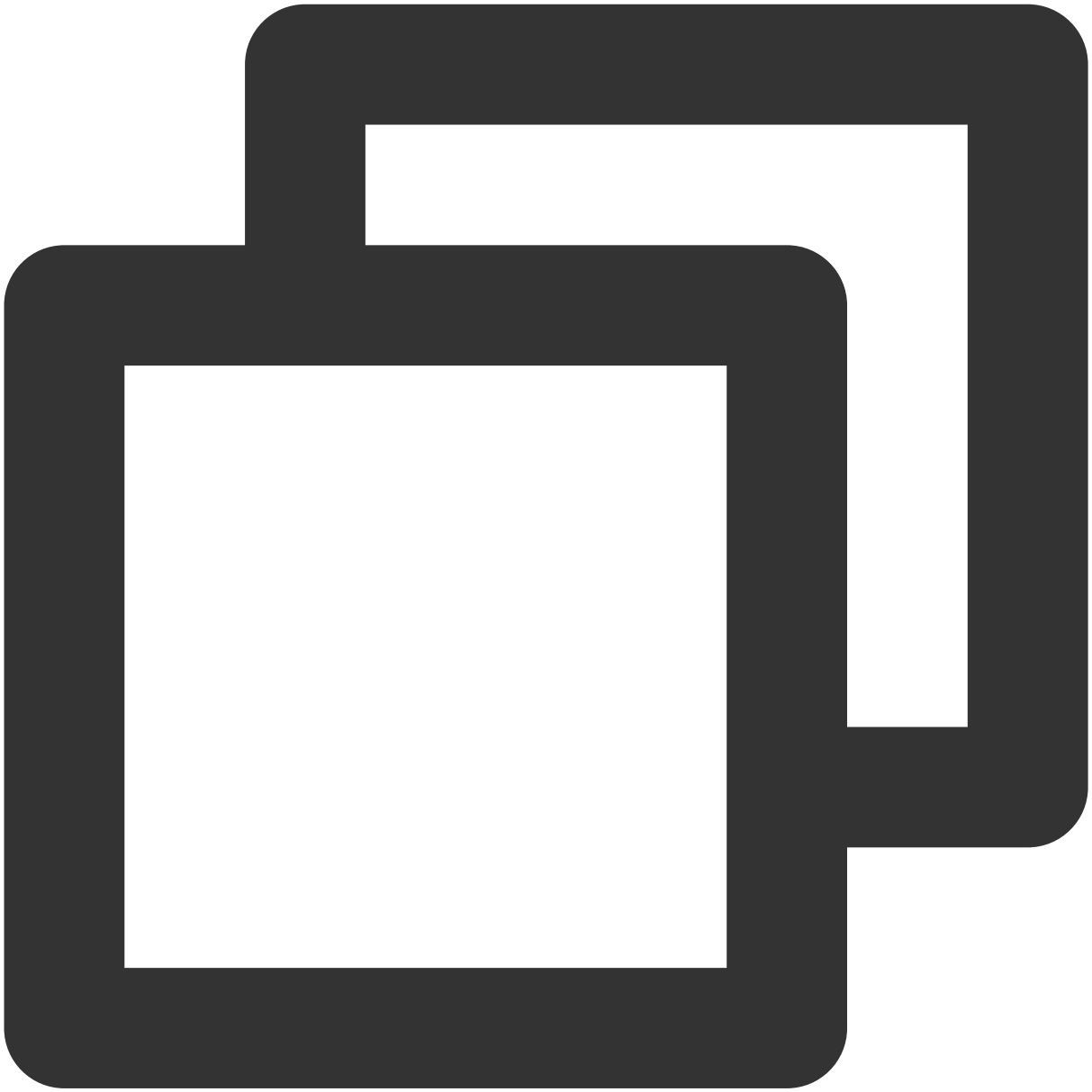
功能说明

用于查询媒体文件的信息。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



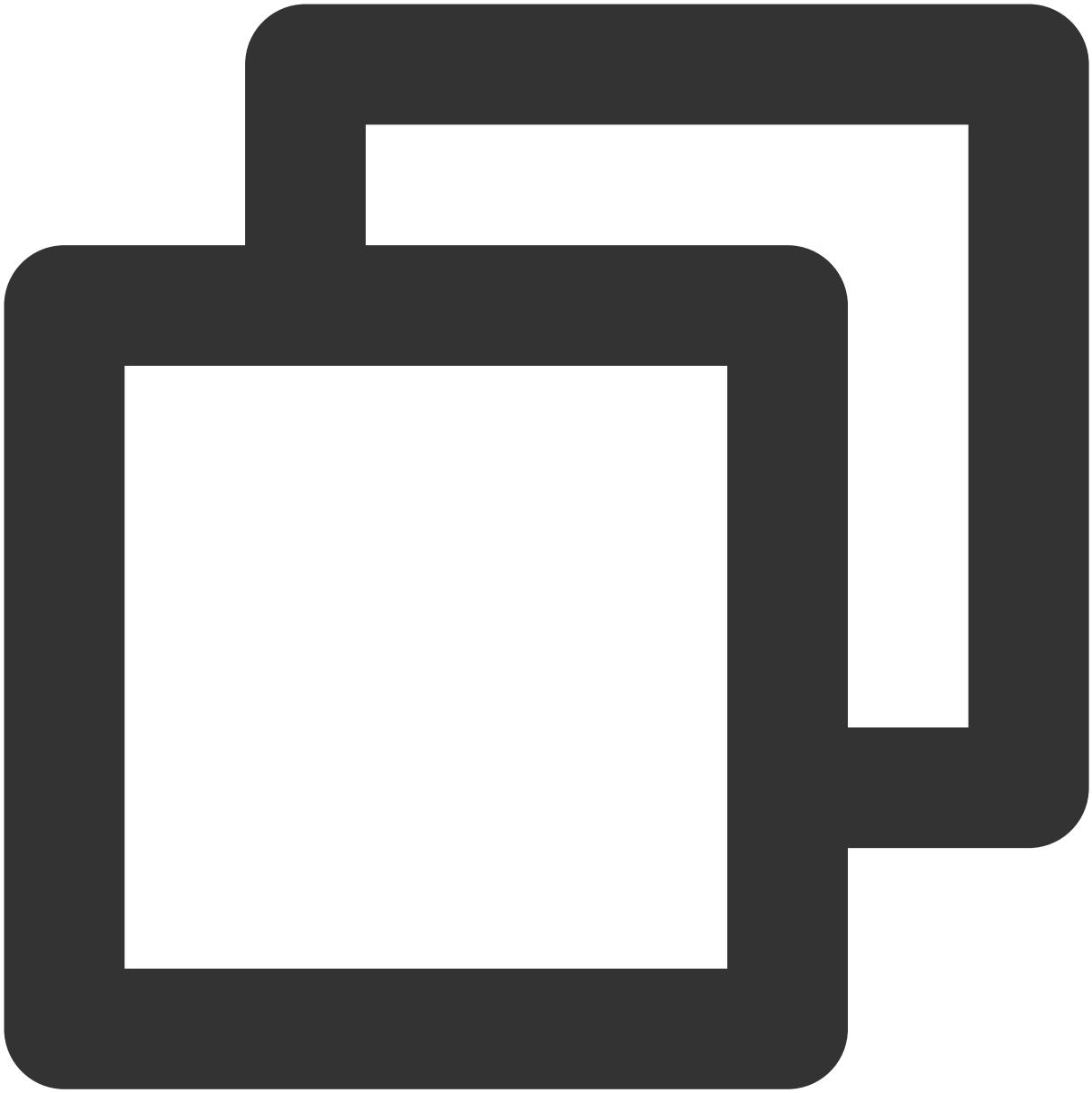
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

```
        'ci-process': 'videoinfo' /** 固定值，必须 */
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
ci-process	操作类型，固定使用 videoinfo	String	是

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Response	响应结果	Object
- - MediaInfo	媒体的详细信息	Object
- - - Stream	流信息	Object
- - - - Video	视频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - Profile	视频编码档位	String
- - - - - Height	视频高，单位 px	Number
- - - - - Width	视频宽，单位 px	Number
- - - - - HasBFrame	是否有B帧。1表示有，0表示无	Number
- - - - - RefFrames	视频编码的参考帧个数	Number
- - - - - Sar	采样宽高比	String
- - - - - Dar	显示宽高比	String
- - - - - PixFormat	像素格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - FieldOrder	场的顺序	String
- - - - - Level	视频编码等级	Number
- - - - - Fps	视频帧率	Number
- - - - - AvgFps	平均帧率	String
- - - - - Timebase	时基	String
- - - - - StartTime	视频开始时间，单位为秒	Number
- - - - - Duration	视频时长，单位为秒	Number
- - - - - Bitrate	比特率，单位为 kbps	Number
- - - - - NumFrames	总帧数	Number
- - - - - Language	语言	String
- - - - Audio	音频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - SampleFmt	采样格式	String
- - - - - SampleRate	采样率	Number
- - - - - Channel	通道数量	Number
- - - - - ChannelLayout	通道格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - Timebase	时基	String
- - - - - StartTime	音频开始时间，单位秒	Number
- - - - - Duration	音频时长，单位秒	Number
- - - - - Bitrate	比特率，单位 kbps	Number
- - - - - Language	语言	String
- - - - Subtitle	字幕信息	Container
- - - - - Index	该流的编号	Number
- - - - - Language	语言，und 表示无查询结果	String
- - - Format	格式信息	Object
- - - - NumStream	Stream（包含 Video、Audio、Subtitle）的数量	Number
- - - - NumProgram	节目的数量	Number
- - - - FormatName	容器格式名字	String
- - - - FormatLongName	容器格式的详细名字	String
- - - - StartTime	起始时间，单位为秒	Number
- - - - Duration	时长，单位为秒	Number
- - - - Bitrate	比特率，单位为 kbps	Number
- - - - Size	大小，单位为 Byte	Number

版权所有：腾讯云计算（北京）有限责任公司

媒体截图接口

最近更新时间：2024-03-13 10:15:16

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

注意：

COS Node.js SDK 版本需要大于等于 v2.11.2。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

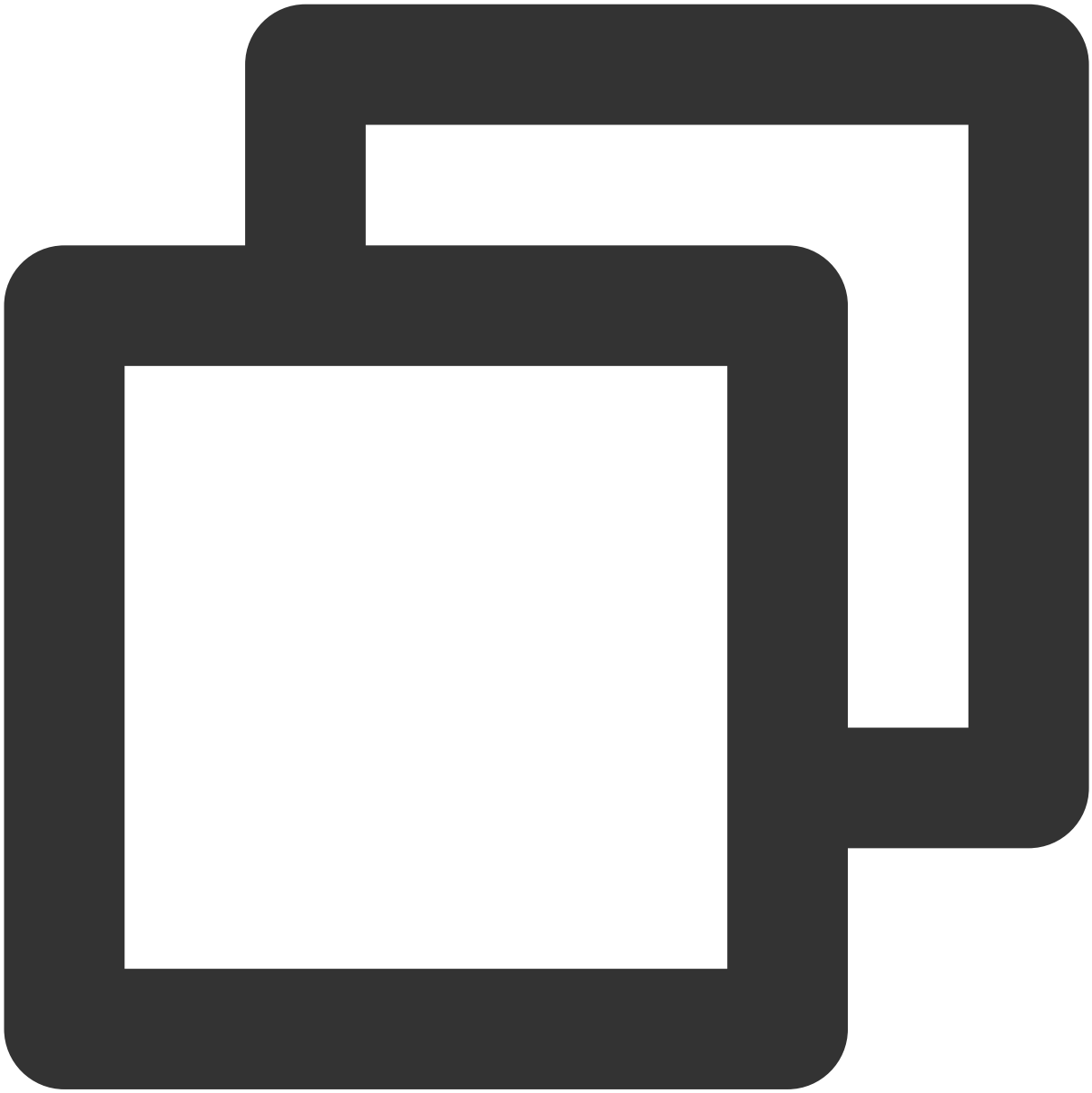
功能说明

用于查询媒体文件在某个时间的截图。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



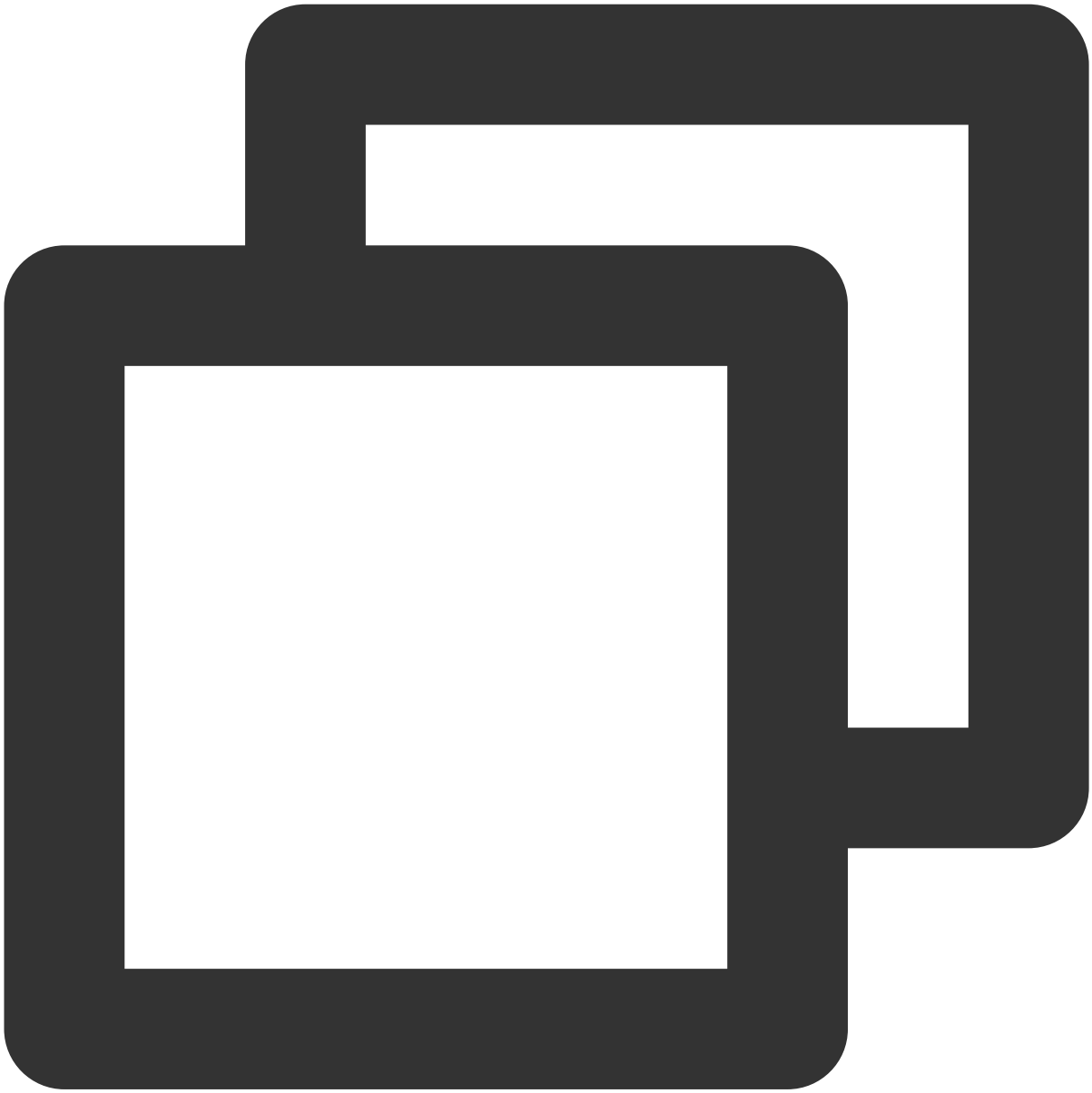
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

```
            '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，默认为'exactframe'，非必须 */
        },
        RawBody: true,
    },
    function(err, data){
        console.log(err || data);
    });
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	对象键（Object 的名称），对象在存储桶中的唯一标识，详情请参见 对象概述	String	是
ci-process	操作类型，固定使用 snapshot	String	是
time	截图的时间点，单位为秒	Number	是
width	截图的宽。默认为0	Number	否
height	截图的高。默认为0。当 width 和 height 都为0时，表示使用视频的宽高；如果单个为0，则以另外一个值按视频宽高比例自动适应	Number	否
format	截图的格式，支持 jpg 和 png，默认 jpg	String	否
rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转 off：不旋转，默认值为 auto	String	否
mode	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，默认值为 exactframe	String	否

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Body	返回的文件内容，默认为 String 形式	String

PHP SDK
快速入门

最近更新时间：2024-03-13 10:25:33

下载与安装

相关资源

对象存储 COS 的 XML PHP SDK 源码下载地址：XML PHP SDK。
SDK 快速下载地址：XML PHP SDK。
示例 Demo 程序地址：PHP sample。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：PHP SDK 常见问题。

说明：

如果您在使用 XML 版本 SDK 时遇到函数或方法不存在等错误，请先将 XML 版本 SDK 升级到最新版再重试。

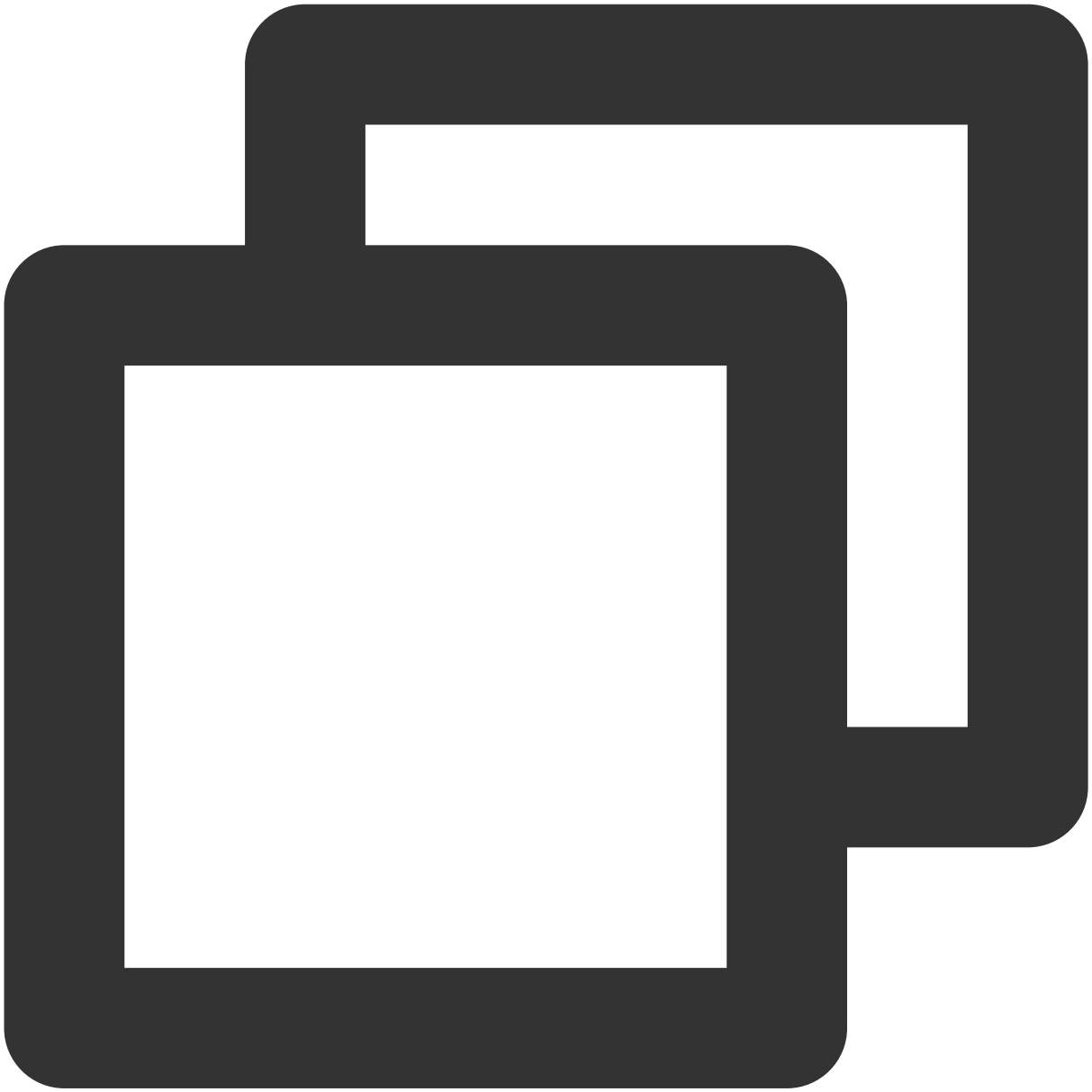
环境依赖

PHP 5.6+
您可以通过 `php -v` 命令查看当前的 PHP 版本。

注意：

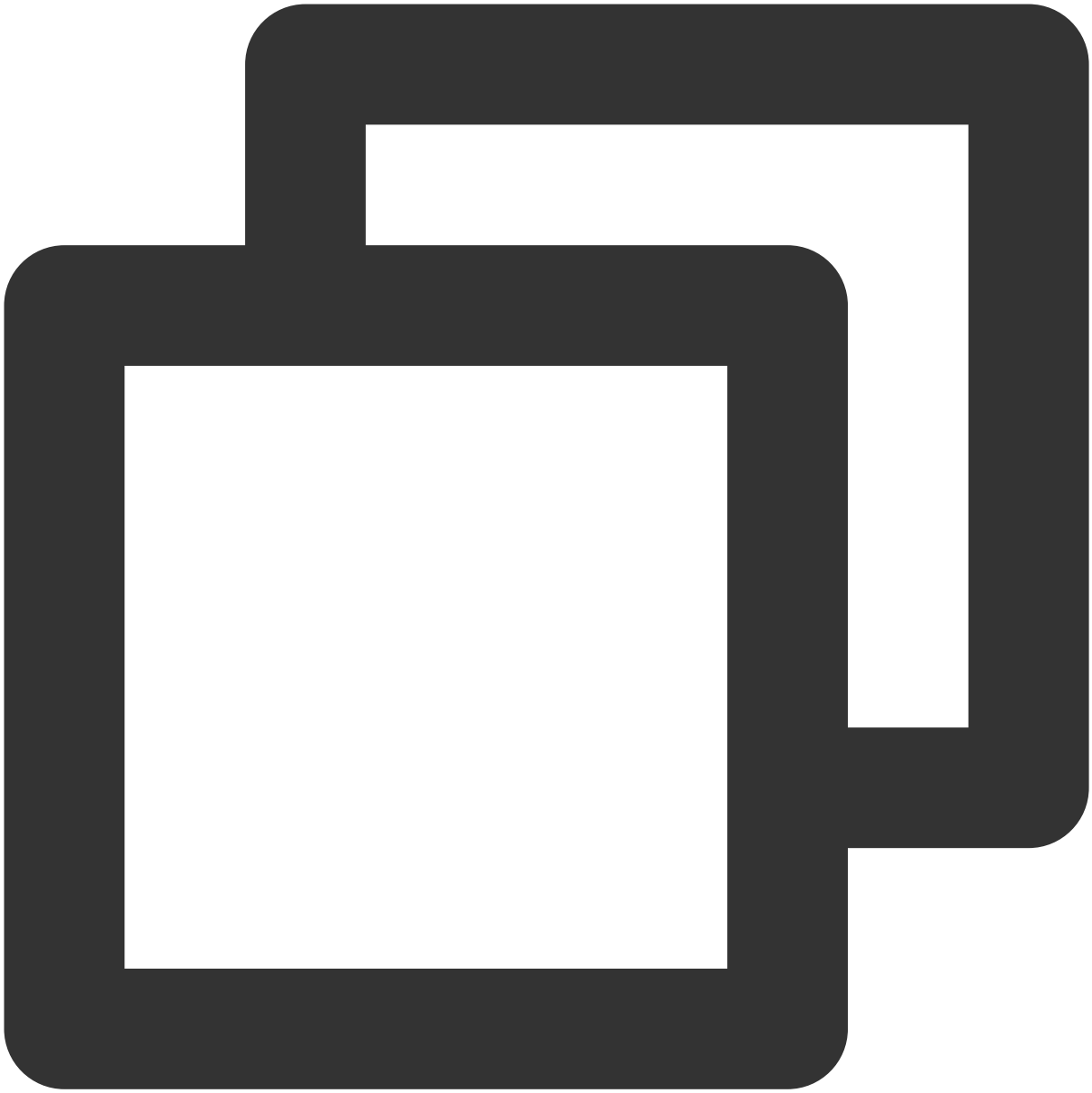
如果您的 PHP 版本 `>=5.3` 且 `<5.6`，请使用 v1.3 版本。

cURL 扩展
xml 扩展
dom 扩展
mbstring 扩展
json 扩展
您可以通过 `php -m` 命令查看以上扩展是否已经安装好。
Ubuntu 系统中，您可以使用 apt-get 包管理器安装 PHP 的 相关 扩展，安装命令如下：



```
sudo apt-get install php-curl php-xml php-dom php-mbstring php-json
```

CentOS 系统中，您可以使用 yum 包管理器安装 PHP 的 cURL 扩展。



```
sudo yum install php-curl php-xml php-dom php-mbstring php-json
```

安装 SDK

SDK 安装有三种方式：Composer 方式、Phar 方式 和 源码方式。

Composer 方式

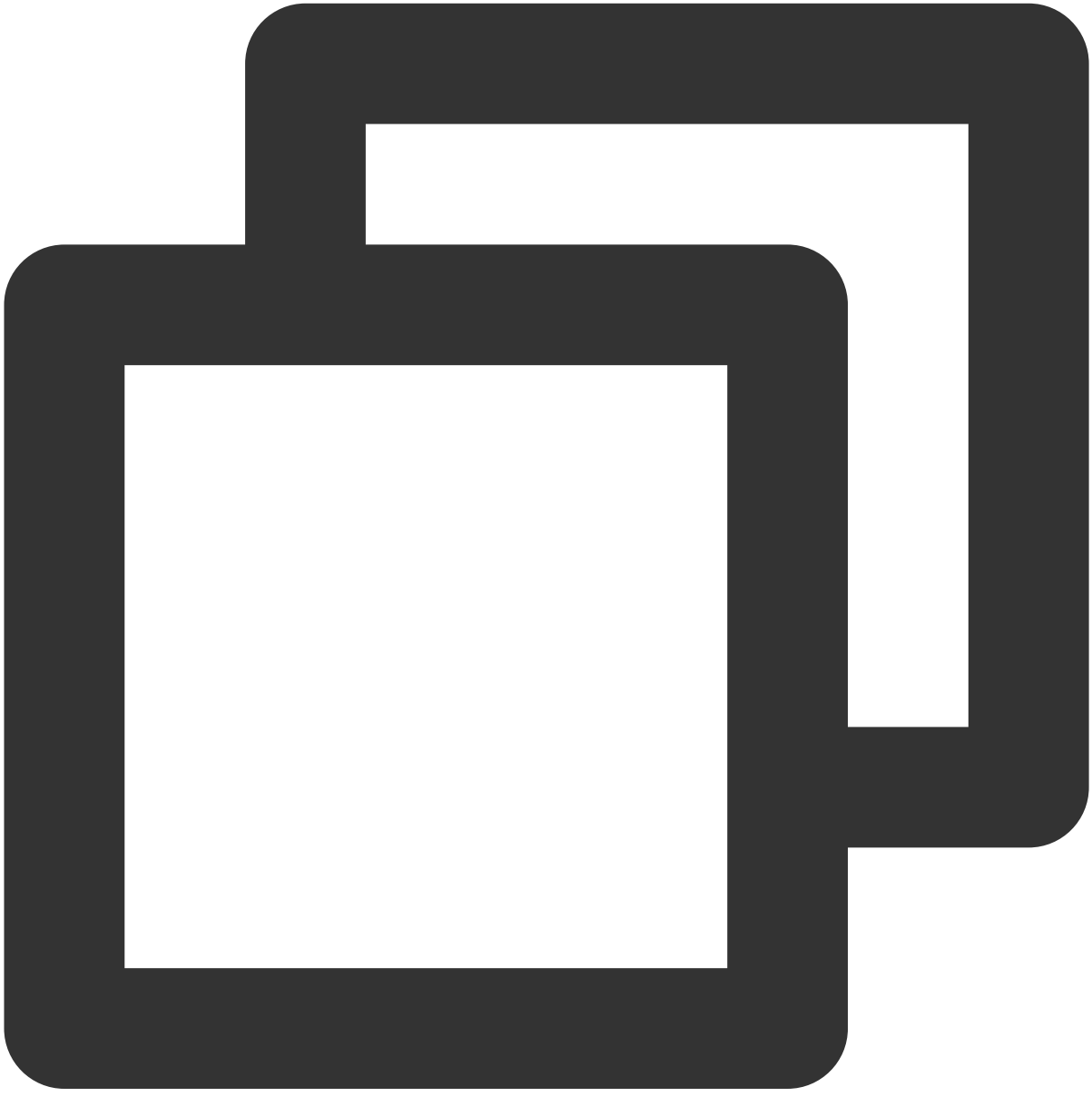
推荐使用 Composer 安装 cos-php-sdk-v5，Composer 是 PHP 的依赖管理工具，允许您声明项目所需的依赖，然后自动将它们安装到您的项目中。

说明：

您可以在 Composer 官网 上找到更多关于如何安装 Composer，配置自动加载以及用于定义依赖项的其他最佳实践等相关信息。

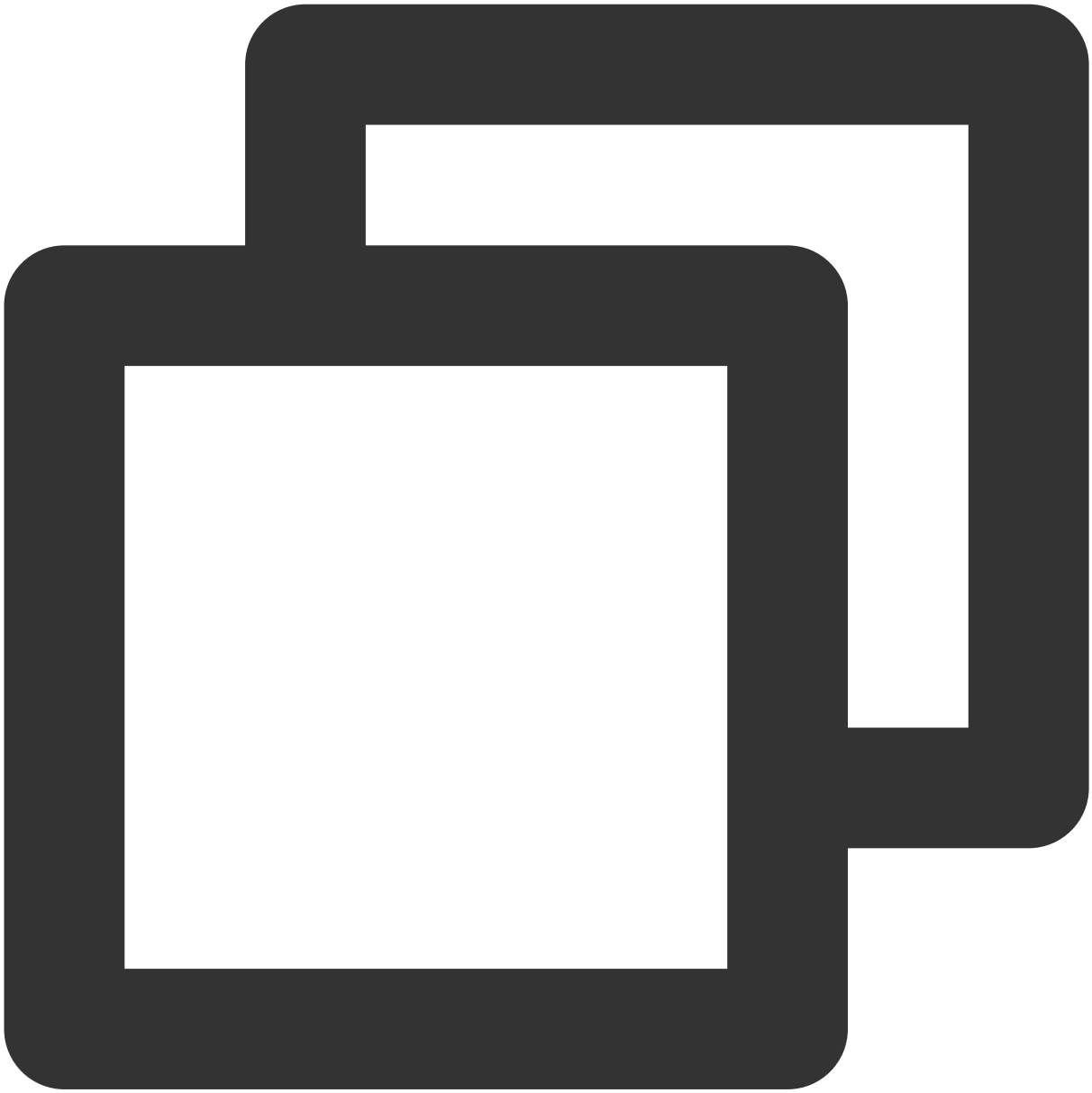
安装步骤

1. 打开终端。
2. 执行以下命令，下载 Composer。



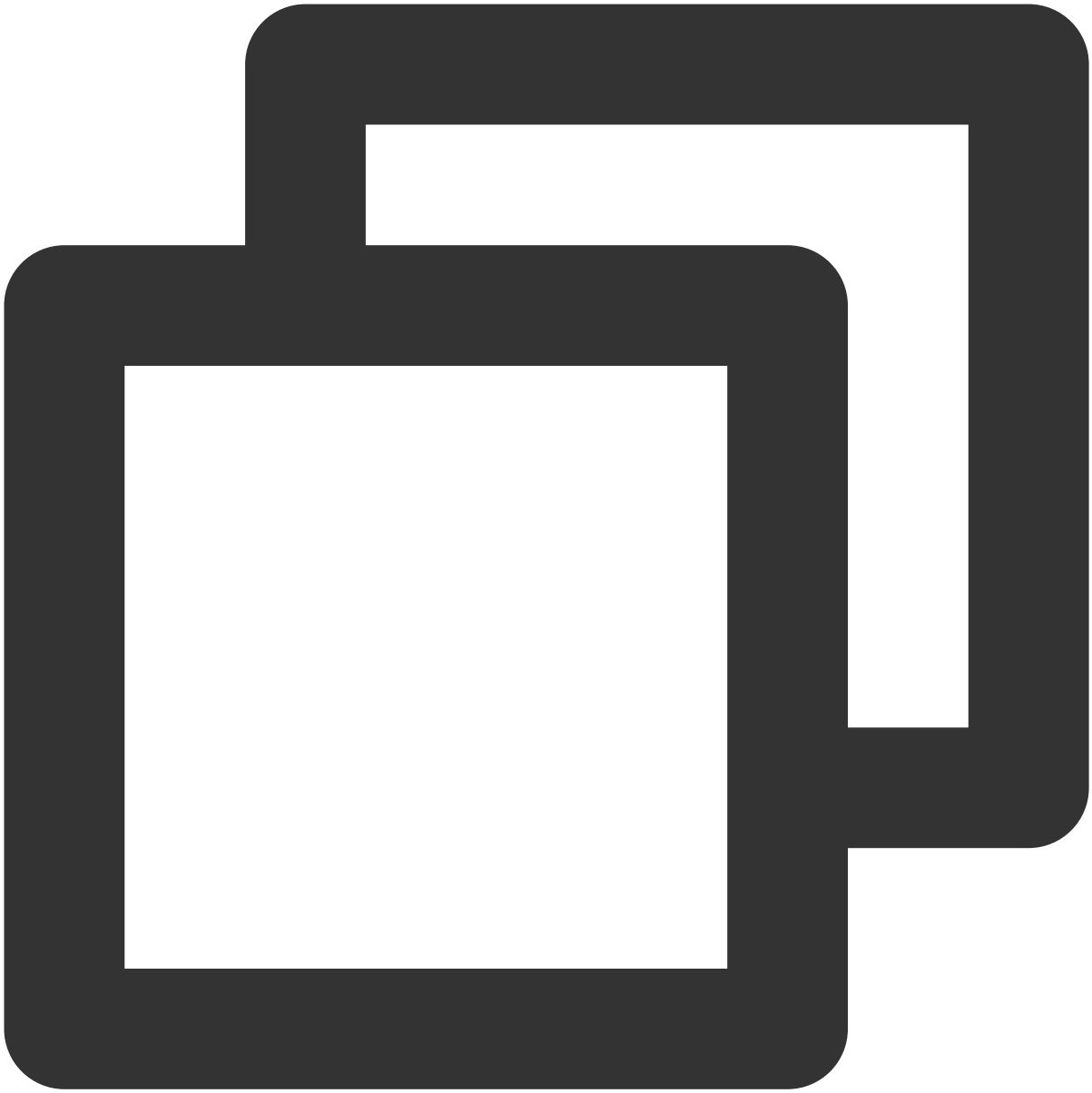
```
curl -sS https://getcomposer.org/installer | php
```

3. 创建一个名为 `composer.json` 的文件，内容如下：



```
{
 "require": {
     "qcloud/cos-sdk-v5": ">=2.0"
 }
}
```

4. 执行以下命令，使用 Composer 安装。



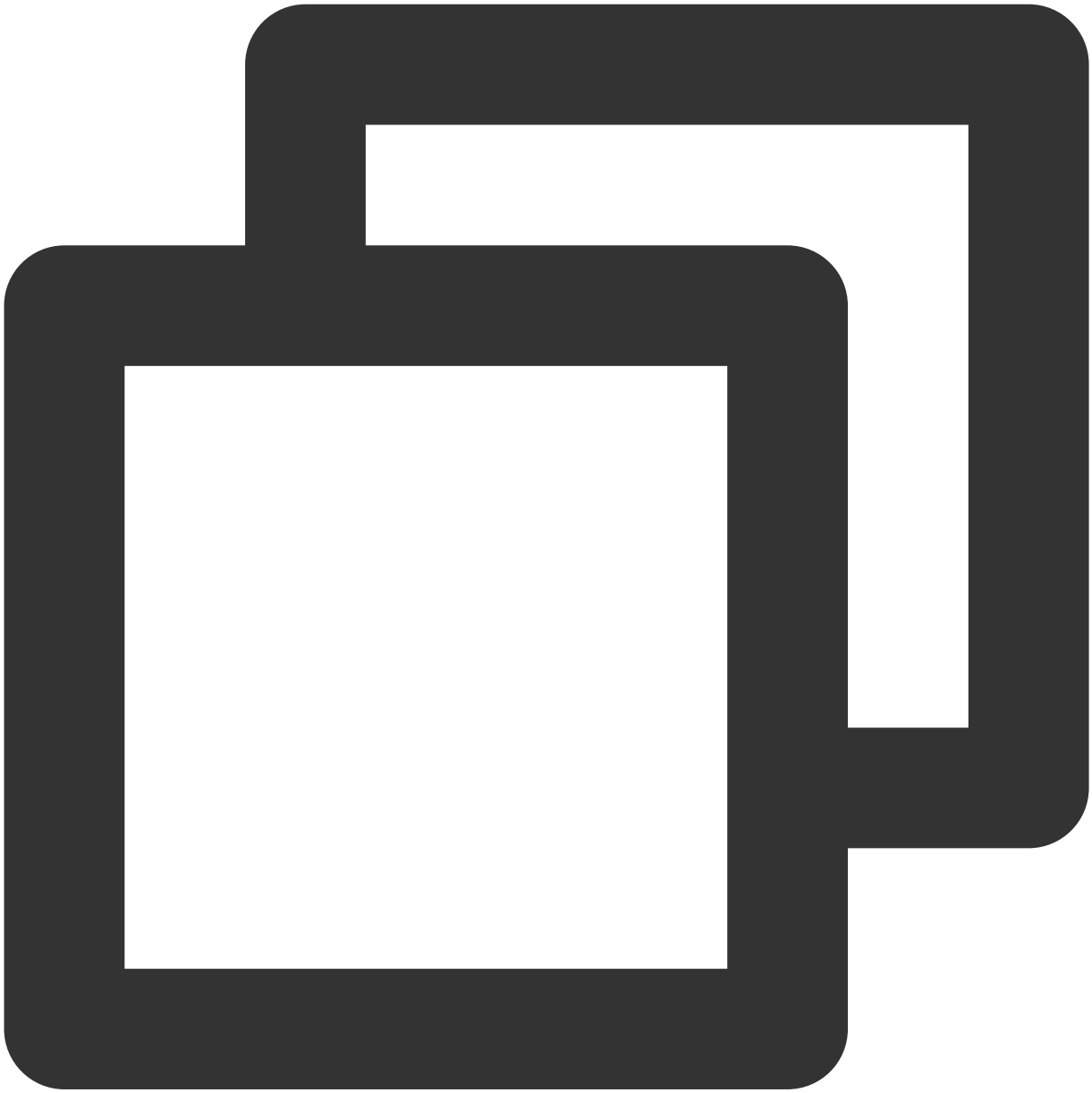
```
php composer.phar install
```

使用该命令后会在当前目录中创建一个 vendor 文件夹，里面包含 SDK 的依赖库和一个 autoload.php 脚本，方便在项目中调用。

注意：

目前已支持 Composer 根据当前 PHP 版本下载 guzzle6 或 guzzle7。guzzle7 版本支持 laravel8 框架。当 PHP 版本 `>= 7.2.5` 时自动下载 guzzle7 版本，反之下载 guzzle6 版本。

5. 通过 autoloader 脚本调用 cos-php-sdk-v5，在代码中引入 autoload.php 文件:



```
require '/path/to/sdk/vendor/autoload.php';
```

注意：

执行 composer 安装后，这里的路径需更改为 autoload.php 文件所对应的路径，否则会调用不到相关方法。如您安装的路径为/Users/username/project，则项目中引用的路径应填写为/Users/username/project/vendor/autoload.php。至此，您的项目已经可以使用 COS XML PHP SDK 了。

Phar 方式

Phar 方式安装 SDK 的步骤如下：

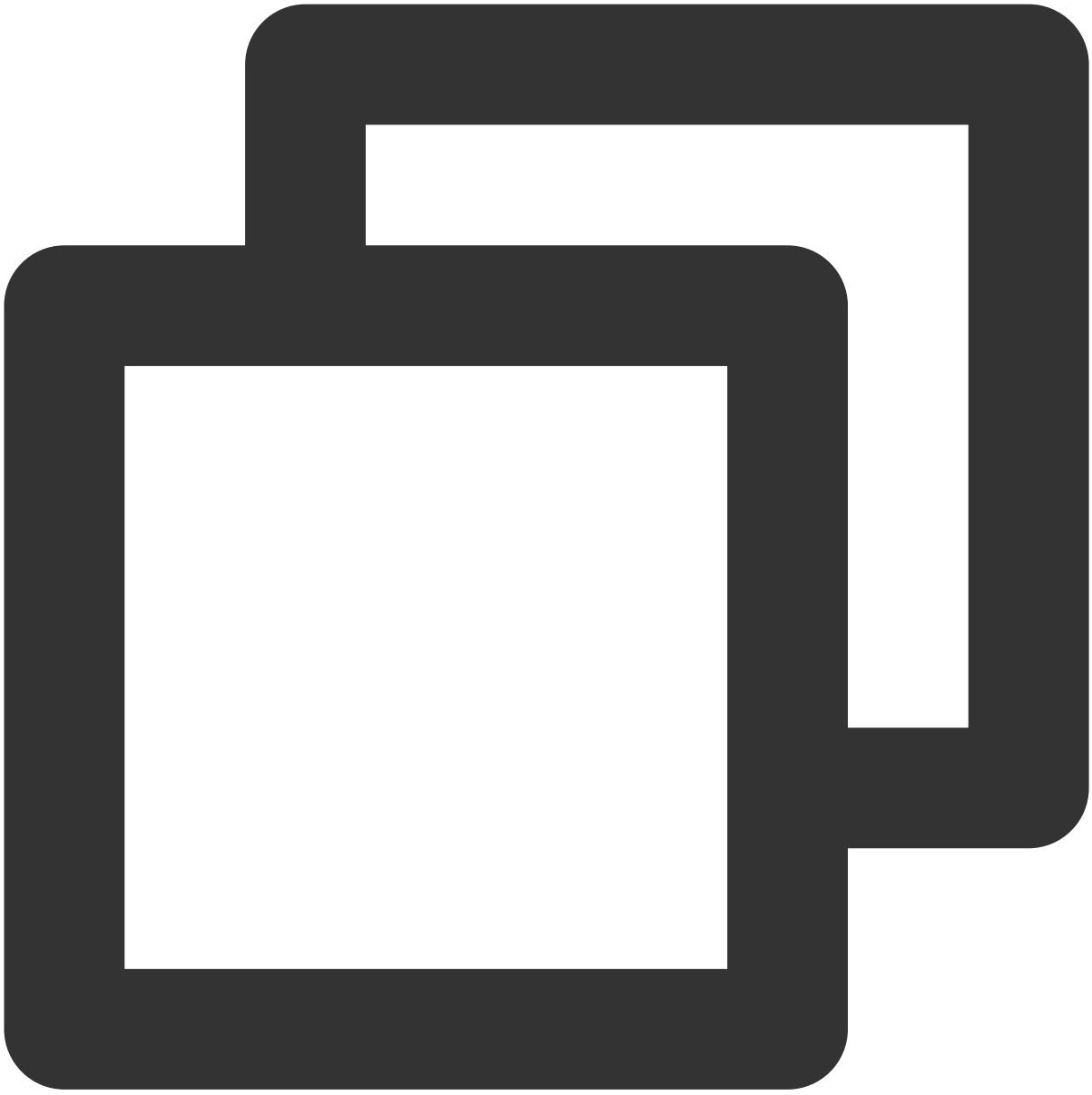
1. 在 GitHub 发布页面 下载相应的 phar 文件。

说明：

对于 PHP 版本 `>= 5.6` 且 `<7.2.5` ，请下载 `cos-sdk-v5-6.phar` ，以使用 Guzzle6 版本。

对于 PHP 版本 `>=7.2.5` 的请下载 `cos-sdk-v5-7.phar` ，以使用 Guzzle7 版本。

2. 在代码中引入 phar 文件：



```
require  '/path/to/cos-sdk-v5-x.phar';
```

源码方式

源码方式安装 SDK 的步骤如下：

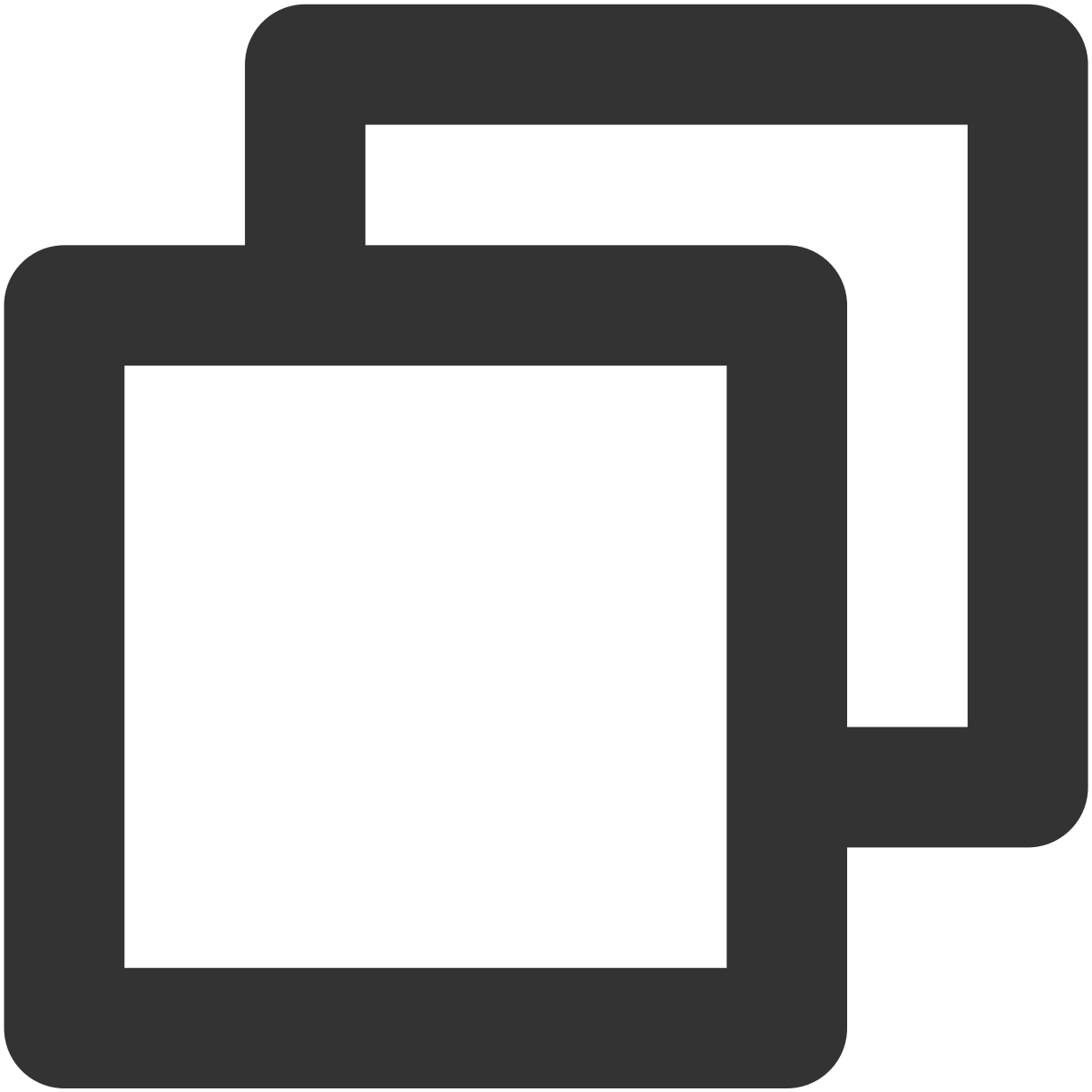
1. 在 SDK 下载页面 下载 `cos-sdk-v5.tar.gz` 压缩文件。

说明：

对于 PHP 版本 `>= 5.6` 且 `<7.2.5` ，请下载 `cos-sdk-v5-6.tar.gz` ，以使用 Guzzle6 版本。

对于 PHP 版本 `>=7.2.5` 的请下载 `cos-sdk-v5-7.tar.gz` ,以使用 Guzzle7 版本。

2. 解压后通过 `autoload.php` 脚本加载 SDK，在代码中引入 autoload.php 文件:



```
require '/path/to/sdk/vendor/autoload.php';
```

注意：

`Source code` 压缩包为 Github 默认打包的代码包，里面不包含 `vendor` 目录。请注意下载release包（cos-sdk-v5-x.tar.gz包）而不是Source包，也不要直接 clone 整个仓库。否则会缺失index.php以及vendor包。

开始使用

下面为您介绍如何使用 COS PHP SDK 完成一个基础操作，如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。关于示例中的参数说明，请参见 存储桶操作 和 对象操作 文档。

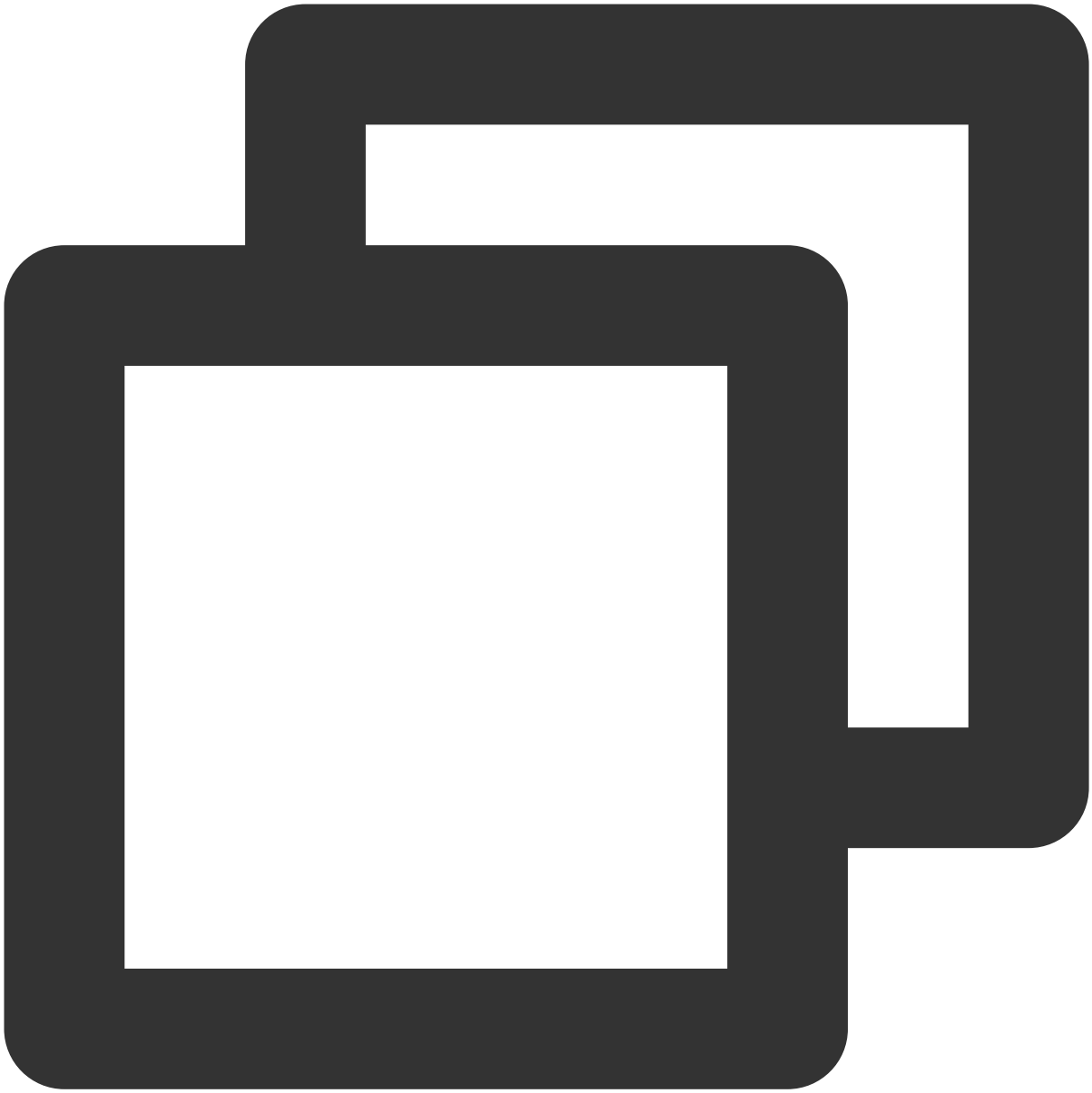
初始化

注意：

建议用户使用子账号密钥 + 环境变量的方式调用 SDK，提高 SDK 使用的安全性。为子账号授权时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。

如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

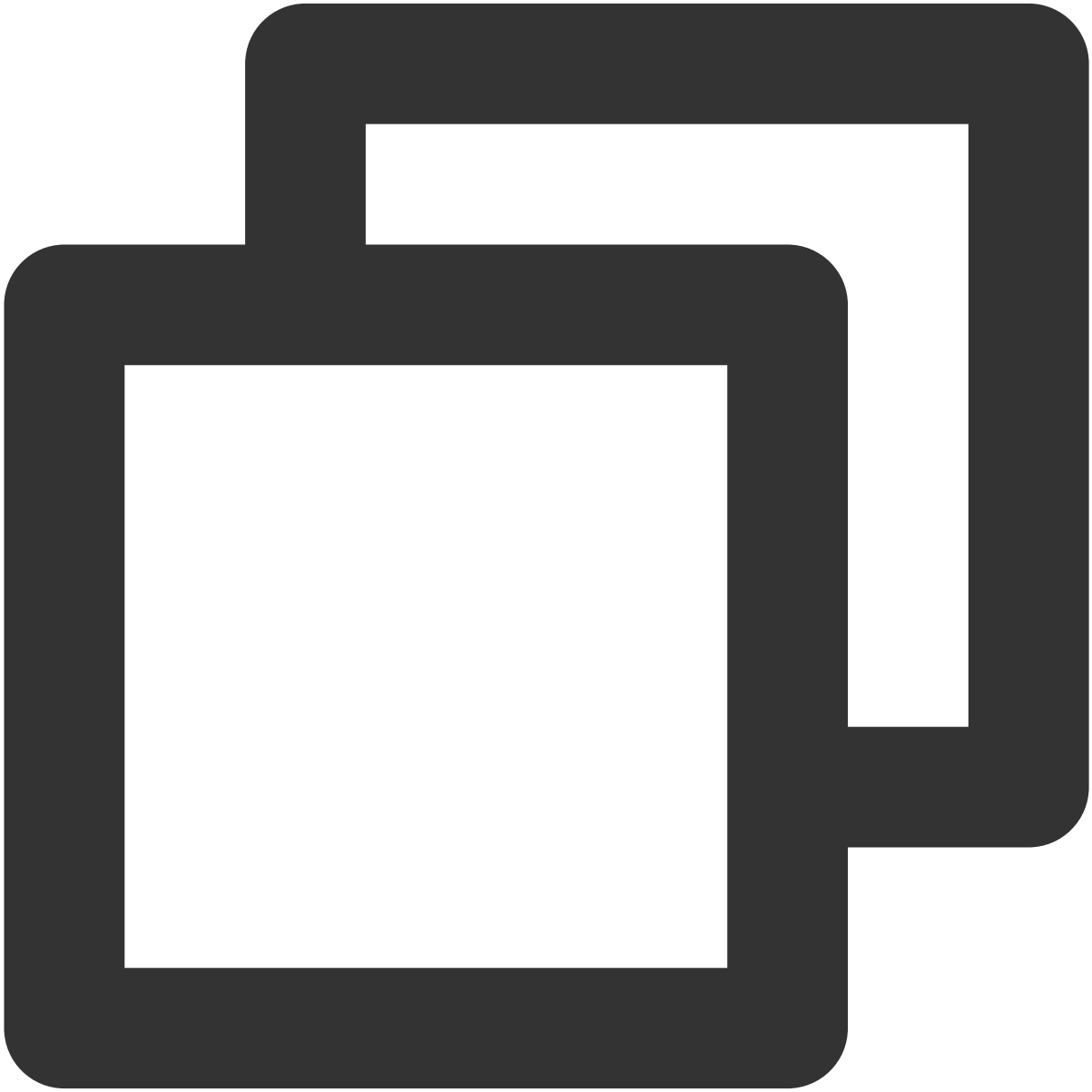
若您使用 临时密钥 初始化，请用下面方式创建实例：



```
$tmpSecretId = "TmpSecretId"; //临时密钥的 SecretId，临时密钥生成和使用指引参见 https://w
$tmpSecretKey = "TmpSecretKey"; //临时密钥的 SecretKey，临时密钥生成和使用指引参见 https:/
$tmpToken = "TmpToken"; //临时密钥的 Token，临时密钥生成和使用指引参见 https://www.tencent
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
        'credentials'=> array(
            'secretId'  => $tmpSecretId,
            'secretKey' => $tmpSecretKey,
```

```
            'token' => $tmpToken)));
```

若您使用永久密钥初始化 COSClient，可以先在访问管理控制台中的 API 密钥管理页面 获取 SecretId、SecretKey，使用永久密钥适用于大部分的应用场景。



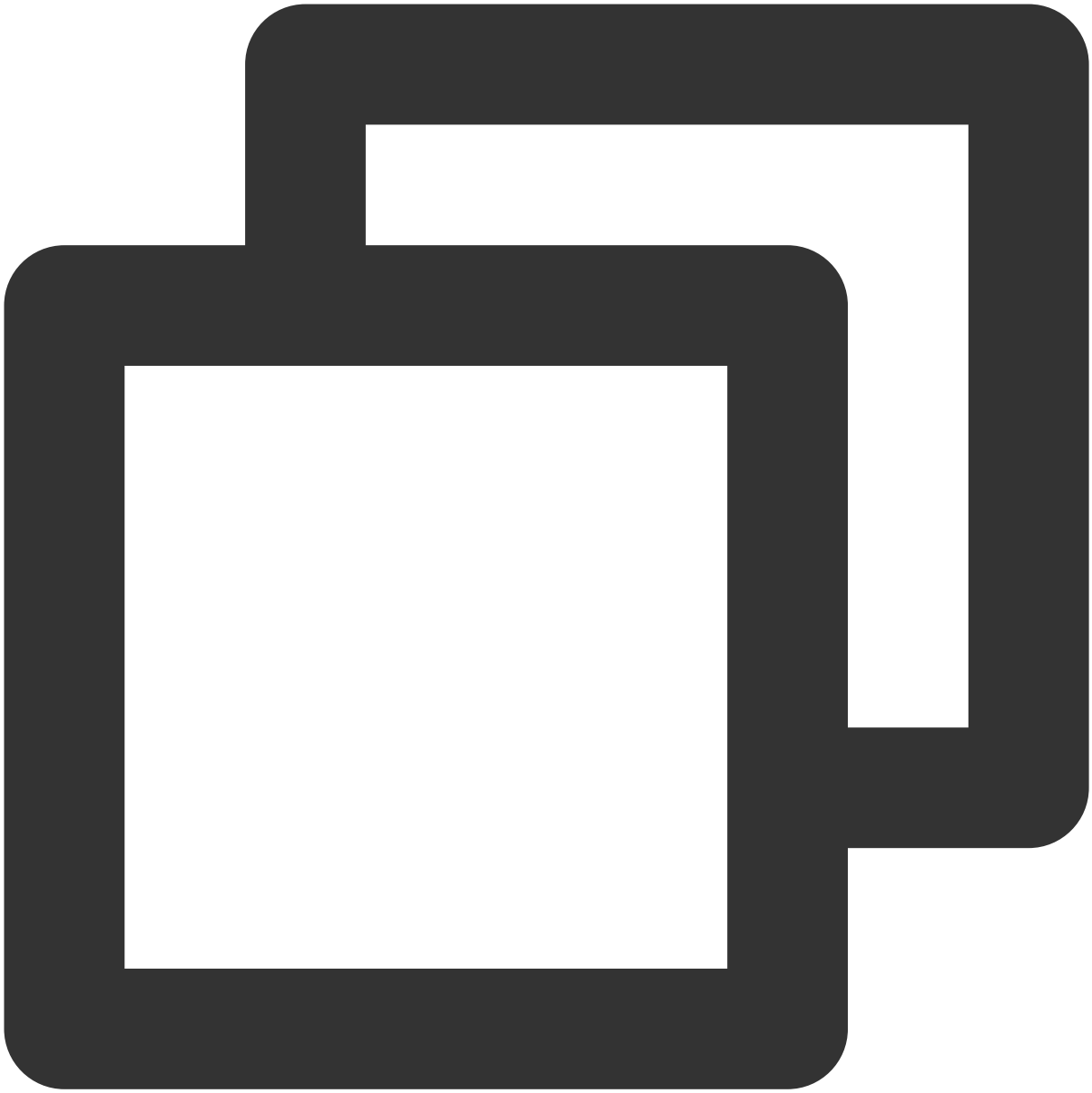
```
// SECRETID 和 SECRETKEY 请登录访问管理控制台进行查看和管理
$secretId = getenv('COS_SECRET_ID'); //用户的 SecretId，建议使用子账号密钥，授权遵循最小权
$secretKey = getenv('COS_SECRET_KEY'); //用户的 SecretKey，建议使用子账号密钥，授权遵循最
$region = "ap-beijing"; //用户的 region，已创建桶归属的 region 可以在控制台查看，https://c
$cosClient = new Qcloud\\Cos\\Client(
    array(
```

```
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
```

注意：

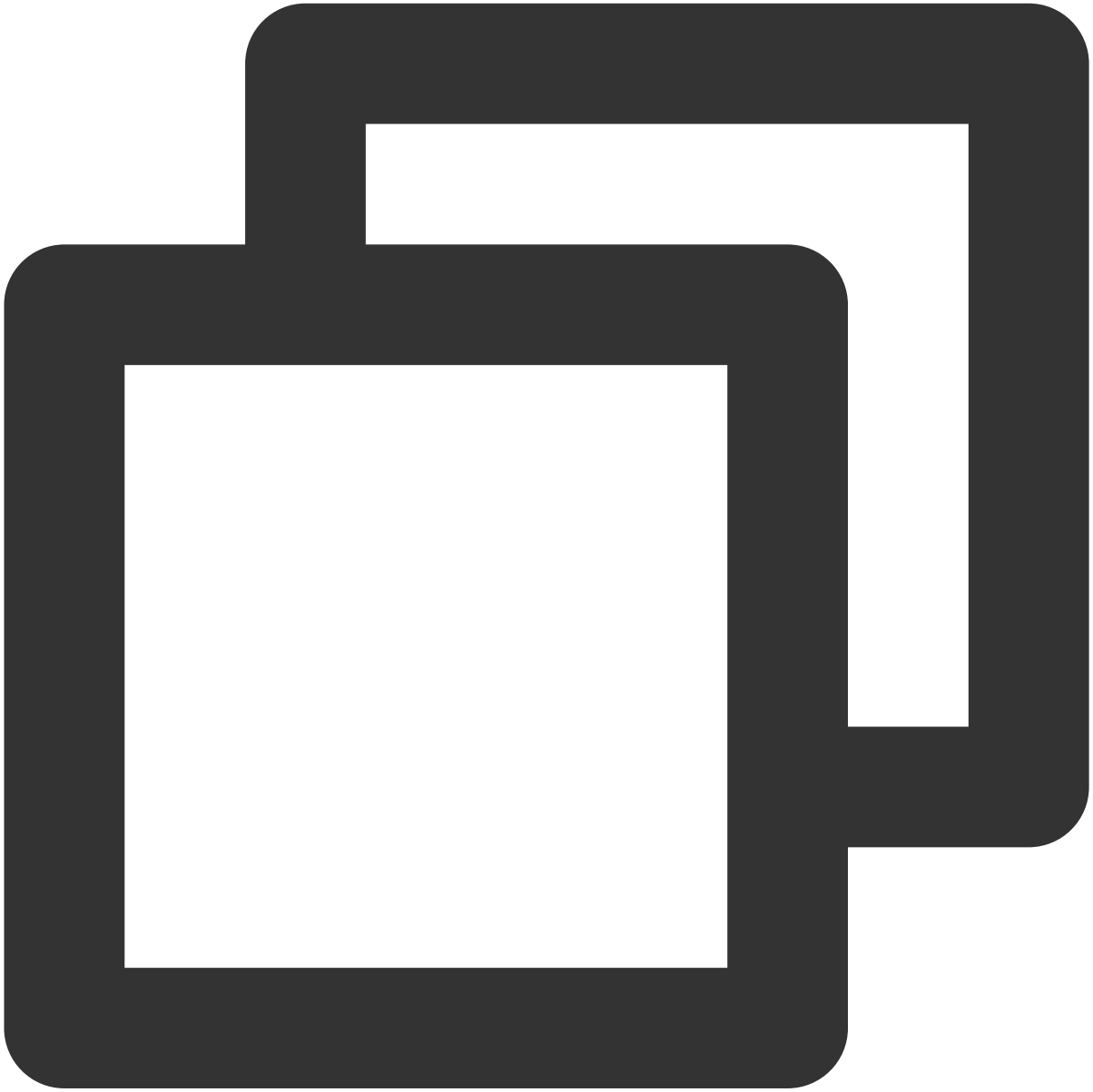
若未配置 HTTPS 证书，则需要删除 schema 参数或填入 `'schema' => 'http'` ；若填入 https 会出现 certificate problem。若您需要配置证书，可参考 PHP SDK 常见问题。

创建存储桶



```
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $result = $cosClient->createBucket(array('Bucket' => $bucket));
    //请求成功
    print_r($result);
} catch (\\Exception $e) {
    //请求失败
    echo($e);
}
```

查询存储桶列表



```
try {
    //请求成功
    $result = $cosClient->listBuckets();
    print_r($result);
} catch (\\Exception $e) {
    //请求失败
    echo($e);
}
```

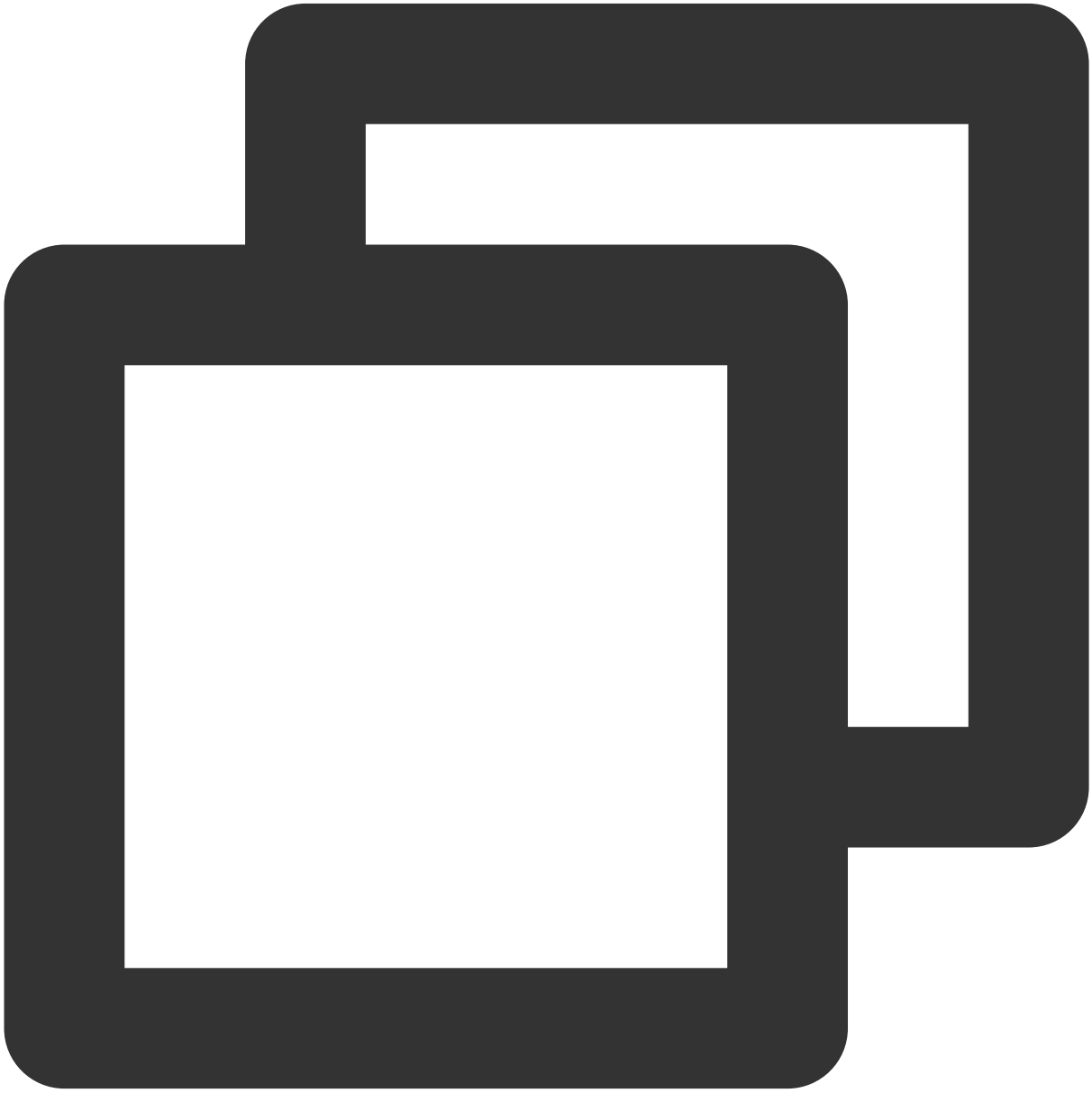
上传对象

注意：

使用 putObject 接口上传文件（最大5G）。

使用 Upload 接口分块上传文件，Upload 接口为复合上传接口，对小文件进行简单上传，对大文件进行分块上传。

参数说明可参见 对象操作 文档。

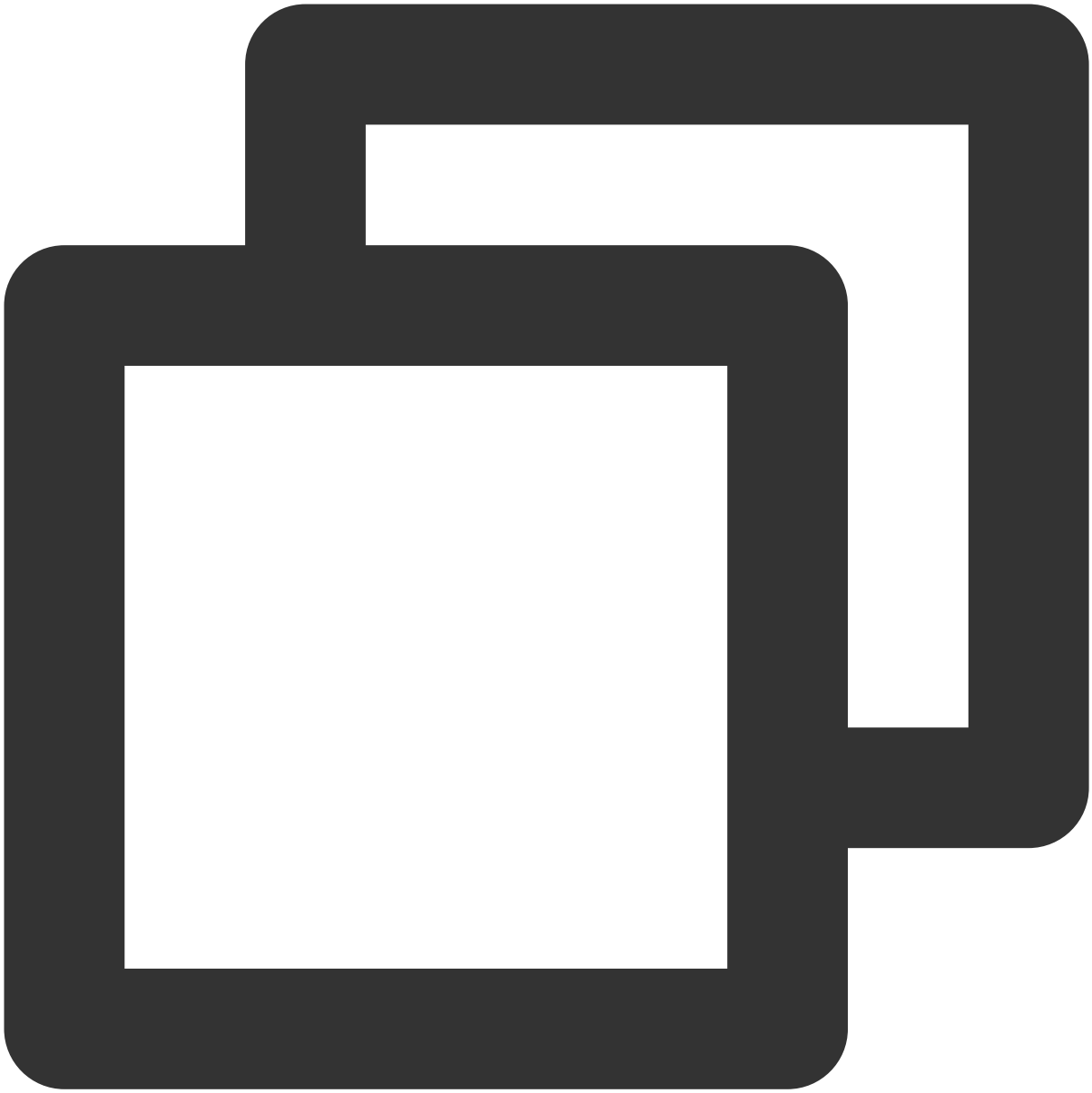


```
# 上传文件
## putObject(上传接口，最大支持上传5G文件)
### 上传内存中的字符串
try {
```

```
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $key = "exampleobject"; //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $result = $cosClient->putObject(array(
        'Bucket' => $bucket,
        'Key' => $key,
        'Body' => 'Hello World!'));
    print_r($result);
} catch (\\Exception $e) {
    echo "$e\\n";
}

### 上传文件流
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $key = "exampleobject"; //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $srcPath = "path/to/localFile";//本地文件绝对路径
    $file = fopen($srcPath, "rb");
    if ($file) {
        $result = $cosClient->putObject(array(
            'Bucket' => $bucket,
            'Key' => $key,
            'Body' => $file));
        print_r($result);
    }
} catch (\\Exception $e) {
    echo "$e\\n";
}

## Upload(高级上传接口，默认使用分块上传最大支持50T)
### 上传内存中的字符串
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $key = "exampleobject"; //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $result = $cosClient->Upload(
        $bucket = $bucket,
        $key = $key,
        $body = 'Hello World!');
    print_r($result);
} catch (\\Exception $e) {
    echo "$e\\n";
}

### 上传文件流
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $key = "exampleobject"; //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $srcPath = "path/to/localFile";//本地文件绝对路径
```

```
    $file = fopen($srcPath, 'rb');
    if ($file) {
        $result = $cosClient->Upload(
            $bucket = $bucket,
            $key = $key,
            $body = $file);
    }
    print_r($result);
} catch (\\Exception $e) {
    echo "$e\\n";
}
```

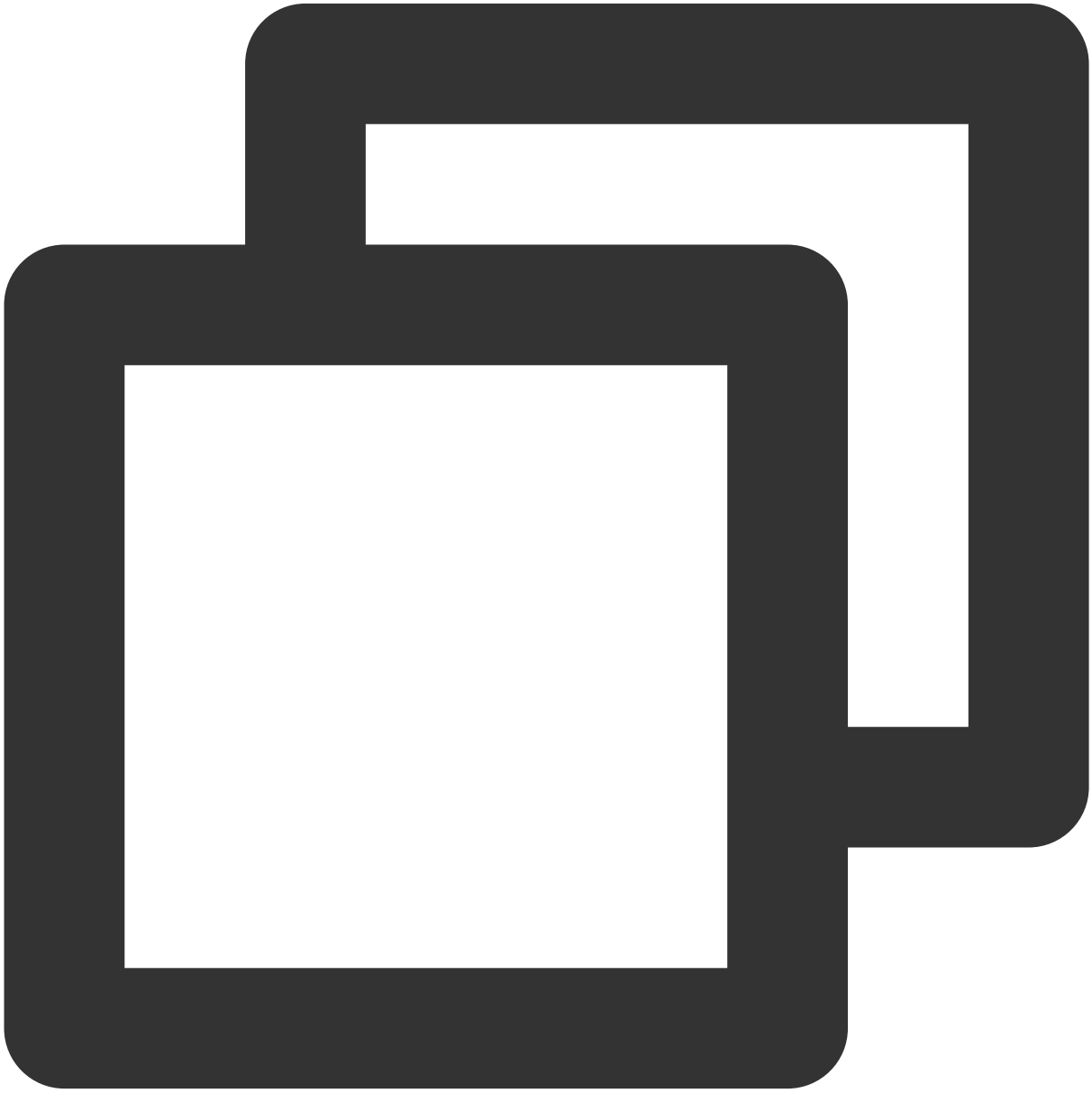
查询对象列表



```
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $result = $cosClient->listObjects(array(
        'Bucket' => $bucket
    ));
    // 请求成功
    if (isset($result['Contents'])) {
        foreach ($result['Contents'] as $rt) {
            print_r($rt);
        }
    }
```

```
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

单次调用 `listObjects` 接口一次只能查询1000个对象，如需要查询所有的对象，则需要循环调用。



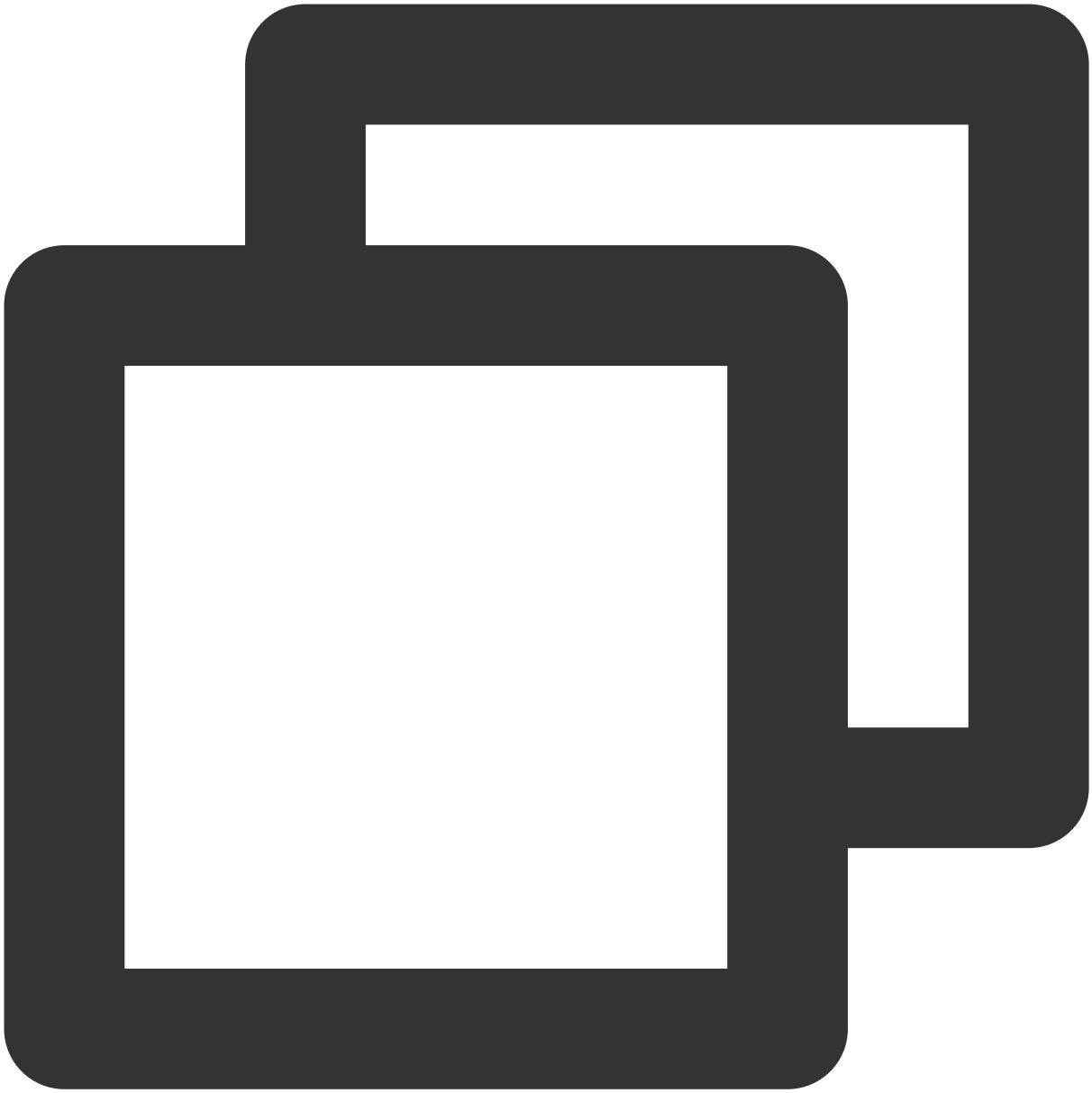
```
try {
    $bucket = "examplebucket-1250000000"; //存储桶名称 格式：BucketName-APPID
    $prefix = ''; //列出对象的前缀
    $marker = ''; //上次列出对象的断点
```

```
    while (true) {
        $result = $cosClient->listObjects(array(
            'Bucket' => $bucket,
            'Marker' => $marker,
            'MaxKeys' => 1000 //设置单次查询打印的最大数量，最大为1000
        ));
        if (isset($result['Contents'])) {
            foreach ($result['Contents'] as $rt) {
                // 打印key
                echo($rt['Key'] . "\\n");
            }
        }
        $marker = $result['NextMarker']; //设置新的断点
        if (!$result['IsTruncated']) {
            break; //判断是否已经查询完
        }
    }
} catch (\\Exception $e) {
    echo($e);
}
```

下载对象

使用 getObject 接口下载文件。

使用 getObjectUrl 接口获取文件下载 URL。

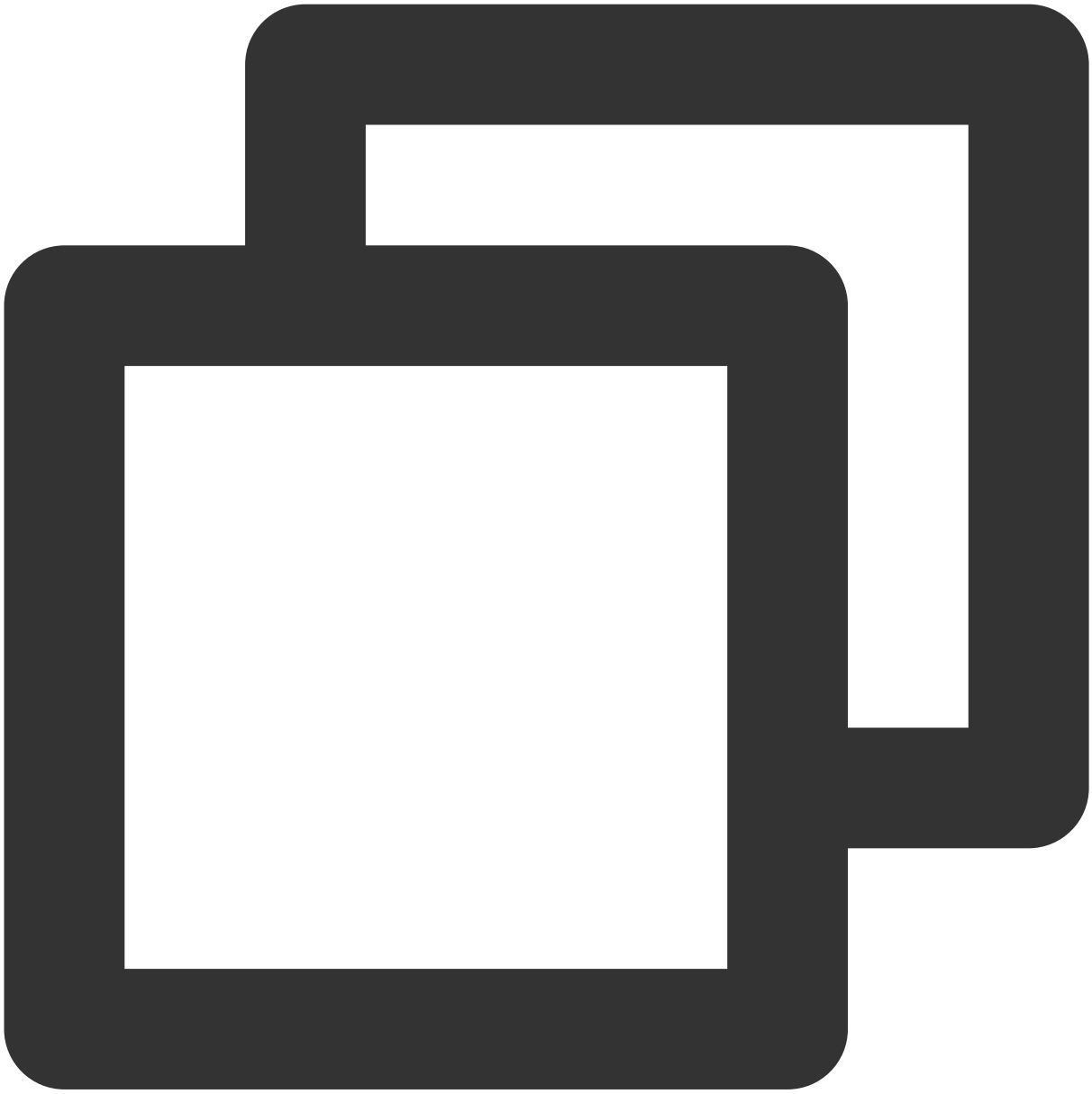


```
# 下载文件
## getObject(下载文件)
### 下载到内存
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key = "exampleobject";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $result = $cosClient->getObject(array(
        'Bucket' => $bucket,
        'Key' => $key));
    // 请求成功
    echo($result['Body']);
```

```
} catch (\\Exception $e) {
    // 请求失败
    echo "$e\\n";
}

### 下载到本地
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key = "exampleobject";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $localPath = @"path/to/localFile";//下载到本地指定路径
    $result = $cosClient->getObject(array(
        'Bucket' => $bucket,
        'Key' => $key,
        'SaveAs' => $localPath));
} catch (\\Exception $e) {
    // 请求失败
    echo "$e\\n";
}

### 指定下载范围
/*
 * Range 字段格式为 'bytes=a-b'
 */
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key = "exampleobject";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $localPath = @"path/to/localFile";//下载到本地指定路径
    $result = $cosClient->getObject(array(
        'Bucket' => $bucket,
        'Key' => $key,
        'Range' => 'bytes=0-10',
        'SaveAs' => $localPath));
} catch (\\Exception $e) {
    // 请求失败
    echo "$e\\n";
}

## getObjectUrl(获取文件 UrL)
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key = "exampleobject";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
    // 请求成功
    echo $signedUrl;
} catch (\\Exception $e) {
    // 请求失败
    print_r($e);
```

```
}
```

删除对象



```
# 删除 object
## deleteObject
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key = "exampleobject";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $result = $cosClient->deleteObject(array(
```

```
        'Bucket' => $bucket,
        'Key' => $key,
        'VersionId' => 'string'
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
# 删除多个 object
## deleteObjects
try {
    $bucket = "examplebucket-1250000000"; //存储桶，格式：BucketName-APPID
    $key1 = "exampleobject1";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $key2 = "exampleobject2";  //此处的 key 为对象键，对象键是对象在存储桶中的唯一标识
    $result = $cosClient->deleteObjects(array(
        'Bucket' => $bucket,
        'Objects' => array(
            array(
                'Key' => $key1,
            ),
            array(
                'Key' => $key2,
            ),
            //...
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

图片处理
基础图片处理

最近更新时间：2024-03-13 10:25:33

简介

腾讯云对象存储（Cloud Object Storage，COS）集成了 数据万象（Cloud Infinite，CI）专业的一体化多媒体解决方案，涵盖以下图片处理功能，详情可见 图片处理概述。

功能	说明
缩放	等比缩放、设定目标宽高缩放等多种方式
裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
旋转	自适应旋转、普通旋转
格式转换	格式转换、GIF 格式优化、渐进显示
质量变换	针对 JPG 和 WEBP 图片进行质量变换
高斯模糊	对图片进行模糊处理
亮度	调节图片亮度
对比度	调节图片对比度
锐化	对图片进行锐化处理
灰度图	将图片设置为灰度图
图片水印	提供图片水印处理功能
文字水印	提供文字水印处理功能
获取图片基本信息	查询图片基本信息，包括格式、长、宽等
获取图片 EXIF	获取 EXIF 信息
获取图片主色调	获取图片主色调信息
去除元信息	可去除图片元信息，包括 exif 信息
快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

版权所有：腾讯云计算（北京）有限责任公司

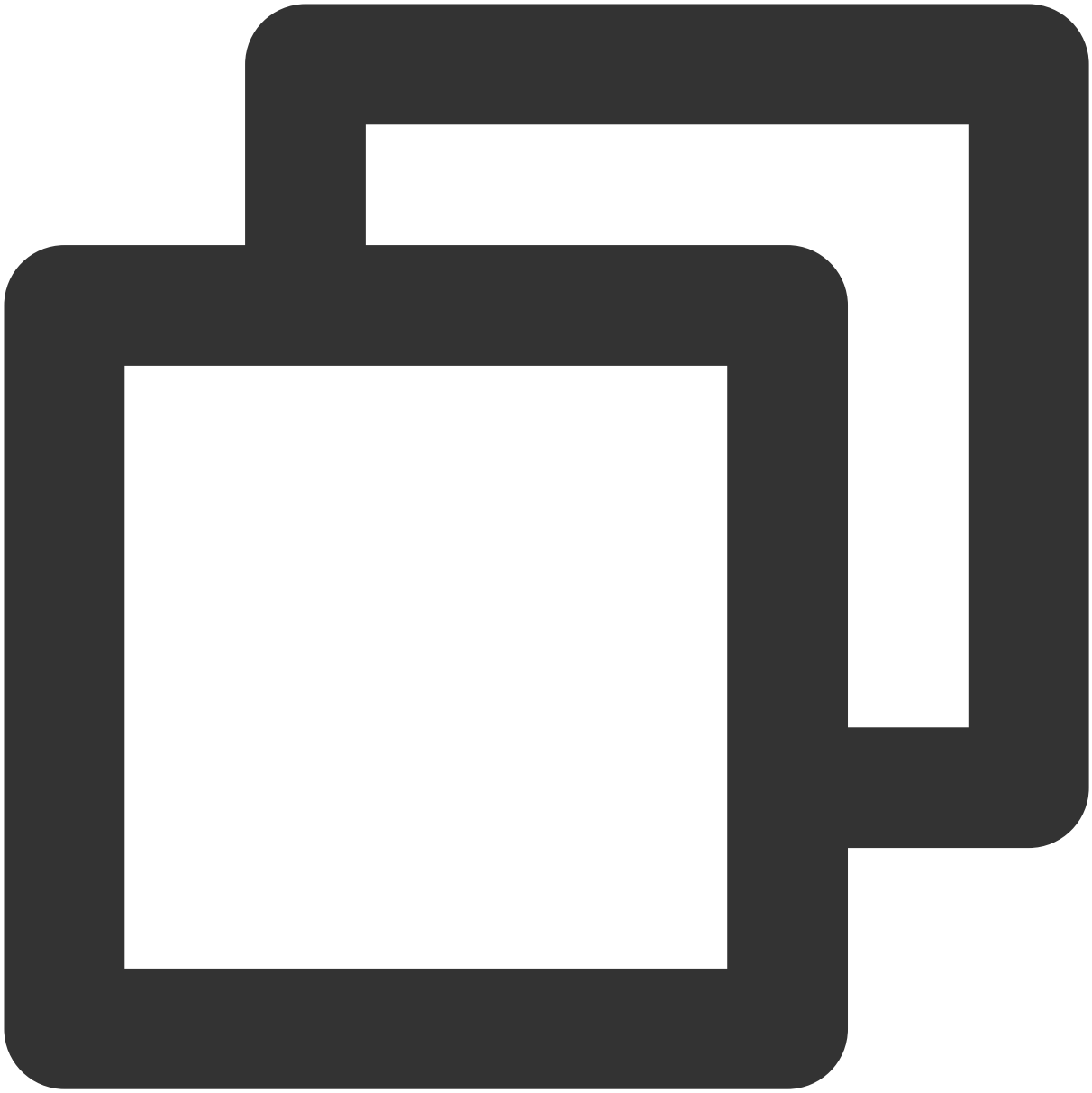
限制图片大小	可限制图片处理（例如缩放、压缩等）后的文件大小
管道操作符	实现对图片按顺序进行多种处理

缩放

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供图片缩放功能。处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-缩放 https://cloud.tencent.com/document/product/460/36540
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->thumbnailByScale(50); // /thumbnail/!<Scale>p
    $imageRule->thumbnailByWidthScale(50); // /thumbnail/!<Scale>px
    $imageRule->thumbnailByHeightScale(50); // /thumbnail/!x<Scale>p
    $imageRule->thumbnailByWidth(60); // /thumbnail/<Width>x
    $imageRule->thumbnailByHeight(60); // /thumbnail/x<Height>
    $imageRule->thumbnailByMaxWH(100, 100); // /thumbnail/<Width>x<Height>
    $imageRule->thumbnailByMinWH(30, 30); // /thumbnail/!<Width>x<Height>r
    $imageRule->thumbnailByWH(50, 50); // /thumbnail/<Width>x<Height>!
    $imageRule->thumbnailByPixel(10000); // /thumbnail/<Area>@
    $imageRule->thumbnailEqualRatioReduceByWH(300, 300); // /thumbnail/<Width>x<Hei
    $imageRule->thumbnailEqualRatioEnlargeByWH(300, 300); // /thumbnail/<Width>x<He
    $imageRule->pad(1); // /pad/1
    $imageRule->color('#3D3D3D'); // /color/IzNEM0QzRA
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
```

```
            'Key' => 'test.png',
            'PicOperations' => $picOperations->queryString(),
        ));
        // 请求成功
        print_r($result);
        // -------------------- 3. 云上数据处理 -------------------- //
    } catch (\\Exception $e) {
        // 请求失败
        echo($e);
    }
```

处理参数说明

操作名称：thumbnail。

参数	含义
ObjectKey	对象文件名，例如 folder/sample.jpg。
/thumbnail/!<Scale>p	指定图片的宽高为原图的 Scale%
/thumbnail/!<Scale>px	指定图片的宽为原图的 Scale%，高度不变
/thumbnail/!x<Scale>p	指定图片的高为原图的 Scale%，宽度不变
/thumbnail/<Width>x	指定目标图片宽度为 Width，高度等比缩放
/thumbnail/x<Height>	指定目标图片高度为 Height，宽度等比缩放
/thumbnail/<Width>x<Height>	限定缩略图的宽度和高度的最大值分别为 Width 和 Height，进行等比缩放
/thumbnail/<Width>x<Height>>	限定缩略图的宽度和高度的最大值分别为 Width 和 Height，进行等比缩小，比例值为宽缩放比和高缩放比的较小值，如果目标宽（高）都大于原图宽（高），则不变
/thumbnail/<Width>x<Height><	限定缩略图的宽度和高度的最大值分别为 Width 和 Height，进行等比放大，比例值为宽缩放比和高缩放比的较小值。如果目标宽(高)小于原图宽(高)，则不变
/thumbnail/!<Width>x<Height>r	限定缩略图的宽度和高度的最小值分别为 Width 和 Height，进行等比缩放
/thumbnail/<Width>x<Height>!	忽略原图宽高比例，指定图片宽度为 Width，高度为 Height，强行缩放图片，可能导致目标图片变形
/thumbnail/<Area>@	等比缩放图片，缩放后的图像，总像素数量不超过 Area
/pad/	将原图缩放为指定 Width 和 Height 的矩形内的最大图片，之后使用 color 参数指定的颜色居中填充空白部分；取值0或1，0代表不使用 pad 模式，1代表

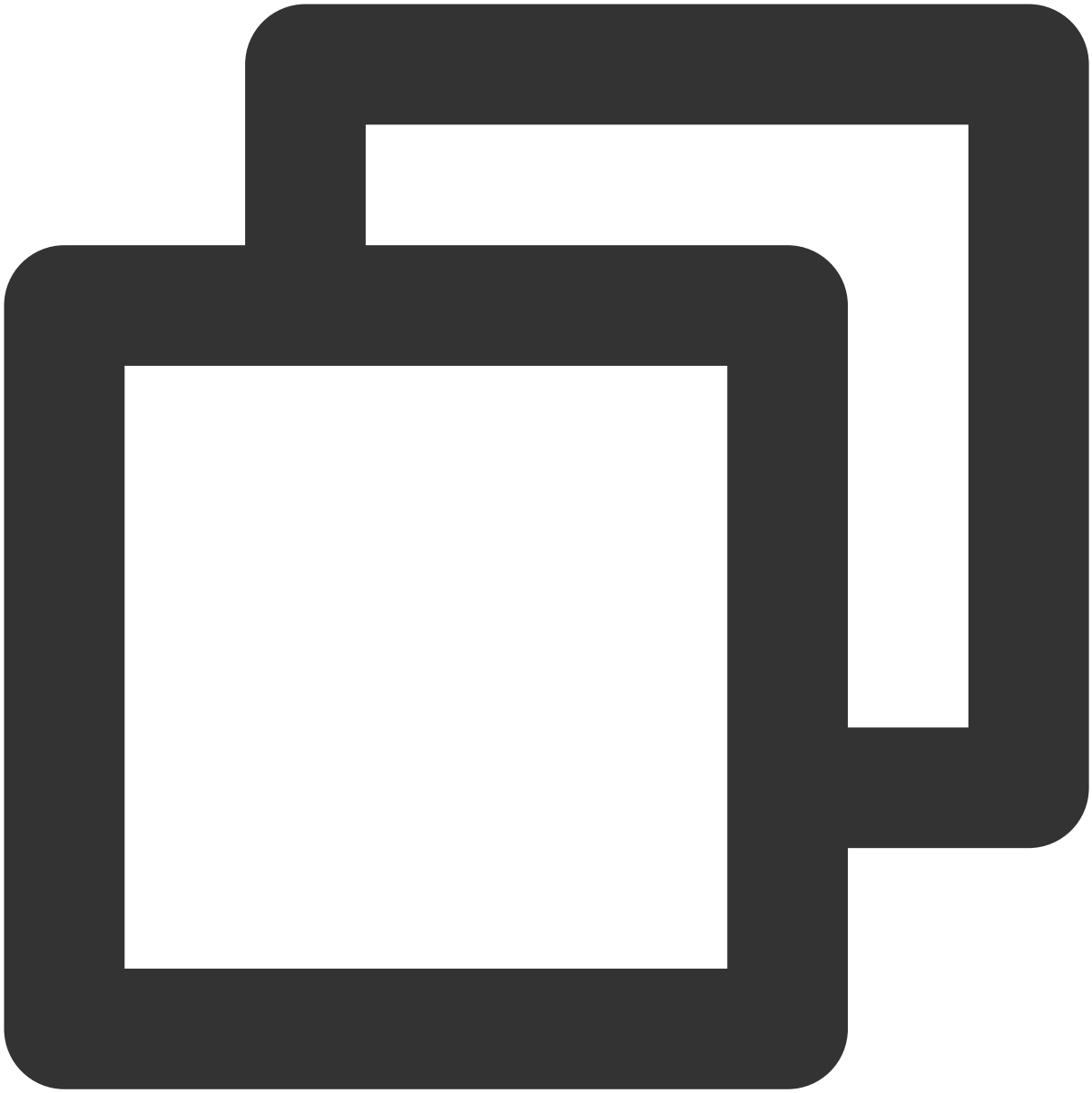
	使用 pad 模式
/color/	填充颜色，缺省为白色，需设置为十六进制 RGB 格式（如 #FF0000），详情参见 RGB 编码表，需经过 URL 安全的 Base64 编码，默认值为 #FFFFFF
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

裁剪

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供裁剪功能，包括普通裁剪、缩放裁剪、内切圆裁剪、圆角裁剪和人脸智能裁剪。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-裁剪 https://cloud.tencent.com/document/product/460/36541
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->cut(400, 400, 100, 100); // 普通裁剪参数 /cut/<width>x<height>x<dx>x<
    $imageRule->cropByWidth(400, 'center'); // 缩放裁剪参数 /crop/<Width>x
    $imageRule->cropByHeight(400, 'center'); // 缩放裁剪参数 /crop/x<Height>
    $imageRule->cropByWH(400, 400, 'center'); // 缩放裁剪参数 /crop/<Width>x<Height>
    $imageRule->iradius(30); // 内切圆裁剪参数 /iradius/<radius>
    $imageRule->rradius(30); // 圆角裁剪参数 /rradius/<radius>
    $imageRule->scrop(30, 30); // 人脸智能裁剪参数 /scrop/<Width>x<Height>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
```

```
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

参数	含义
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

普通裁剪参数说明

操作名称：cut。

参数	含义
<width>	指定目标图片的宽为 width
<height>	指定目标图片的高为 height
<dx>	相对于图片左上顶点水平向右偏移 dx
<dy>	相对于图片左上顶点水平向下偏移 dy

说明：

参数取值范围应大于0，小于原图宽高。

缩放裁剪参数说明

操作名称：crop。

参数	含义
/crop/<Width>x	指定目标图片宽度为 Width，高度不变。Width 取值范围应大于0，小于原图宽度
/crop/x<Height>	指定目标图片高度为 Height，宽度不变。Height 取值范围应大于0，小于原图高度
/crop/<Width>x<Height>	指定目标图片宽度为 Width，高度为 Height 。Width 和 Height 取值范围都应大于0，小于原图宽度/高度

在进行缩放裁剪操作时，您也可以使用 gravity 参数指定操作的起点位置，详情请参见下文中 缩放裁剪操作示例。

内切圆裁剪参数说明

操作名称：iradius。

参数	含义
/iradius/<radius>	内切圆裁剪功能，radius 是内切圆的半径，取值范围为大于0且小于原图最小边一半的整数。内切圆的圆心为图片的中心。图片格式为 gif 时，不支持该参数。

圆角裁剪参数说明

操作名称：rradius。

参数	含义
/rradius/<radius>	圆角裁剪功能，radius 为图片圆角边缘的半径，取值范围为大于0且小于原图最小边一半的整数。圆角与原图边缘相切。图片格式为 gif 时，不支持该参数。

人脸智能裁剪参数说明

操作名称：scrop。

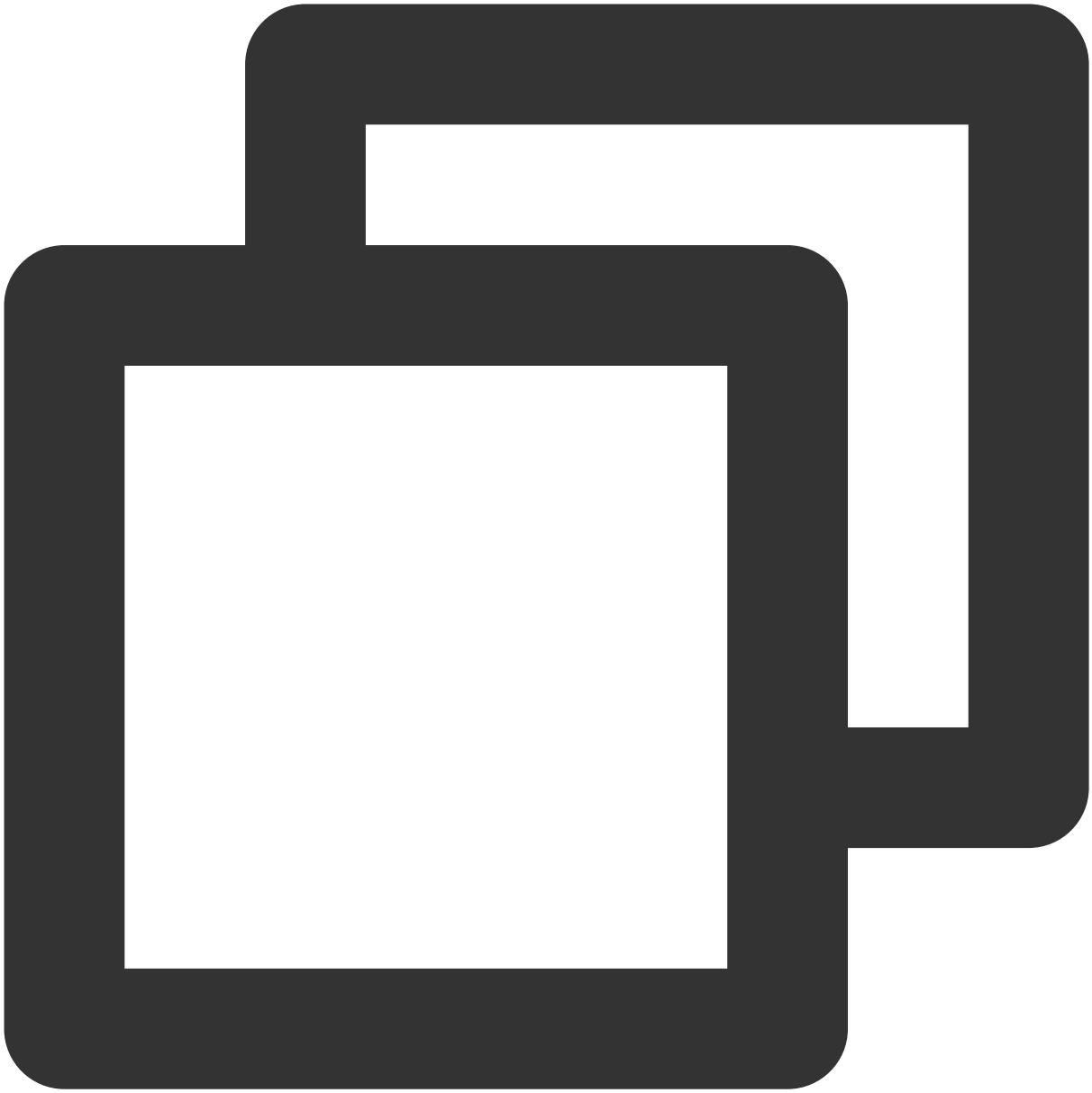
参数	含义
/scrop/<Width>x<Height>	基于图片中的人脸位置进行缩放裁剪。目标图片的宽度为 Width、高度为 Height。

旋转

功能说明

腾讯云数据万象通过 **imageMogr2** 接口提供旋转功能，包括普通旋转和自适应旋转。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-旋转 https://cloud.tencent.com/document/product/460/36542
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->rotate(90); // /rotate/<rotateDegree>
    $imageRule->autoOrient(); // /auto-orient
    $imageRule->flip('vertical'); // /flip/<flip>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称, 由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称, 由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

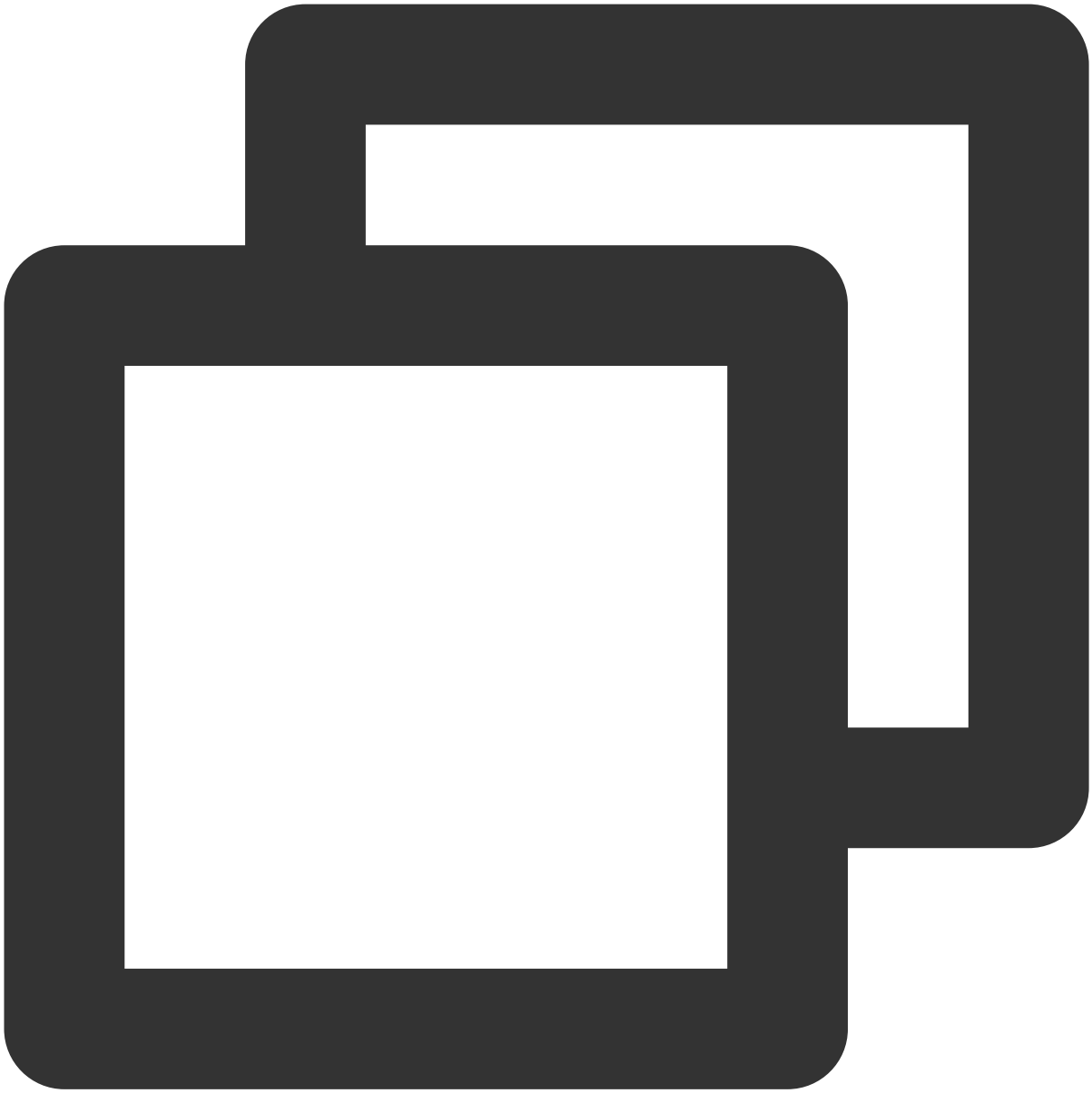
参数	含义
/rotate/<rotateDegree>	普通旋转：图片顺时针旋转角度，取值范围0 - 360，默认不旋转。
/auto-orient	自适应旋转：根据原图 EXIF 信息将图片自适应旋转回正。
/flip/<flip>	镜像翻转：flip 值为 vertical 表示垂直翻转，horizontal 表示水平翻转
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

格式转换

功能说明

腾讯云数据万象通过 imageMogr2 接口提供格式转换、gif 格式优化、渐进显示功能。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-格式转换 https://cloud.tencent.com/document/product/460/36543
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->format('jpg'); // 格式转换 /format/<Format>
    $imageRule->gifOptimization(1); // gif 格式优化 /cgif/<FrameNumber>
    $imageRule->jpegInterlaceMode(1); // 输出为渐进式 jpg 格式。Mode 可为0或1 /interlac
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

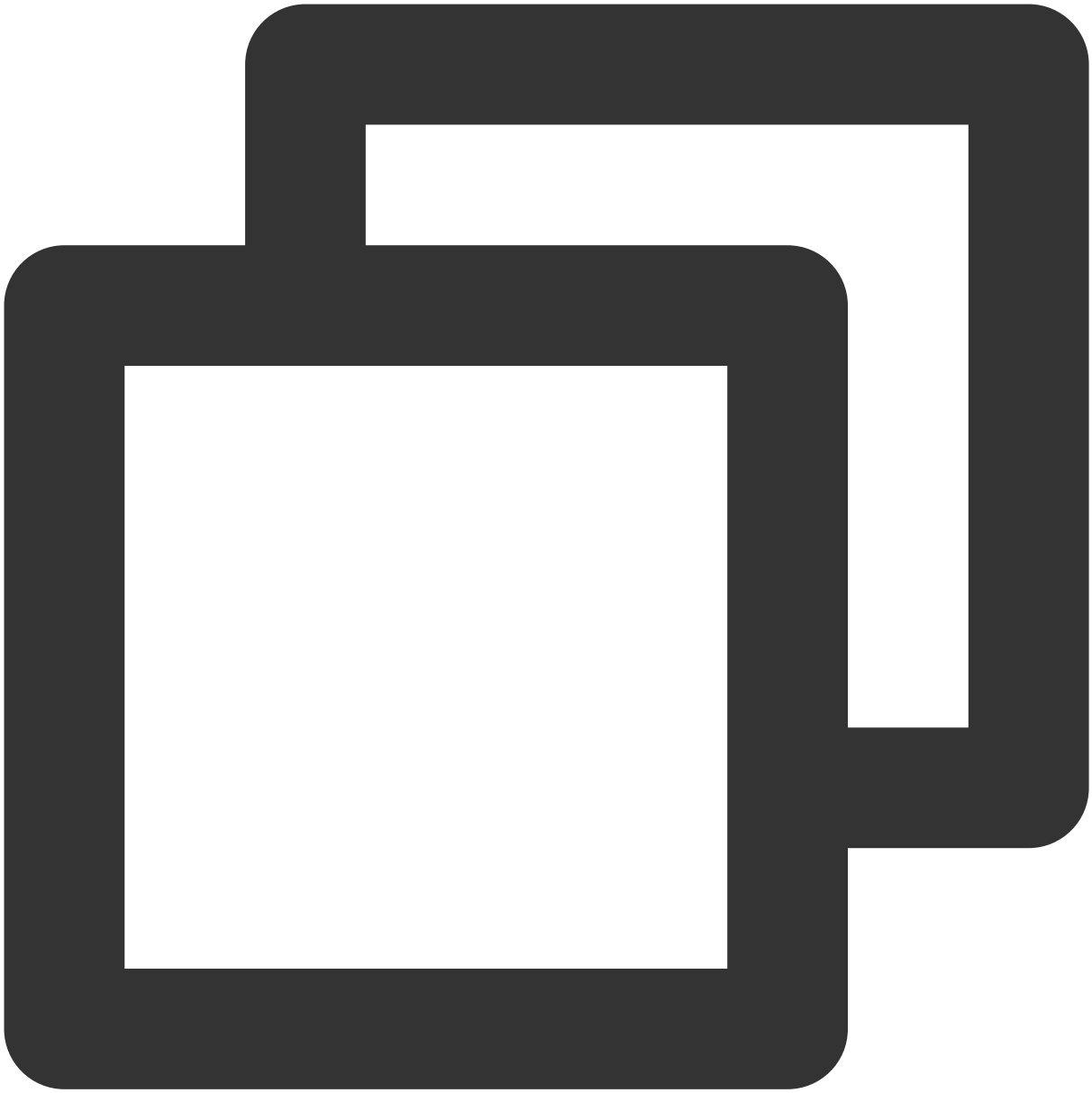
参数	含义
/format/<Format>	格式转换：目标缩略图的图片格式可为：jpg，bmp，gif，png，webp，yjpeg 等，其中 yjpeg 为数据万象针对 jpeg 格式进行的优化，本质为 jpg 格式；缺省为原图格式。
/cgif/<FrameNumber>	gif 格式优化：只针对原图为 gif 格式，对 gif 图片格式进行的优化，降帧降颜色。分为以下两种情况：FrameNumber=1，则按照默认帧数30处理，如果图片帧数大于该帧数则截取。FrameNumber 取值(1,100]，则将图片压缩到指定帧数（FrameNumber）。
/interlace/<Mode>	输出为渐进式 jpg 格式。Mode 可为0或1。0：表示不开启渐进式；1：表示开启渐进式。该参数仅在输出图片格式为 jpg 格式时有效。如果输出非 jpg 图片格式，会忽略该参数，默认值0。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

质量变换

功能说明

腾讯云数据万象通过 imageMogr2 接口对图片质量进行调节。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-质量变换 https://cloud.tencent.com/document/product/460/36544
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->quality(90, 1); // 绝对质量 /quality/<Quality>
    $imageRule->lowestQuality(90); // 最低质量 /lquality/<quality>
    $imageRule->relativelyQuality(90); // 相对质量 /rquality/<quality>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：quality。

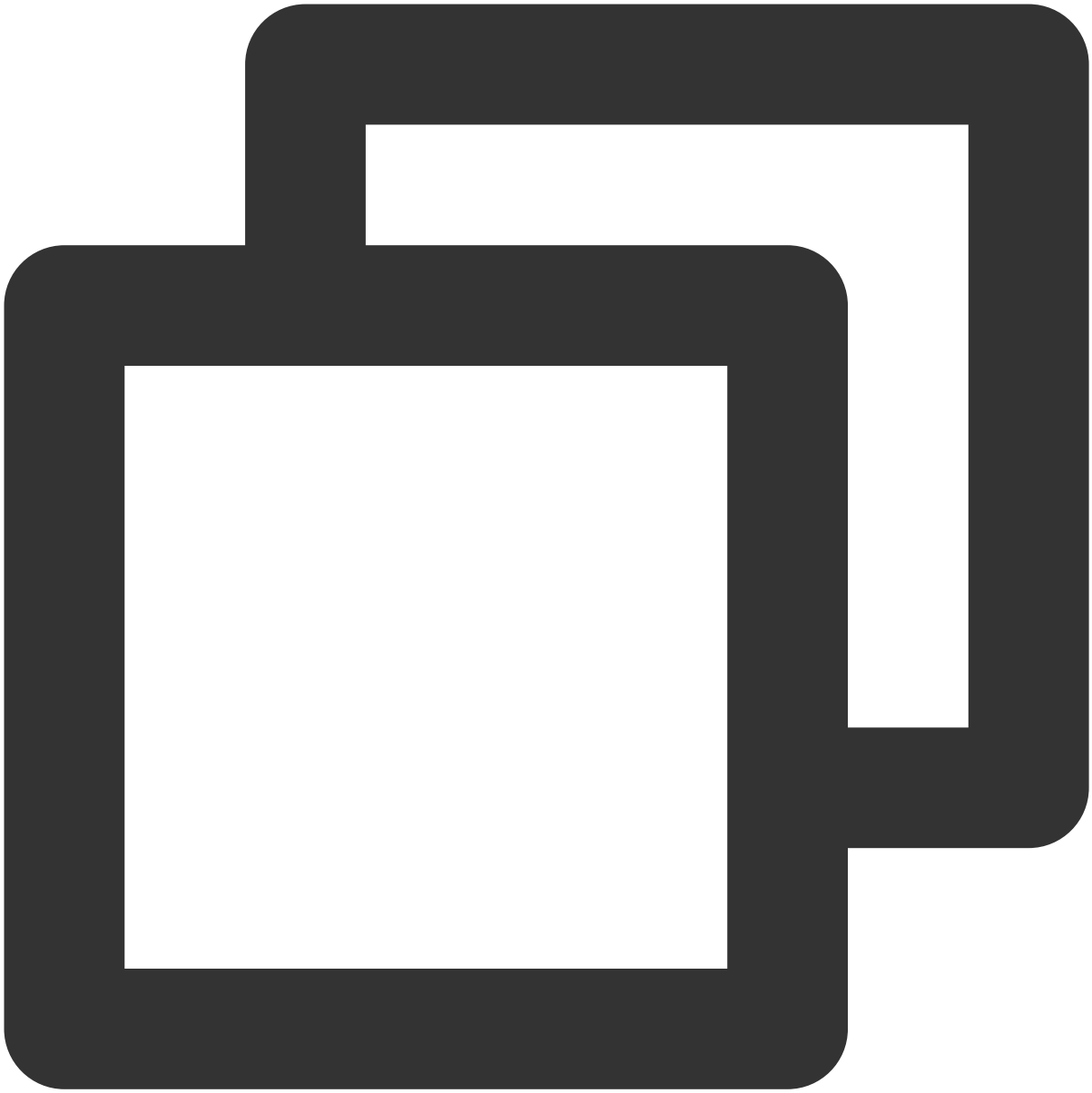
参数	含义
/quality/<Quality>	图片的绝对质量，取值范围0 - 100，默认值为原图质量；取原图质量和指定质量的最小值；<Quality>后面加“!”表示强制使用指定值，例如：`90!`。
/rquality/<quality>	图片的相对质量，取值范围0 - 100，数值以原图质量为标准。例如原图质量为80，将 rquality 设置为80后，得到处理结果图的图片质量为64（80x80%）。
/lquality/<quality>	图片的最低质量，取值范围0 - 100，设置结果图的质量参数最小值。 例如原图质量为85，将 lquality 设置为80后，处理结果图的图片质量为85。例如原图质量为60，将 lquality 设置为80后，处理结果图的图片质量会被提升至80。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

高斯模糊

功能说明

腾讯云数据万象通过 **imageMogr2** 接口对图片进行模糊处理。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 图片处理-高斯模糊 https://cloud.tencent.com/document/product/460/36545
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->blur(8, 5); // 高斯模糊 /blur/<radius>x<sigma>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：blur。

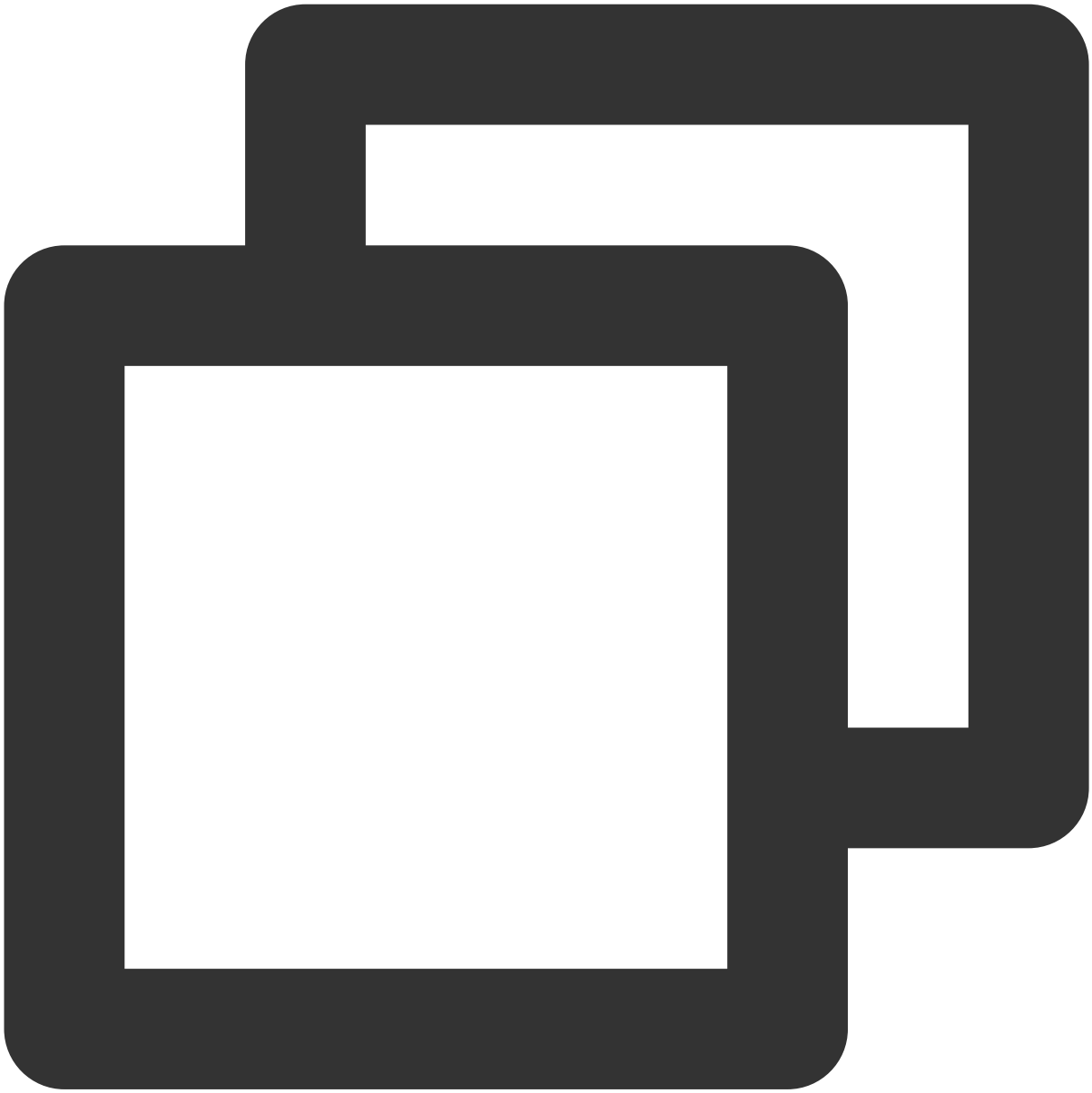
参数	含义
radius<radius>	模糊半径，取值范围为1 - 50
sigma<sigma>	正态分布的标准差，必须大于0
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错

亮度

功能说明

腾讯云数据万象通过 **imageMogr2** 接口调节图片亮度。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-亮度 https://cloud.tencent.com/document/product/460/51808
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->bright(70); // 图片亮度调节 /bright/<value>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：bright。

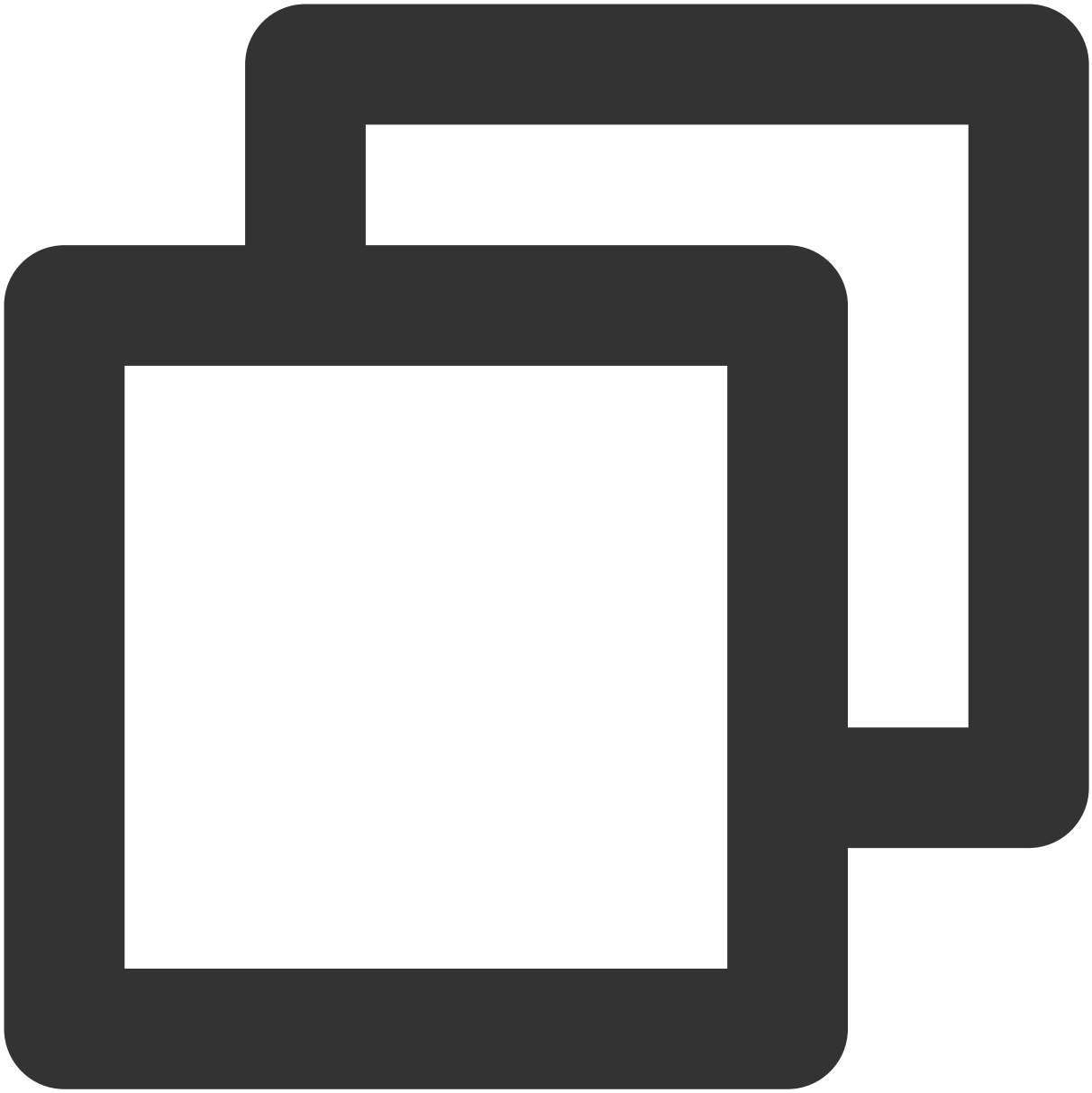
参数	含义
/bright/<value>	图片亮度调节功能，value 为亮度参数值，取值范围为[-100, 100]的整数。 取值＜0：降低图片亮度。取值 = 0：不调整图片亮度。取值＞0：提高图片亮度。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错。

对比度

功能说明

对比度指一幅图像中明暗区域最亮的白和最暗的黑之间不同亮度层级的测量，即指一幅图像灰度反差的大小。腾讯云数据万象通过 **imageMogr2** 接口调节图片对比度。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-对比度 https://cloud.tencent.com/document/product/460/51809
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->contrast(-70); // 图片对比度调节 /contrast/<value>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：contrast。

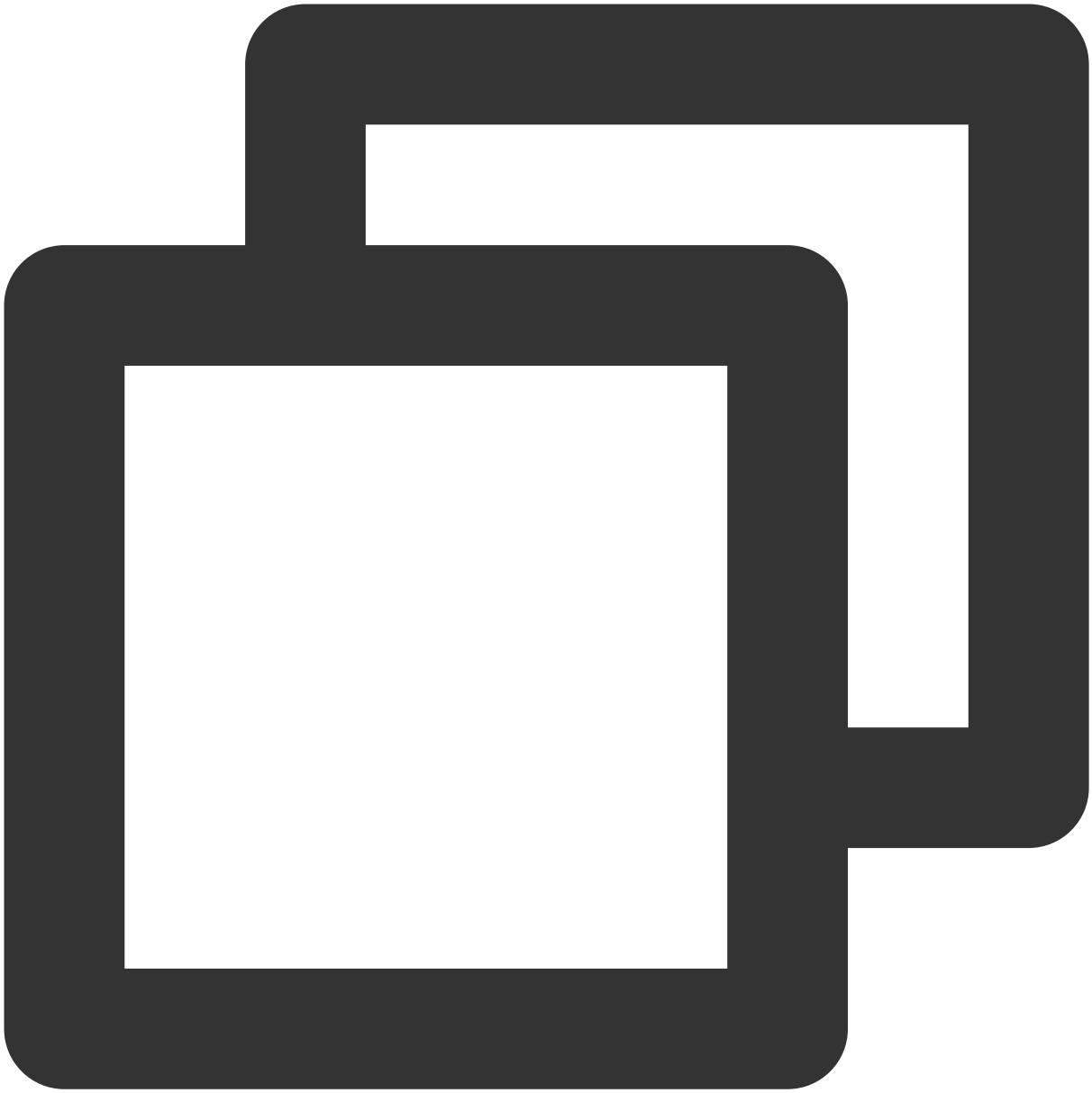
参数	含义
/contrast/<value>	图片对比度调节功能，value 为对比度参数值，取值范围为[-100, 100]的整数。 取值＜0：降低图片对比度。取值 = 0：不调整图片对比度。取值＞0：提高图片对比度。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错。

锐化

功能说明

腾讯云数据万象通过 **imageMogr2** 接口对图片进行锐化处理。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-锐化 https://cloud.tencent.com/document/product/460/36546
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->sharpen(70); // 图片锐化 /sharpen/<value>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：sharpen。

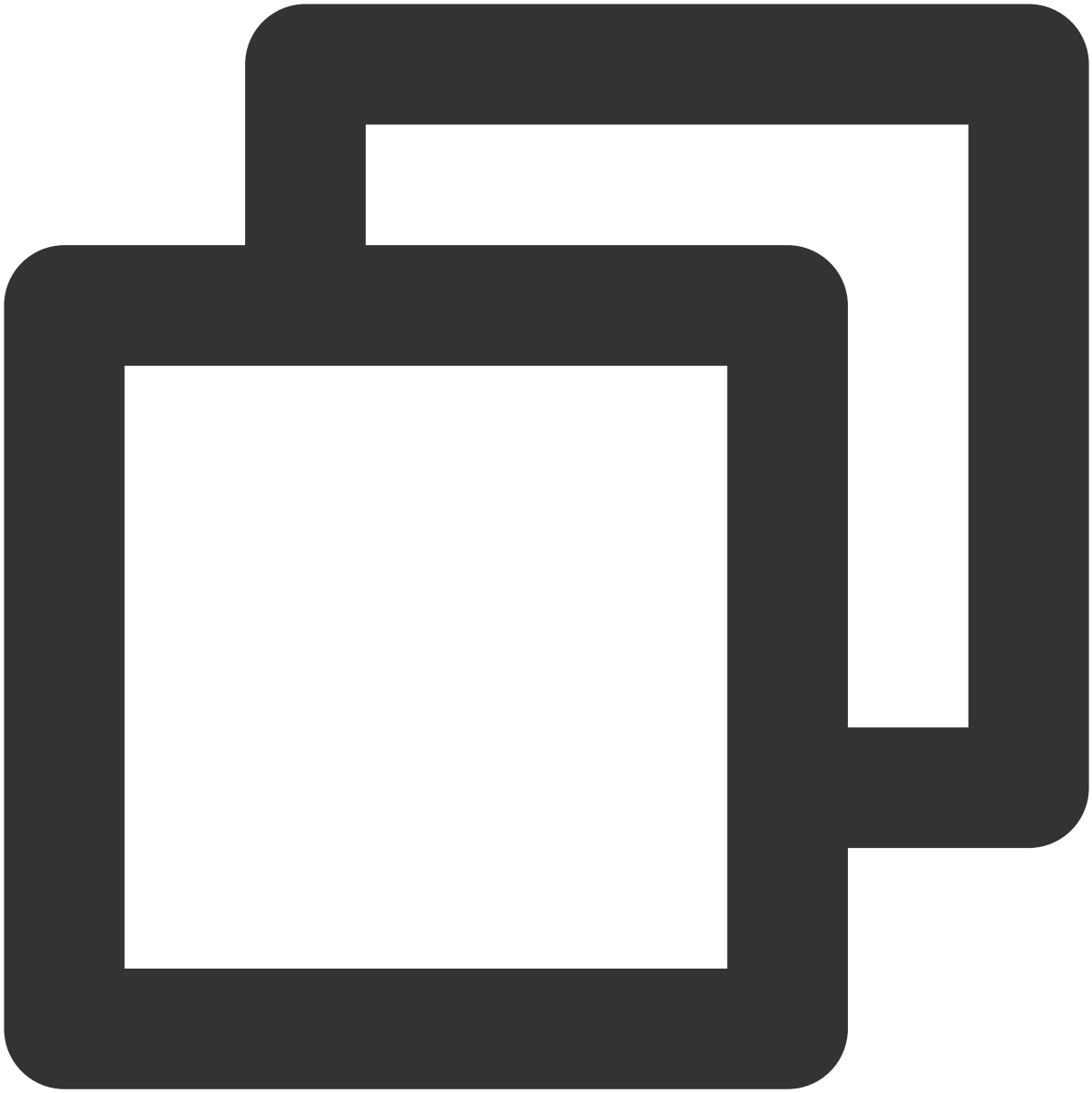
参数	含义
/sharpen/<value>	图片锐化功能，value 为锐化参数值，取值范围为10 - 300间的整数（推荐使用70）。参数值越大，锐化效果越明显。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大、参数超限等导致处理失败的场景，会直接返回原图而不报错。

灰度图

功能说明

对象存储通过数据万象 **imageMogr2/grayscale** 接口将图片设置为灰度图。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-灰度图 https://cloud.tencent.com/document/product/460/66519
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->grayscale(1); // 将图片设置为灰度图 /grayscale/<value>
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：grayscale。

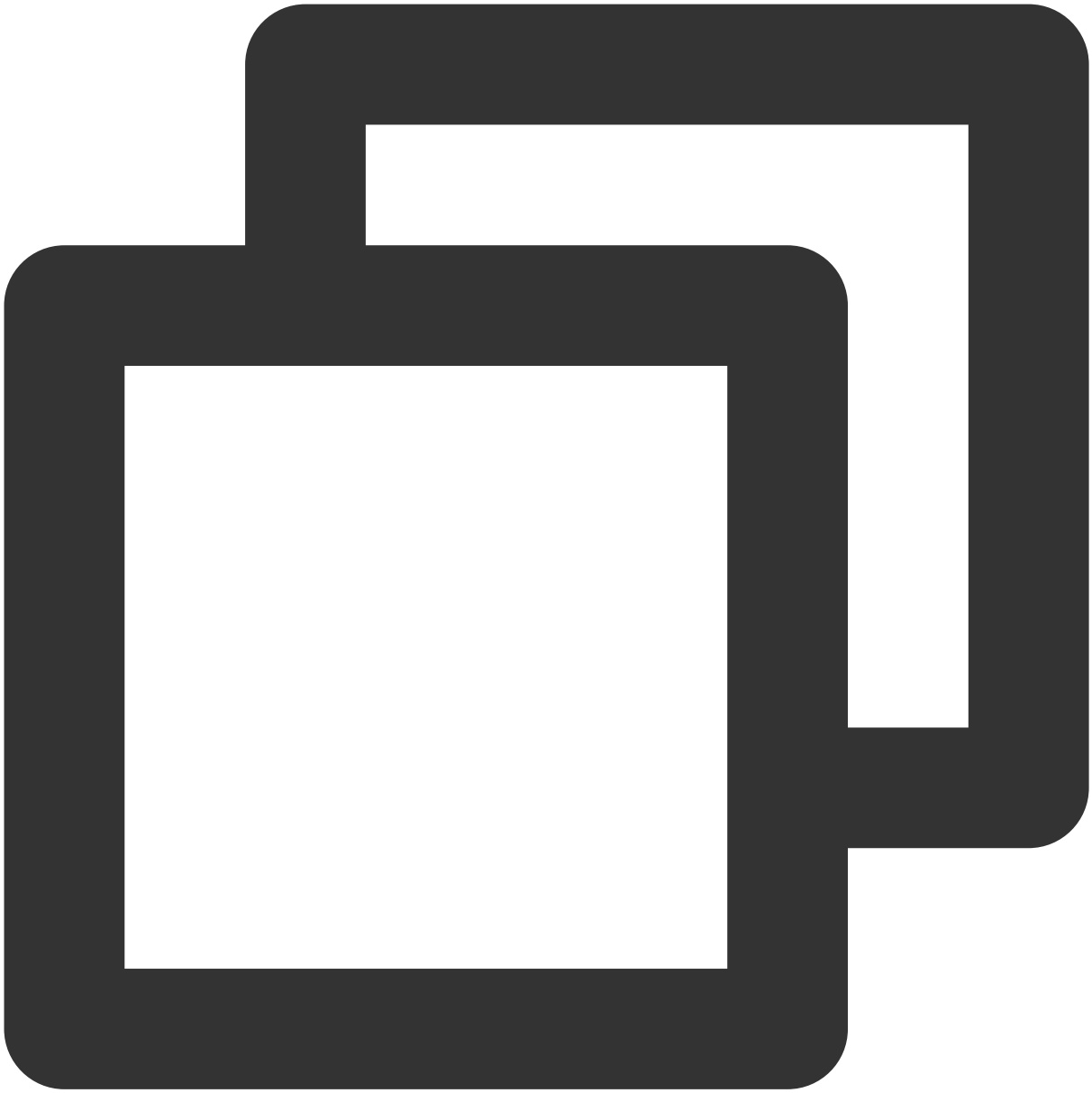
参数	含义
/grayscale/<value>	将图片设置为灰度图。 value 取值为0表示不改变图片。 value 取值为1表示将图片变为灰度图。
/ignore-error/1	当处理参数中携带此参数时，针对文件过大导致处理失败的场景，会直接返回原图而不报错。

图片水印

功能说明

对象存储通过数据万象 **watermark** 接口提供图片水印处理功能。目前水印图片必须指定为已存储于数据万象绑定（或新建）的存储桶中的图片。关于图片水印等图片基础处理的限制说明，请参见 规则与限制。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-图片水印 https://cloud.tencent.com/document/product/460/6930
    $imageWatermarkRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageWatermarkTempla
    $imageWatermarkRule->setImage('https://www.xxx.com/xxx.jpg'); // 水印图片地址
    $imageWatermarkRule->setGravity('SouthEast'); // 图片水印位置
    $imageWatermarkRule->setDx(10); // 水平（横轴）边距，单位为像素，缺省值为0
    $imageWatermarkRule->setDy(10); // 垂直（纵轴）边距，单位为像素，默认值为0
    $imageWatermarkRule->setBlogo(1); // 水印图适配功能，适用于水印图尺寸过大的场景（如水印
    $imageWatermarkRule->setScatype(1); // 根据原图的大小，缩放调整水印图的大小
    $imageWatermarkRule->setSpcent(200); // 与 scatype 搭配使用
    $imageWatermarkRule->setDissolve(70); // 图片水印的透明度
    $imageWatermarkRule->setBatch(1); // 平铺水印功能，可将图片水印平铺至整张图片
    $imageWatermarkRule->setDegree(90); // 当 batch 值为1时生效。图片水印的旋转角度设置

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageWatermarkRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageWatermarkRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
```

```
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：watermark，数字1，表示水印类型为图片水印。

参数	含义
/image/	水印图片地址，需要经过 URL 安全的 Base64 编码。例如，水印图片为 `http://examplebucket` `shanghai.myqcloud.com/shuiyin_2.png` ，则该处编码后的字符串为 `aHR0cDovL2V4YW1wbGVidWNrZXQtMTI1MDAwMDAwMC5jb3MuYXAtc2hhbmdoYWkubXlxY2xv`
/gravity/	图片水印位置，九宫格位置（参考九宫格方位图 ），默认值为 SouthEast
/dx/	水平（横轴）边距，单位为像素，缺省值为0
/dy/	垂直（纵轴）边距，单位为像素，默认值为0
/blogo/	水印图适配功能，适用于水印图尺寸过大的场景（如水印墙）。一共有两种类型： 当 blogo 设置为1时，水印图会被直接裁剪至与原图相似大小后添加 当 blogo 设置为2时，水印图会被直接裁剪至与原图相似大小后添加
/scatype/	根据原图的大小，缩放调整水印图的大小： 当 scatype 设置为1时，按原图的宽缩放 当 scatype 设置为2时，按原图的高缩放 当 scatype 设置为3时，按原图的整体面积缩放
/spcent/	与 scatype 搭配使用：当 scatype 设置为1时，该有效值为[1, 1000]，单位为千分比当 scatype 设置为2时，该有效值为[1, 100]，单位为百分比当 scatype 设置为3时，该有效值为[1, 250]，单位为千分比。示例： `http://examples-125` `shanghai.myqcloud.com/sample.jpeg?watermark/1/image/xxxxxxx/scatype/3/spc`
/dissolve/	图片水印的透明度，取值为1 - 100，默认90（90%不透明度）
/batch/	平铺水印功能，可将图片水印平铺至整张图片。值为1时，表示开启平铺水印功能
/degree/	当 batch 值为1时生效。图片水印的旋转角度设置，取值范围为0 - 360，默认0

注意：

指定的水印图片必须同时满足如下3个条件：

水印图片与源图片必须位于同一个存储桶下。

URL 需使用 COS 域名（不能使用 CDN 加速域名，例如 `examplebucket-1250000000.file.myqcloud.com/shuiyin_2.png` 不可用 ），且需保证水印图可访问（如果水印图读取权限为私有，则需要携带有效签名）。

版权所有：腾讯云计算（北京）有限责任公司

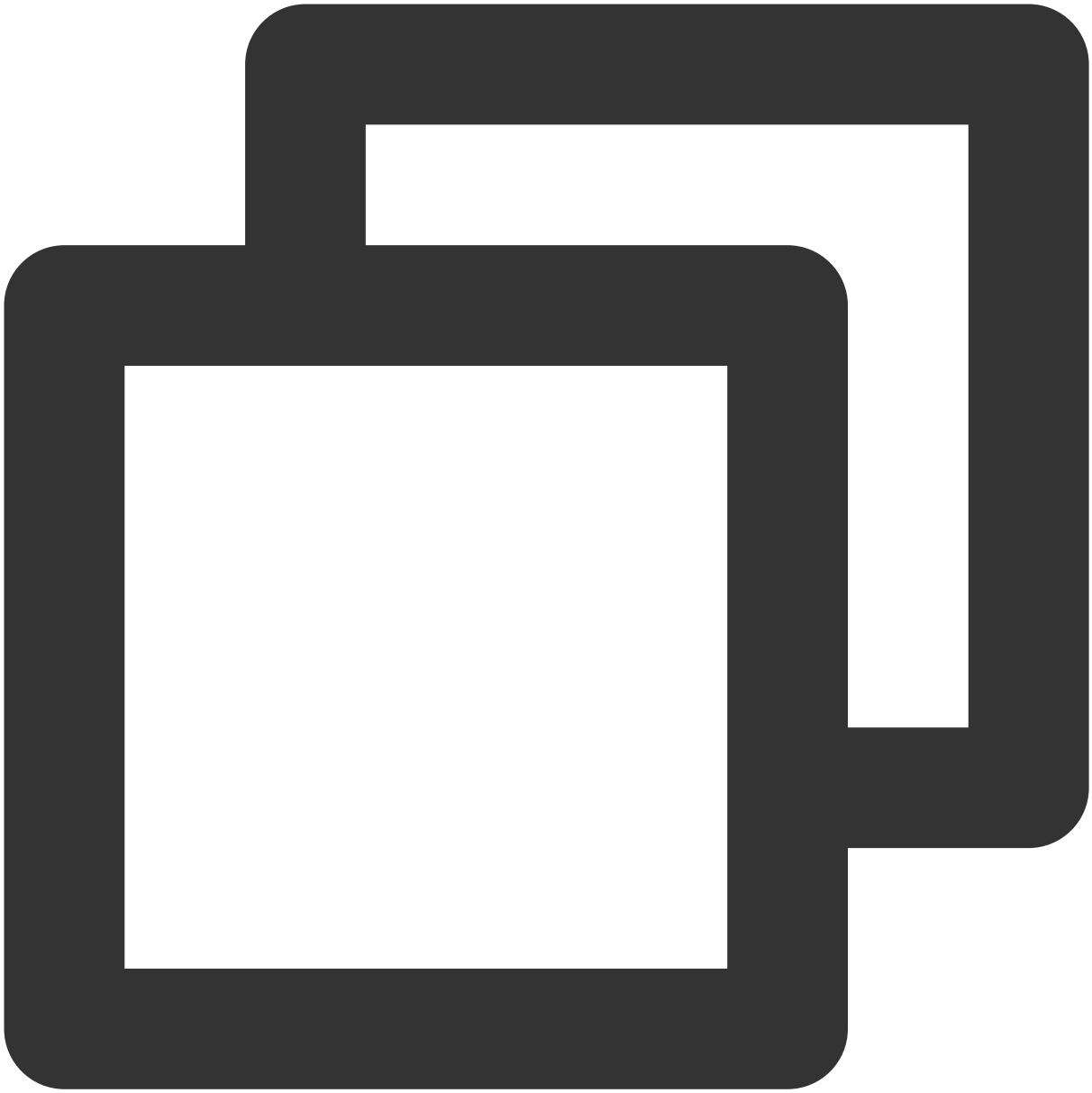
URL 必须以 `http://` 开始，不能省略 HTTP 头，也不能填 HTTPS 头，例如 `examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png`，`https://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` 为非法的水印 URL。

文字水印

功能说明

对象存储通过数据万象 **watermark** 接口提供实时文字水印处理功能。处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-文字水印 https://cloud.tencent.com/document/product/460/6951
    $textWatermarkRule = new Qcloud\\Cos\\ImageParamTemplate\\TextWatermarkTemplate
    $textWatermarkRule->setText('水印内容'); // 水印内容
    $textWatermarkRule->setFont('tahoma.ttf'); // 水印字体
    $textWatermarkRule->setFontsize(13); // 水印文字字体大小
    $textWatermarkRule->setFill('#3D3D3D'); // 字体颜色，缺省为灰色，需设置为十六进制 RGB
    $textWatermarkRule->setDissolve(90); // 文字透明度，取值1 - 100 ，默认90（90%不透明度
    $textWatermarkRule->setGravity('SouthEast'); // 文字水印位置，九宫格位置（参见九宫格方
    $textWatermarkRule->setDx(10); // 水平（横轴）边距，单位为像素，缺省值为0
    $textWatermarkRule->setDy(10); // 垂直（纵轴）边距，单位为像素，默认值为0
    $textWatermarkRule->setBatch(1); // 平铺水印功能，可将文字水印平铺至整张图片。值为1时，
    $textWatermarkRule->setDegree(10); // 当 batch 值为1时生效。文字水印的旋转角度设置，取
    $textWatermarkRule->setShadow(10); // 文字阴影效果，有效值为[0,100]，默认为0，表示无阴

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($textWatermarkRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $textWatermarkRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
```

```
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：watermark，数字2，表示水印类型为文字水印。

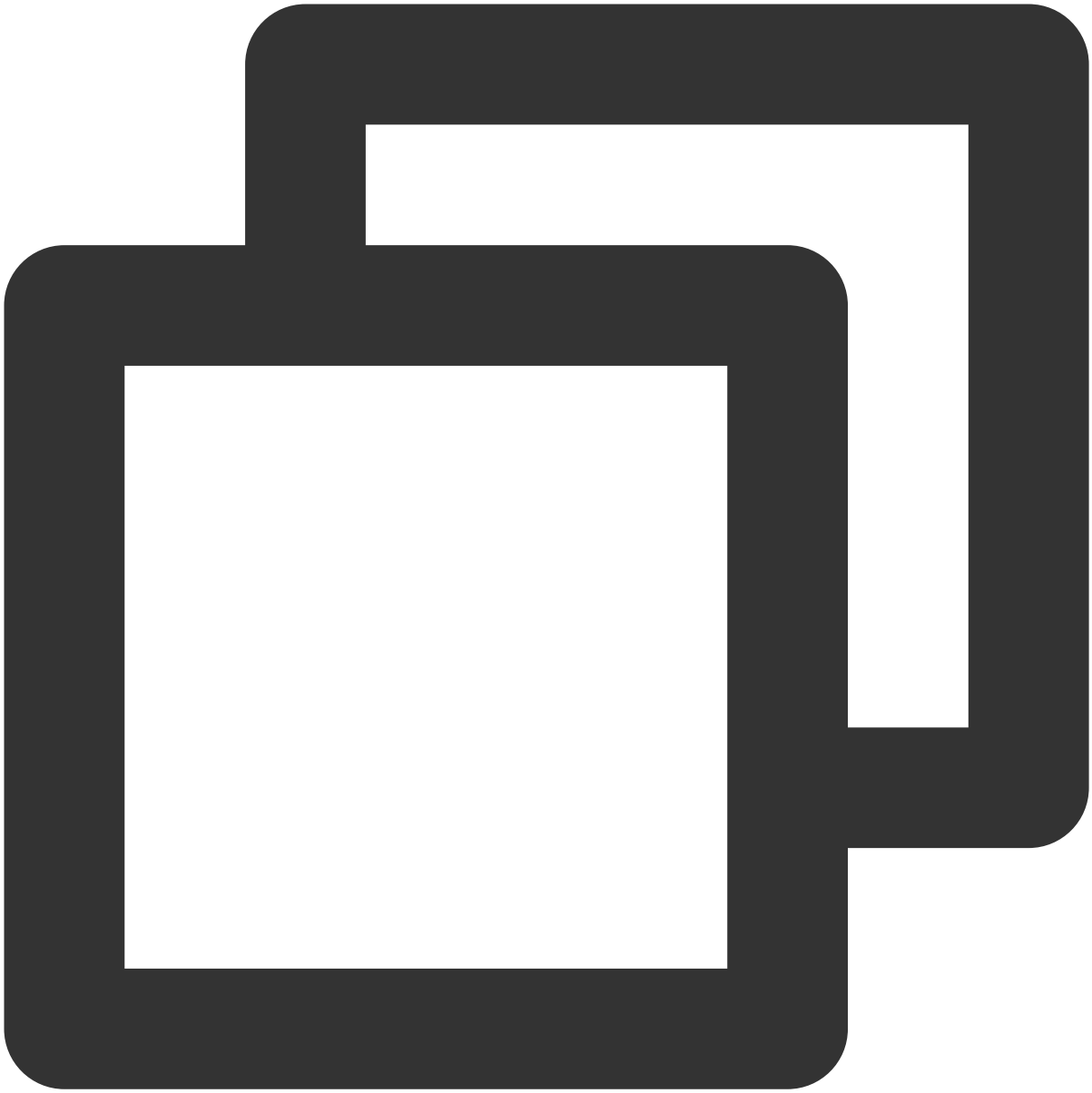
参数	含义
/text/	水印内容，需要经过 URL 安全的 Base64 编码
/font/	水印字体，需要经过 URL 安全的 Base64 编码，默认值 tahoma.ttf 。水印字体列表参考 支持字体列表
/fontsize/	水印文字字体大小，单位为磅，缺省值13。如需按照原图片的大小比例缩放文字水印，请将文字水印转换为 PNG 图片，并参考 图片水印 文档进行配置
/fill/	字体颜色，缺省为灰色，需设置为十六进制 RGB 格式（例如 #FF0000），详情可参考 RGB 编码表，需经过 URL 安全的 Base64 编码，默认值为 #3D3D3D
/dissolve/	文字透明度，取值1 - 100，默认90（90%不透明度）
/gravity/	文字水印位置，九宫格位置（参见九宫格方位图），默认值 SouthEast
/dx/	水平（横轴）边距，单位为像素，缺省值为0
/dy/	垂直（纵轴）边距，单位为像素，默认值为0
/batch/	平铺水印功能，可将文字水印平铺至整张图片。值为1时，表示开启平铺水印功能
/degree/	当 batch 值为1时生效。文字水印的旋转角度设置，取值范围为0 - 360，默认0
/shadow/	文字阴影效果，有效值为[0,100]，默认为0，表示无阴影

获取图片基本信息

功能说明

对象存储通过数据万象 **imageInfo** 接口查询图片基本信息，包括格式、长、宽等。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->ImageInfo(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'exampleobject',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：imageInfo。

参数	含义
Key	对象文件名，例如 folder/sample.jpg。

响应参数

参数	含义
format	图片类型。例如 png、gif 等
width	图片的宽度。单位为像素（px）
height	图片的高度。单位为像素（px）
size	图片的大小。单位为 Bytes
md5	图片的 md5 值
frame_count	图片的帧数。静态图为1，动图为对应的帧数

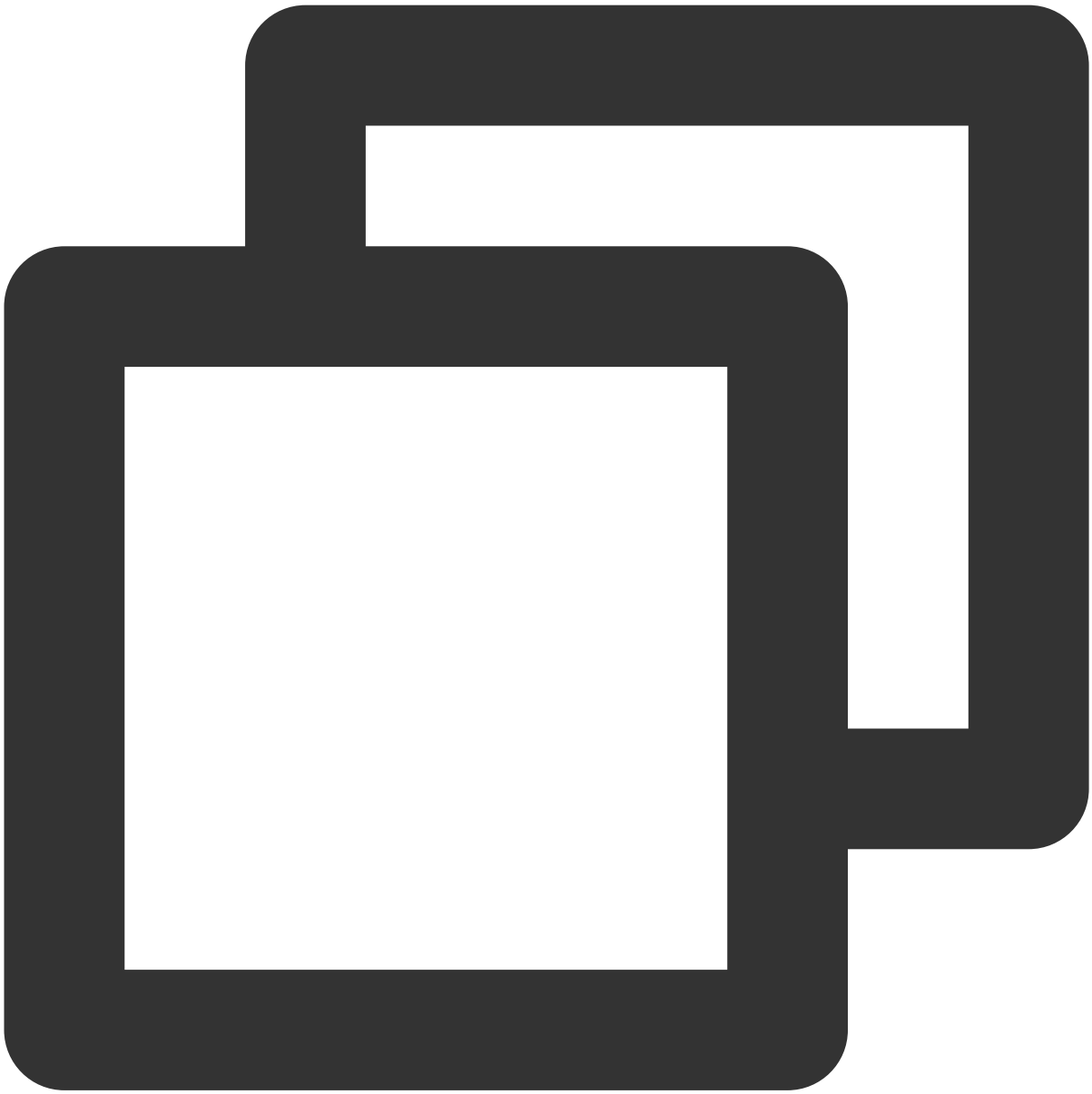
获取图片 EXIF

功能说明

EXIF（Exchangeable Image File）全称为可交换图像文件，可记录数码照片的拍摄参数、缩略图及其他属性信息。对象存储通过数据万象的 **exif** 接口获取 EXIF 信息。处理图片原图大小不超过32MB、宽高不超过30000像素且总像

素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
```

```
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->ImageExif(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'exampleobject',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：exif。

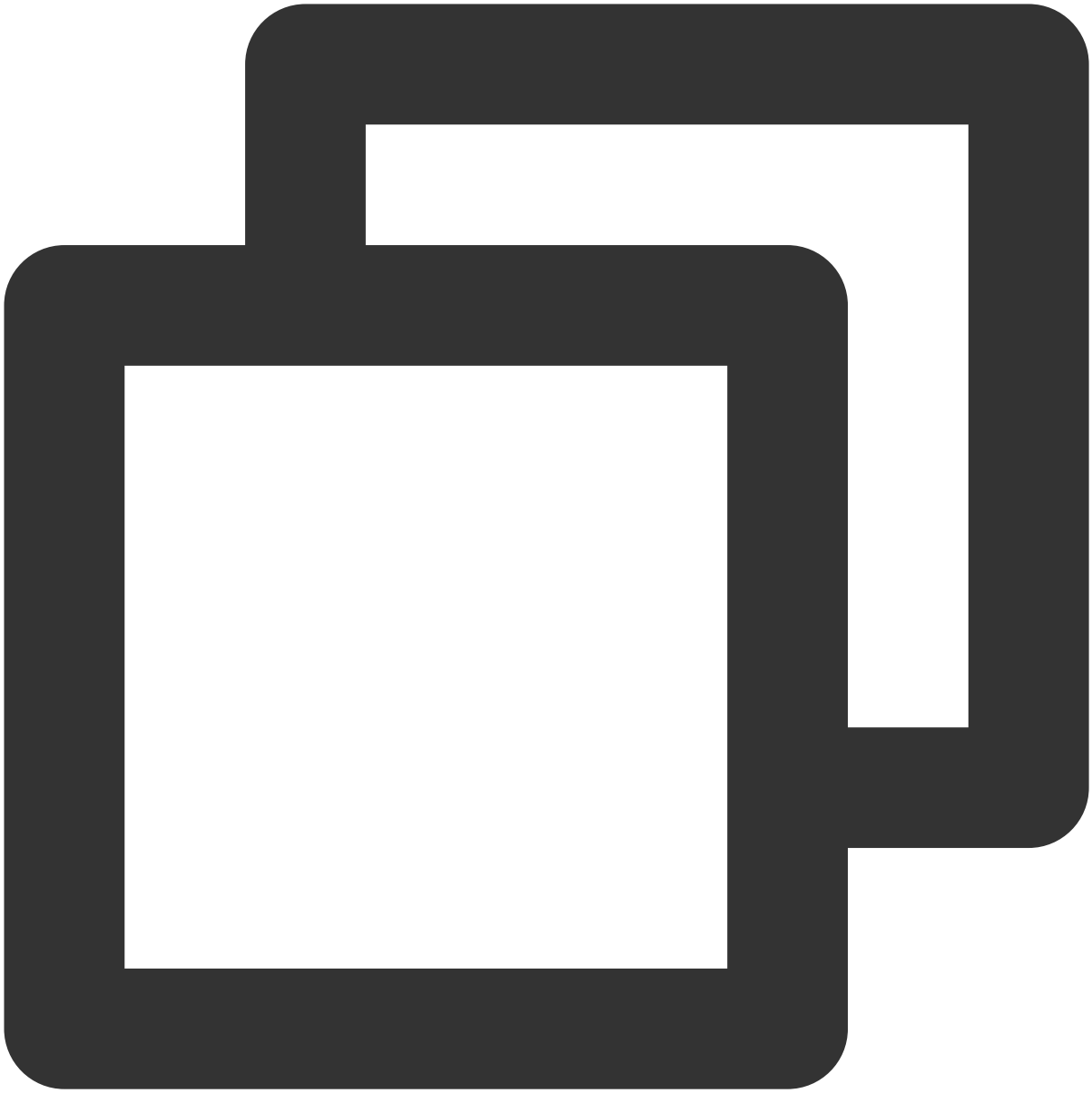
参数	含义
Key	对象文件名，例如 folder/sample.jpg。

获取图片主色调

功能说明

对象存储通过数据万象 **imageAve** 接口获取图片主色调信息。处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

代码示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->ImageAve(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'exampleobject',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：imageAve。

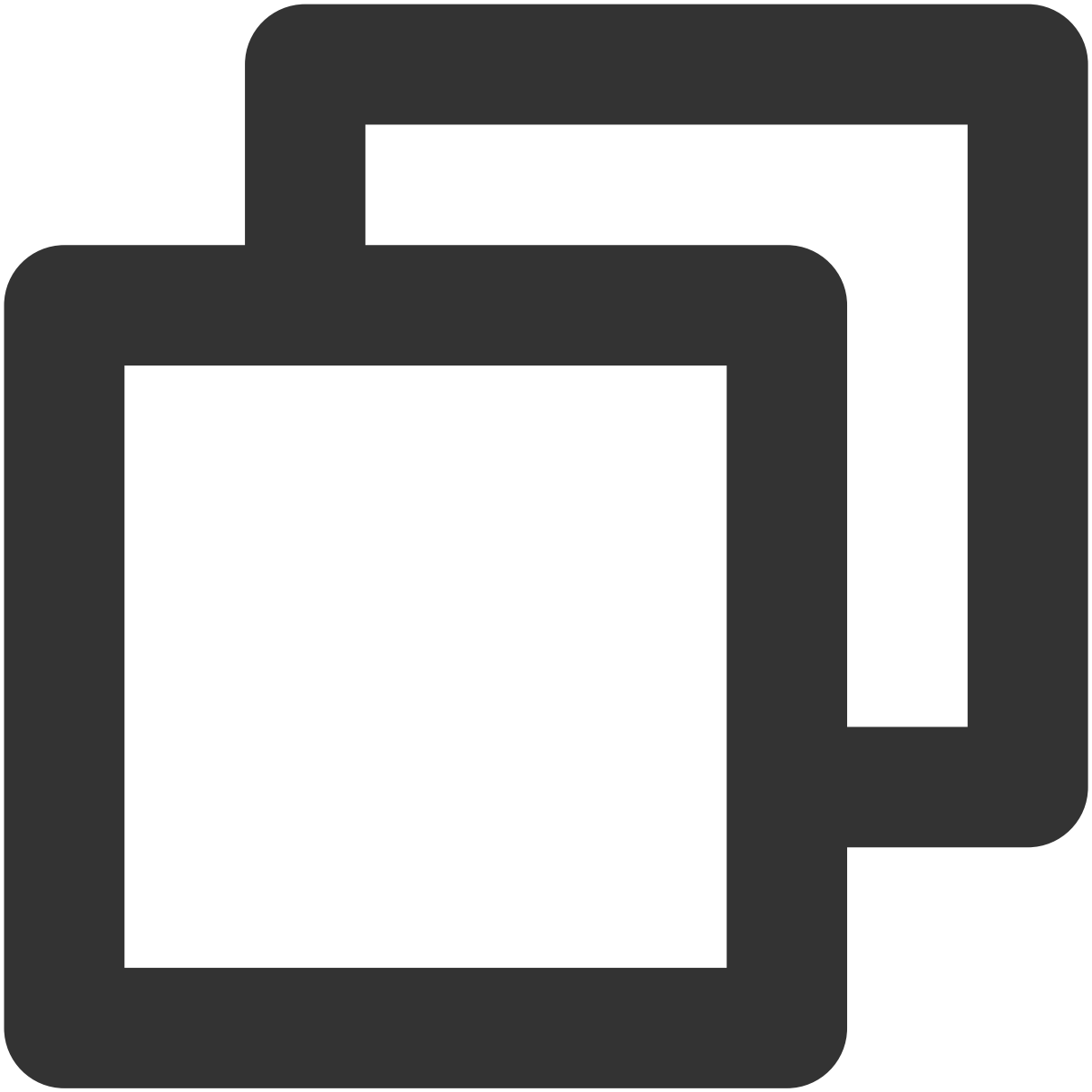
参数	含义
Key	对象文件名，例如 folder/sample.jpg。

去除元信息

功能说明

对象存储通过数据万象 **imageMogr2** 接口可去除图片元信息，包括 exif 信息。

代码示例



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-去除元信息 https://cloud.tencent.com/document/product/460/36547
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->strip(); // 去除图片元信息，包括 exif 信息  /strip
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：strip。

参数	含义
/ignore-error/1	当处理参数中携带此参数时，针对文件过大导致处理失败的场景，会直接返回原图而不报错。

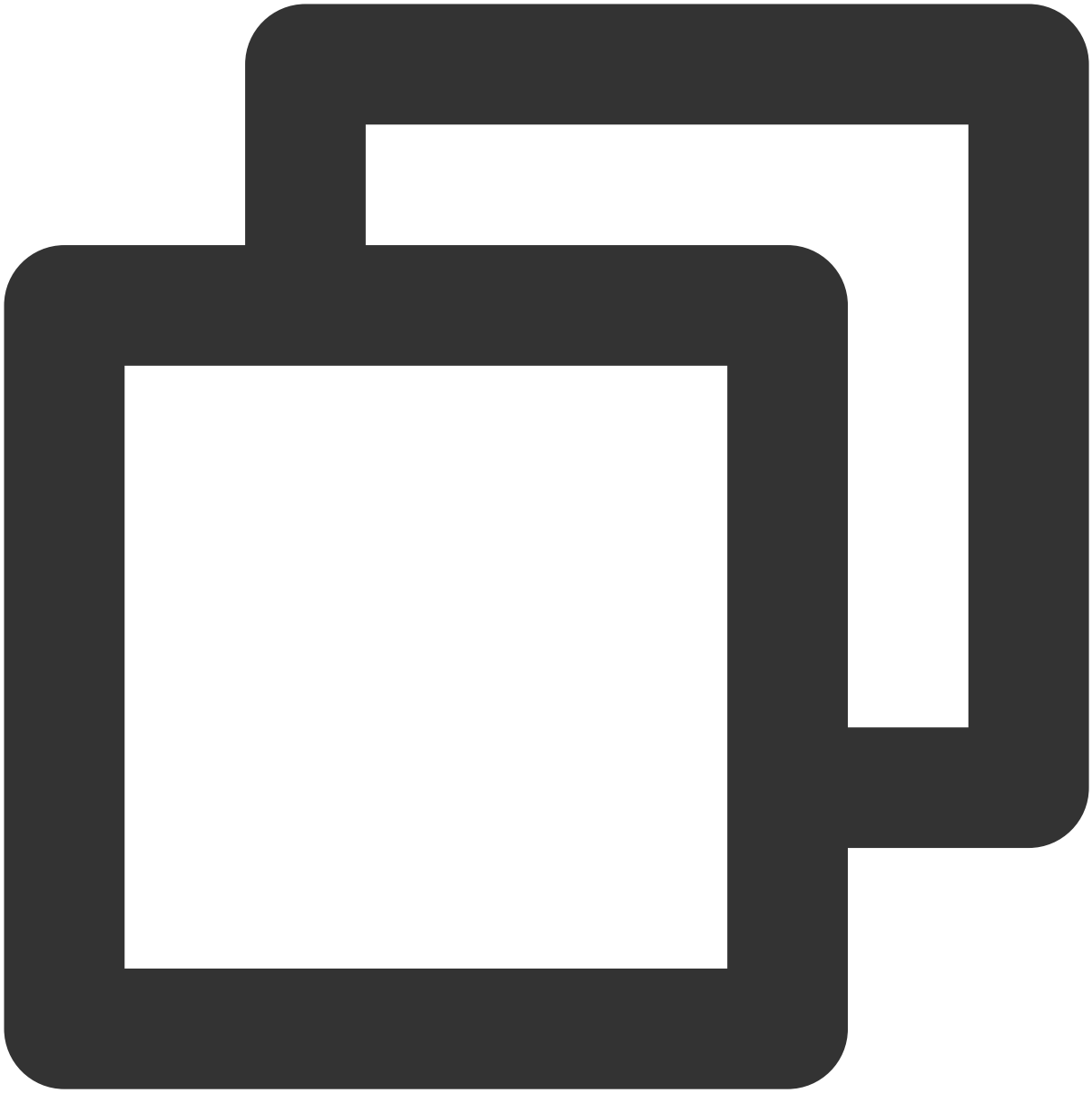
快速缩略模板

功能说明

对象存储通过数据万象 **imageView2** 接口提供常用图片处理模板。开发者根据业务需求，只需在下载 URL 后面附加相应的参数，就可以生成相应的缩略图。处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素，动图 gif 帧数限300帧。

代码示例

版权所有：腾讯云计算（北京）有限责任公司



```
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-快速缩略模板 https://cloud.tencent.com/document/product/460/6929
    $imageViewRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageViewTemplate();
    $imageViewRule->setMode(1); // /<mode>
    $imageViewRule->setWidth(400); // /w/<Width>
    $imageViewRule->setHeight(600); // /h/<Height>
    $imageViewRule->setFormat('jpg'); // /format/<Format>
    $imageViewRule->setQuality(1, 85); // 1-/q/<Quality>; 2-/rq/<quality>; 3-/lq/<q
    $imageViewRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageViewRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageViewRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
```

```
    // 请求失败
    echo($e);
}
```

处理参数说明

参数	含义
/1/w/<Width>/h/<Height>	限定缩略图的宽高最小值。该操作会将图像等比缩放直至某一边达到设定最小值，之后将另一边居中裁剪至设定值。若只指定一边，则表示宽高相等的正方形 例如，原图大小为1000x500，将参数设定为?imageView2/1/w/500/h/400 后，图像会先等比缩放至800x400，之后左右各裁剪150，得到500x400大小的图像
/2/w/<Width>/h/<Height>	限定缩略图的宽高最大值。该操作会将图像等比缩放至宽高都小于设定最大值 例如，原图大小为 1000x500，将参数设定为?imageView2/2/w/500/h/400后，图像会等比缩放至500x250。如果只指定一边，则另一边自适应
/3/w/<Width>/h/<Height>	限定缩略图的宽高最小值。该操作会将图像等比缩放至宽高都大于设定最小值 例如，原图大小为 1000x500，将参数设定为?imageView2/3/w/500/h/400后，图像会等比缩放至800x400。如果只指定一边，则另一边设为相同值
/4/w/<LongEdge>/h/<ShortEdge>	限定缩略图的长边和短边的最小值分别为 LongEdge 和 ShortEdge，进行等比压缩；如果只指定一边，代表另外一边为同样的值
/5/w/<LongEdge>/h/<ShortEdge>	限定缩略图的长边和短边的最大值分别为 LongEdge 和 ShortEdge，进行等比压缩，居中裁剪；如果只指定一边，则表示宽高相等的正方形；缩放后其中一边多余的部分会被裁剪掉
/format/<Format>	目标缩略图的图片格式，Format 可为：jpg，bmp，gif，png，webp，缺省为原图格式
/q/<Quality>	图片质量，取值范围0 - 100，默认值为原图质量；取原图质量和指定质量的最小值；<Quality> 后面加!（注意为英文字符），表示强制使用指定值
/rq/<quality>	图片的相对质量，取值范围0 - 100，数值以原图质量为标准。例如原图质量为80，将 rquality 设置为80后，得到处理结果图的图片质量为64（80x80%）
/lq/<quality>	图片的最低质量，取值范围0 - 100，设置结果图的质量参数最小值。 例如，原图质量为85，将 lquality 设置为80后，处理结果图的图片质量为85 例如，原图质量为60，将 lquality 设置为80后，处理结果图的图片质量会被提升至80

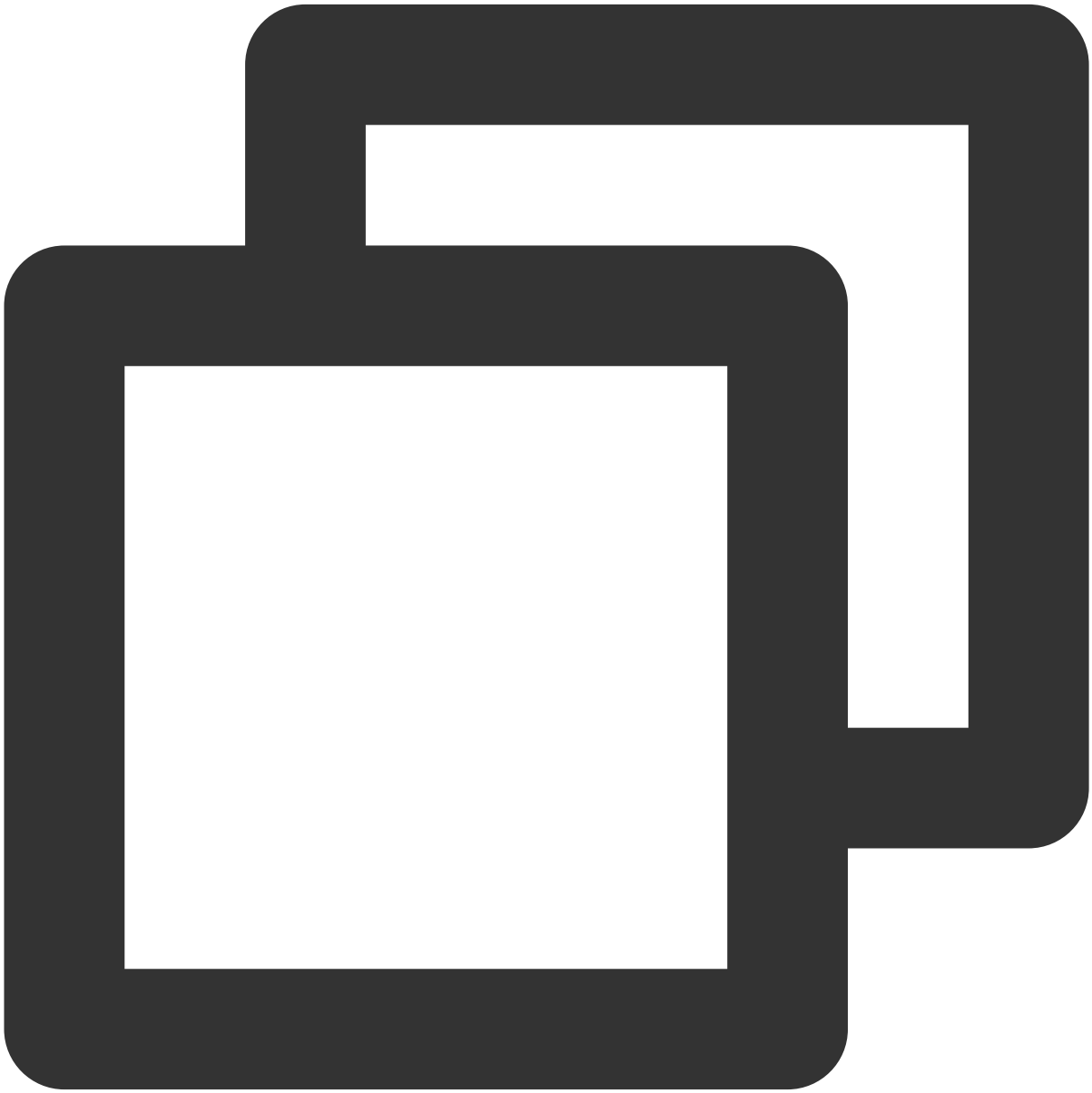
版权所有：腾讯云计算（北京）有限责任公司

限制图片大小

功能说明

对象存储通过数据万象 **imageMogr2/size-limit** 接口可限制图片处理（例如缩放、压缩等）后的文件大小。

代码示例



```
<?php
```

```
require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，htt
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-限制图片大小 https://cloud.tencent.com/document/product/460/56732
    $imageRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageRule->sizeLimit(15, 1); // 可限制图片处理（例如缩放、压缩等）后的文件大小 /size-l
    $imageRule->ignoreError(); // /ignore-error/1

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($imageRule, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $imageRule->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
```

版权所有：腾讯云计算（北京）有限责任公司

```
    // 请求成功
    print_r($result);
    // ---------------------- 3. 云上数据处理 ---------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

操作名称：size-limit。

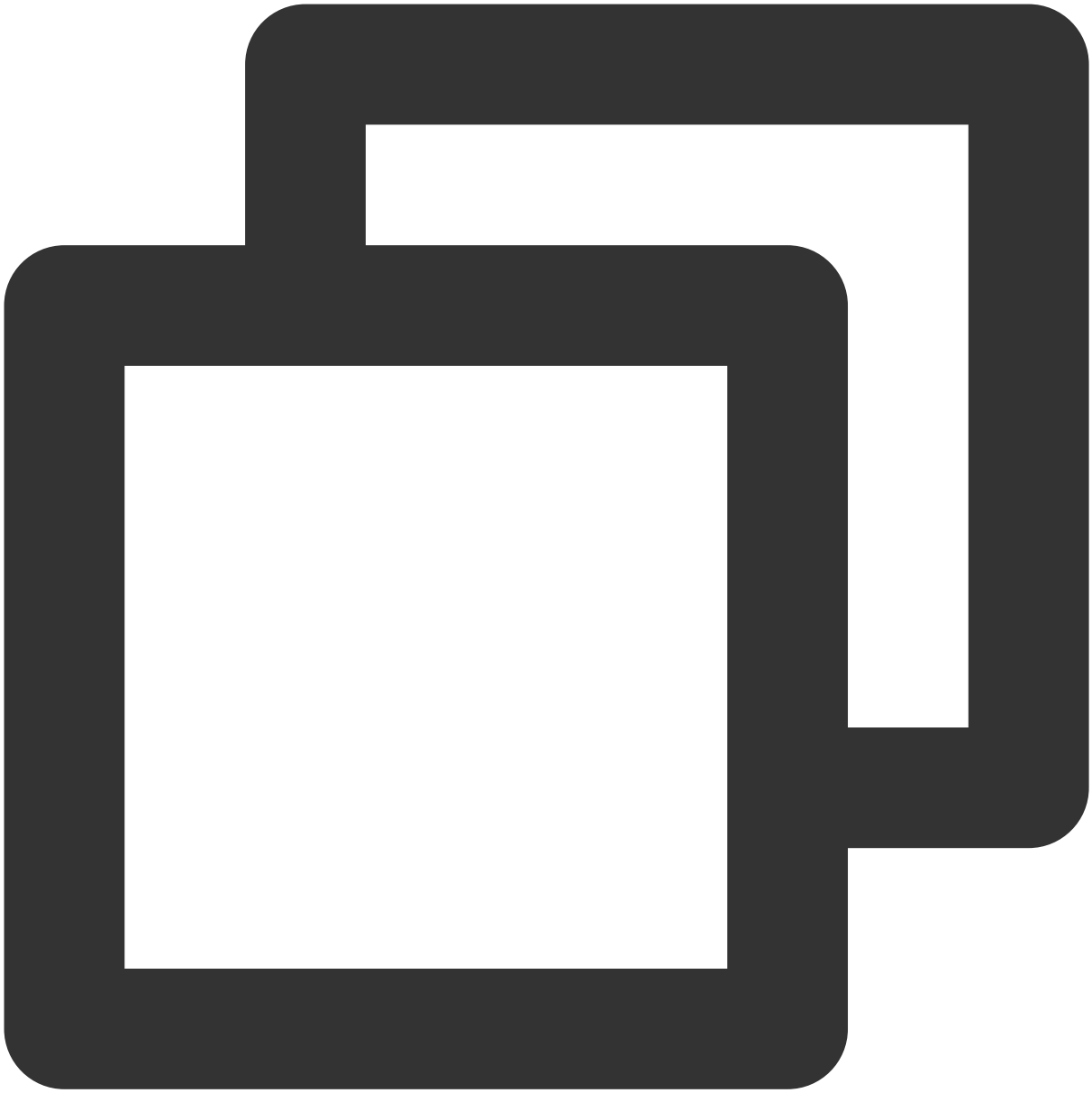
参数	含义
size-limit	限制图片转换后的大小，支持以兆字节（m）和千字节（k）为单位 1. 仅支持 JPG 格式的图片，可以用于限制处理后图片的大小 2. 若在尾部加上 `!`，表示用处理后的图片大小与原图大小做比较，如果处理后的图片比原图小，则返回处理后的图片，否则返回原图。例如：examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/picture.jpg?imageMogr2/size-limit/15k! 3. 建议搭配 `strip` 参数使用，去除图片的一些冗余信息，会有更好的效果。例如：examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/picture.jpg?imageMogr2/strip/format/png/size-limit/15k!
/ignore-error/1	当处理参数中携带此参数时，针对文件过大导致处理失败的场景，会直接返回原图而不报错

管道操作符

功能说明

对象存储通过数据万象的管道操作符 `|` 实现对图片按顺序进行多种处理。用户可以通过管道操作符将多个处理参数分隔开，从而实现在一次访问中按顺序对图片进行不同处理。处理图片原图大小不超过32MB、宽高不超过30000像素且总像素不超过2.5亿像素，处理结果图宽高设置不超过9999像素；针对动图，原图宽 x 高 x 帧数不超过2.5亿像素。

代码示例



```php
<?php


require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
```

```
        'schema' => 'https', //协议头部，默认为 http
        'credentials' => array(
            'secretId' => $secretId,
            'secretKey' => $secretKey)));
try {
    // 图片处理-管道操作符 https://cloud.tencent.com/document/product/460/15293
    $imageMogrRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate();
    $imageMogrRule->thumbnailByScale(50);
    $imageMogrRule->rotate(50);
    $imageViewRule = new Qcloud\\Cos\\ImageParamTemplate\\ImageViewTemplate();
    $imageViewRule->setMode(1);
    $imageViewRule->setWidth(400);
    $imageViewRule->setHeight(600);
    $imageViewRule->setQuality(1, 85);
    $ciParamChannel = new Qcloud\\Cos\\ImageParamTemplate\\CIParamTransformation();
    $ciParamChannel->addRule($imageMogrRule); // 多种处理方式 以 “|” 连接
    $ciParamChannel->addRule($imageViewRule); // 多种处理方式 以 “|” 连接

    $picOperations = new Qcloud\\Cos\\ImageParamTemplate\\PicOperationsTransformati
    $picOperations->setIsPicInfo(1); // is_pic_info
    $picOperations->addRule($ciParamChannel, "output.png"); // rules

    // -------------------- 1. 下载时处理 -------------------- //
//    $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'
    $downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/
    $rule = $ciParamChannel->queryString();
    echo "{$downloadUrl}?{$rule}";
//    echo "{$downloadUrl}&{$rule}"; // 携带签名的图片地址以“&”连接
    // -------------------- 1. 下载时处理 -------------------- //

    // -------------------- 2. 上传时处理 -------------------- //
    $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'object.jpg',
        'Body' => fopen('/tmp/local.jpg', 'rb'), // 本地文件
        'PicOperations' => $picOperations->queryString(),
    ));
    // 请求成功
    print_r($result);
    // -------------------- 2. 上传时处理 -------------------- //

    // -------------------- 3. 云上数据处理 -------------------- //
    $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'test.png',
        'PicOperations' => $picOperations->queryString(),
    ));
```

```
    // 请求成功
    print_r($result);
    // -------------------- 3. 云上数据处理 -------------------- //
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

处理参数说明

多个处理样式之间以管道操作符 `|` 分隔，样式按照先后顺序执行，目前最多支持10层管道。

图片样式

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于图片样式相关的 API 概览以及 SDK 示例代码。

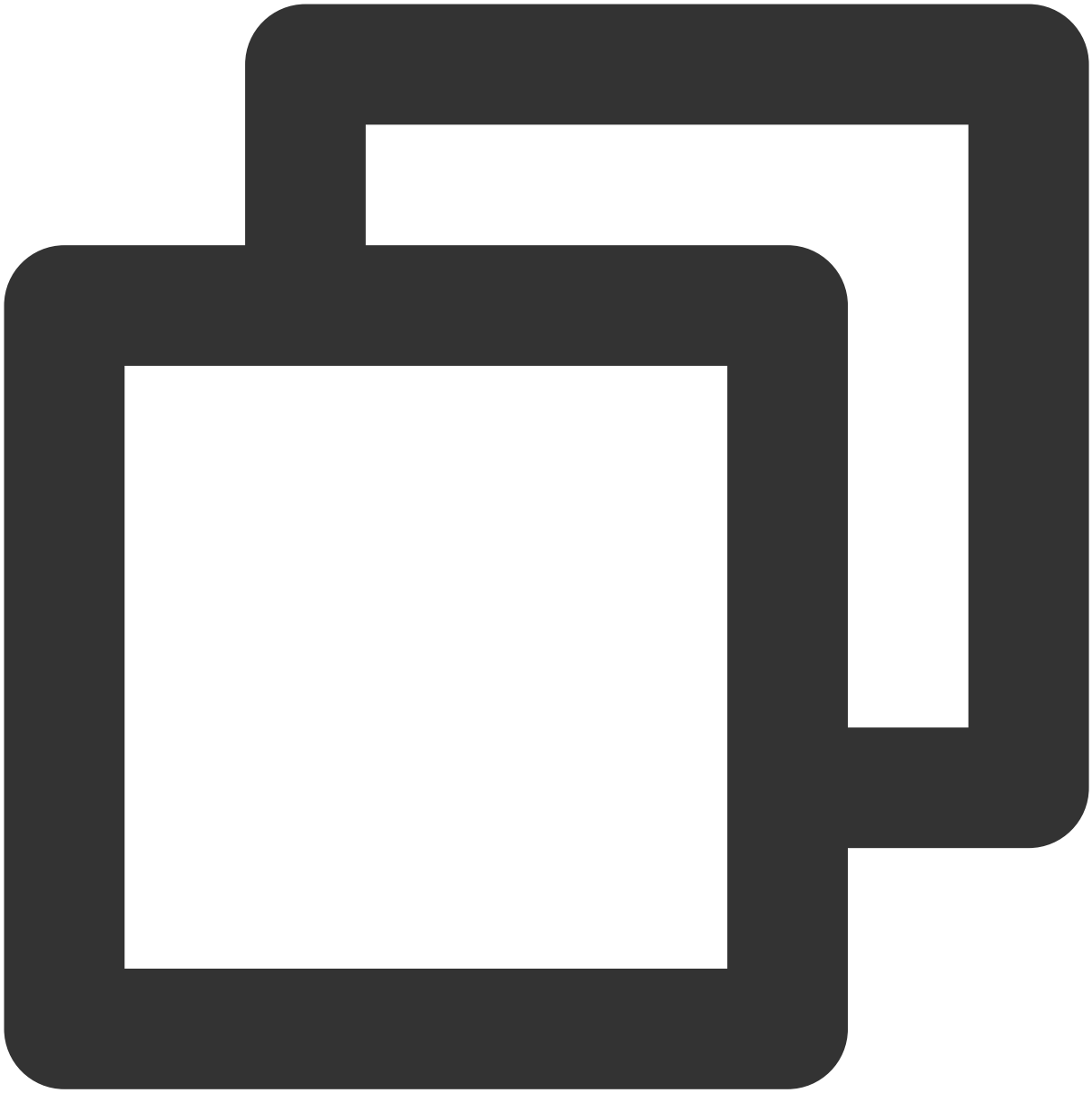
API	说明
增加样式	增加存储桶样式
查询样式	查询存储桶样式
删除样式	删除存储桶样式

增加样式

功能说明

对某一个存储桶设置样式功能，后续上传到该存储桶的图片文件都会被添加指定的样式。

示例代码



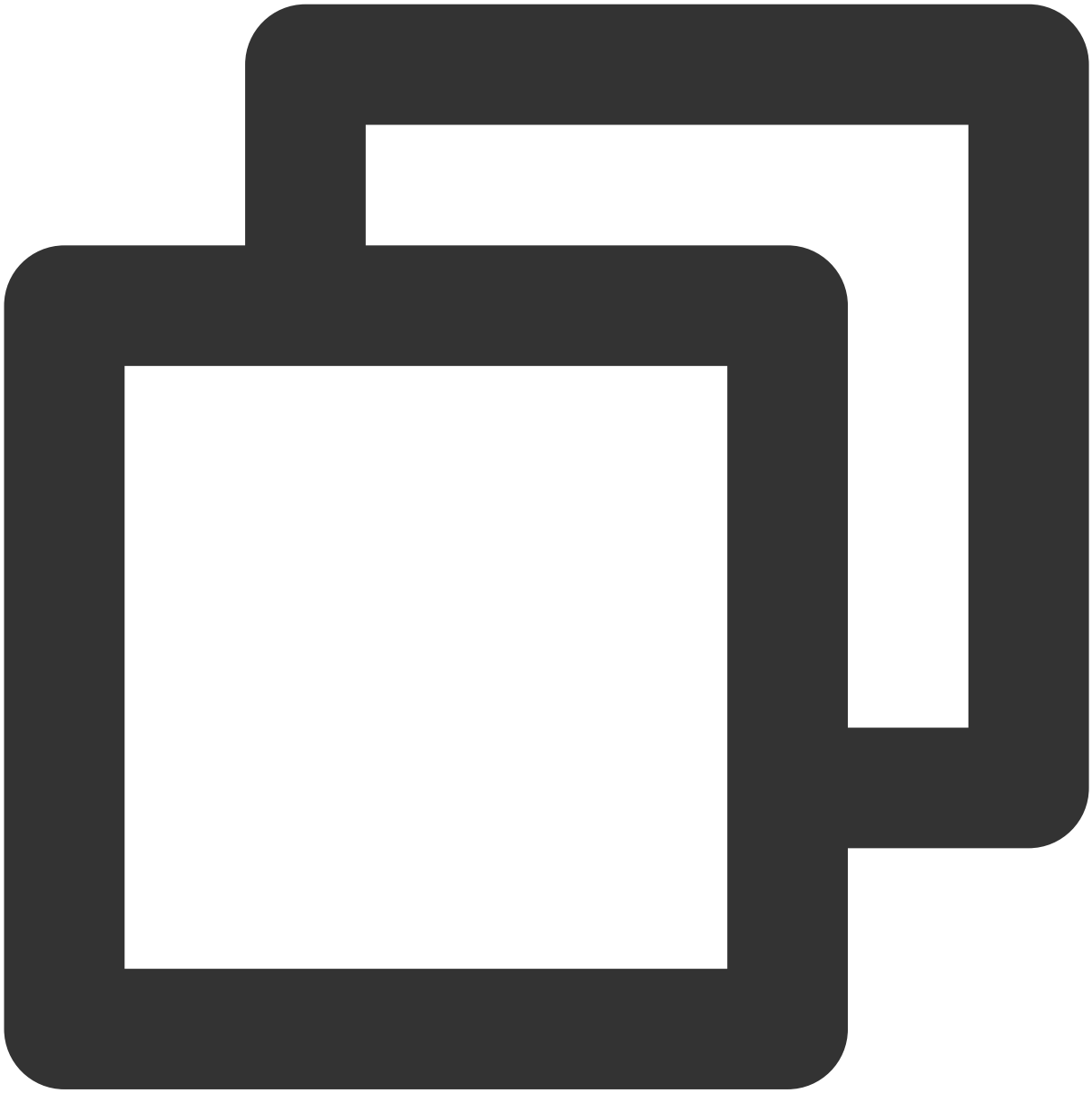
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->PutBucketImageStyle(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'StyleName' => 'style_name',//样式名称
        'StyleBody' => 'imageMogr2/thumbnail/!50px', //样式信息
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
StyleName	String	样式名称	是
StyleBody	String	样式信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
        )
)
```

返回结果说明

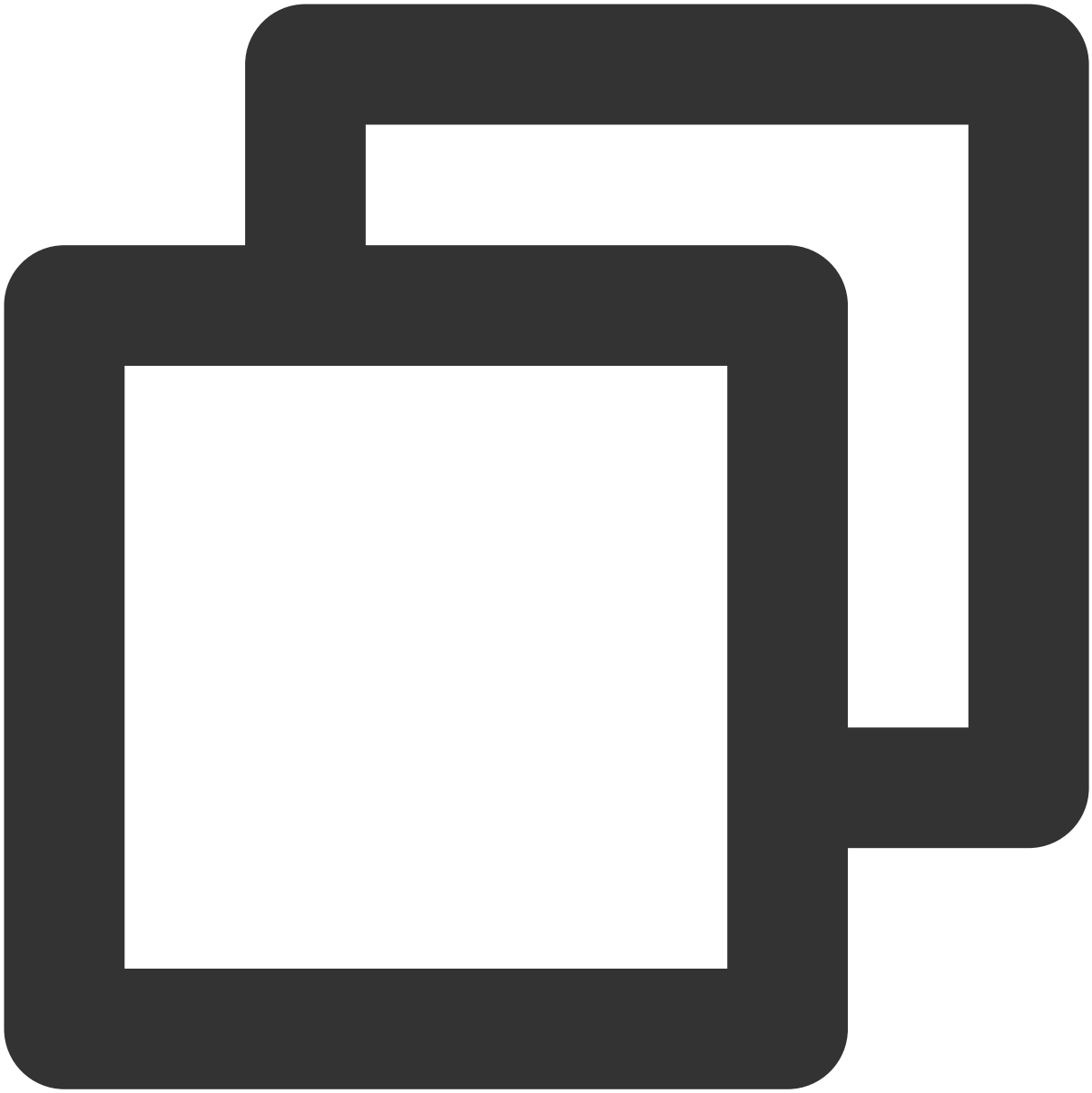
参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

查询样式

功能说明

查询某个存储桶下已有的样式。

示例代码



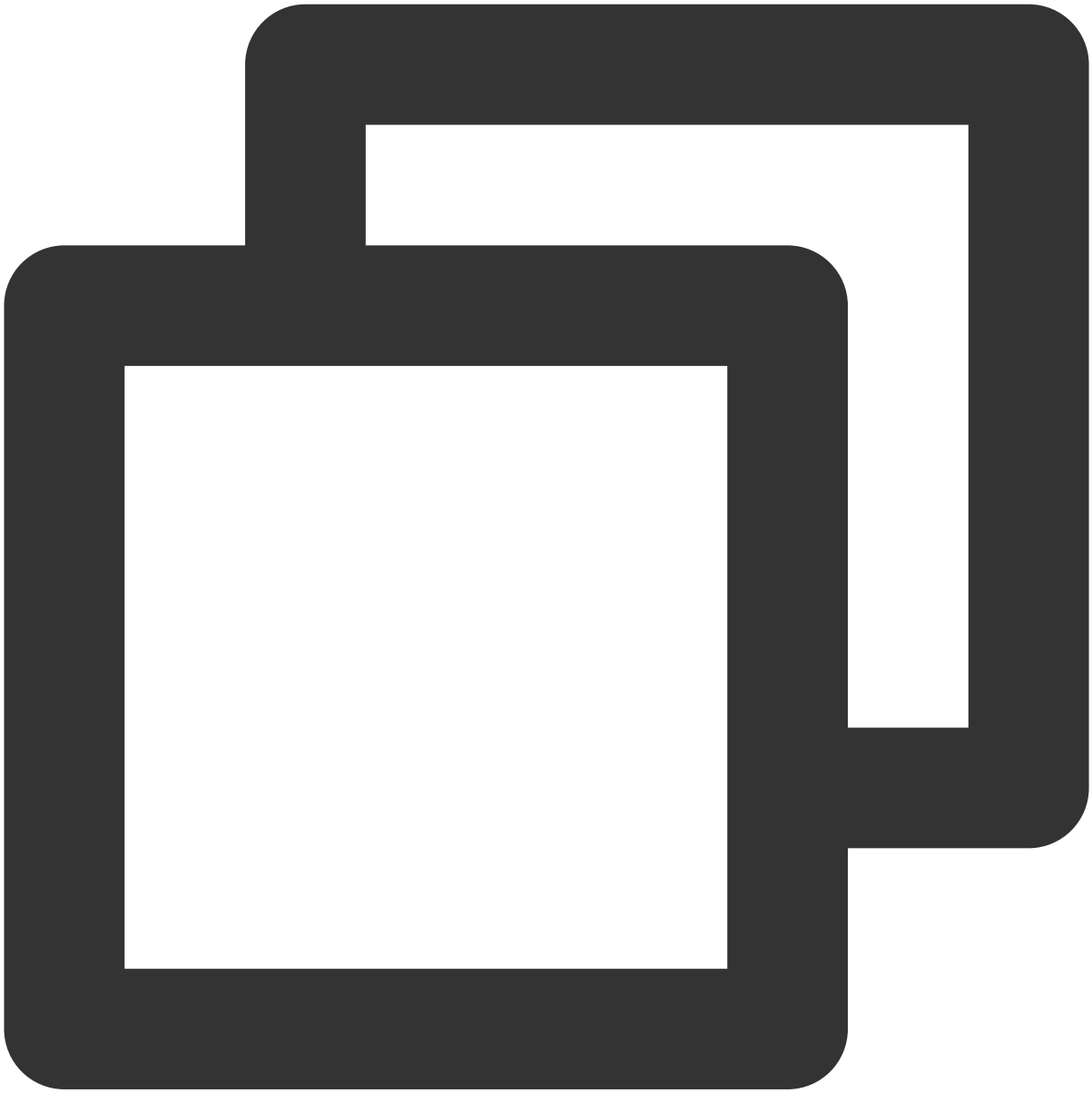
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->GetBucketImageStyle(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'StyleName' => 'style_name', //样式名称
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
StyleName	String	样式名称	否

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
            [StyleRule] => Array(
                [0] => Array(
                    [StyleName] => style_name
```

```
                    [StyleBody] => imageMogr2/thumbnail/!50px
                )
            )
        )
)
```

返回结果说明

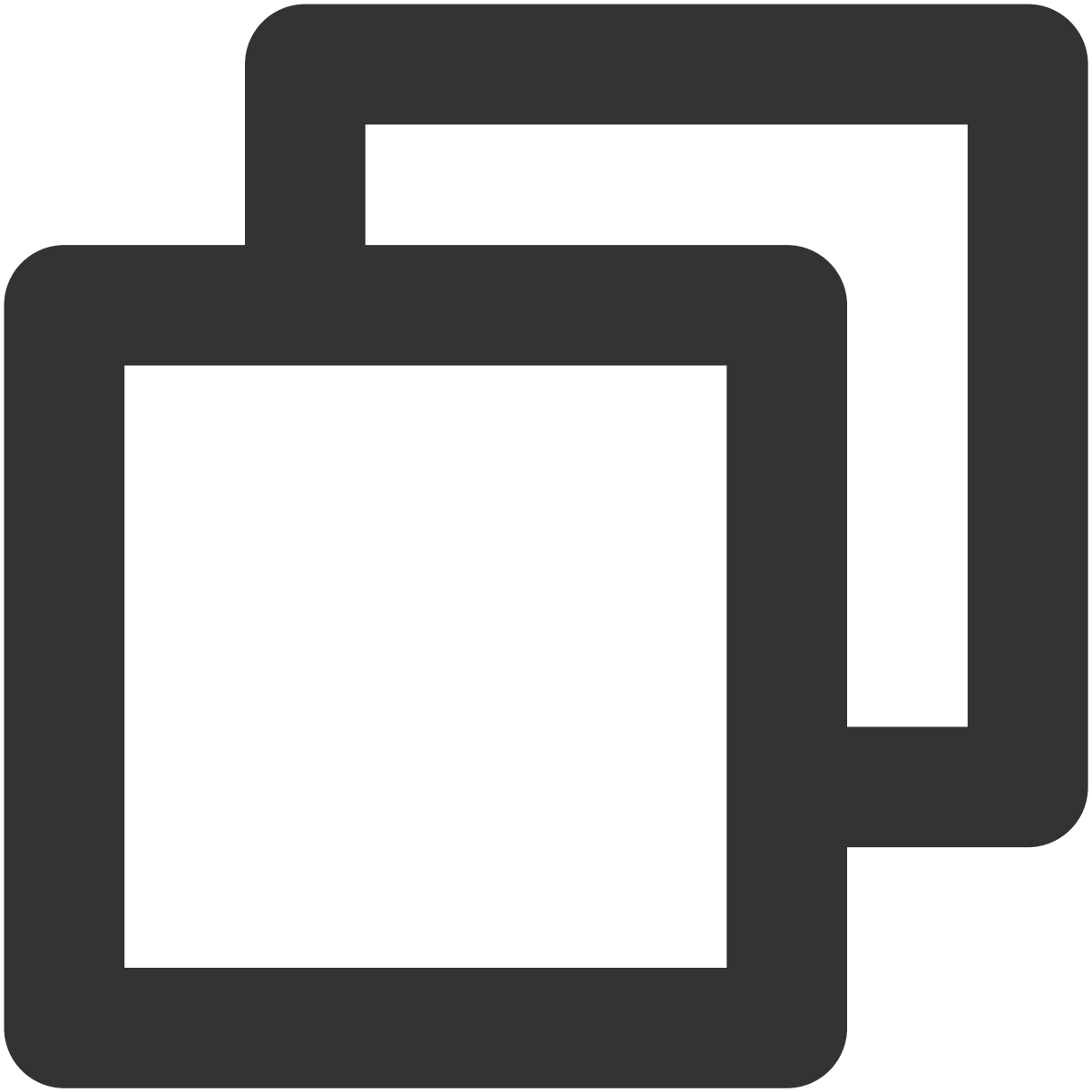
参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
StyleRule	Array	样式信息列表	无

删除样式

功能说明

删除某一特定样式。

示例代码



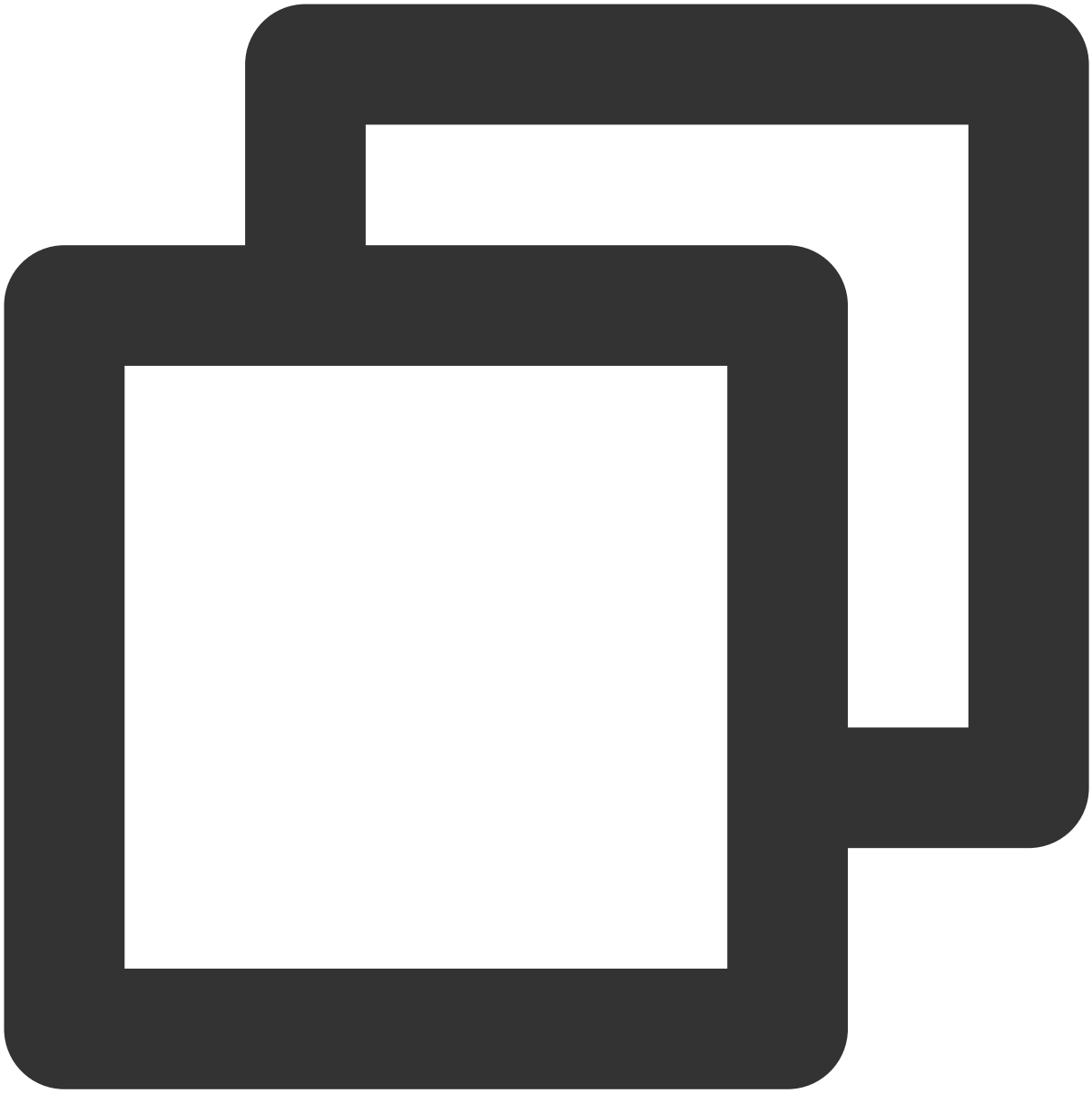
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->DeleteBucketImageStyle(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'StyleName' => 'style_name', //样式名称
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
StyleName	String	样式名称	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
        )
)
```

返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

图片持久化处理

最近更新时间：2024-03-13 10:25:35

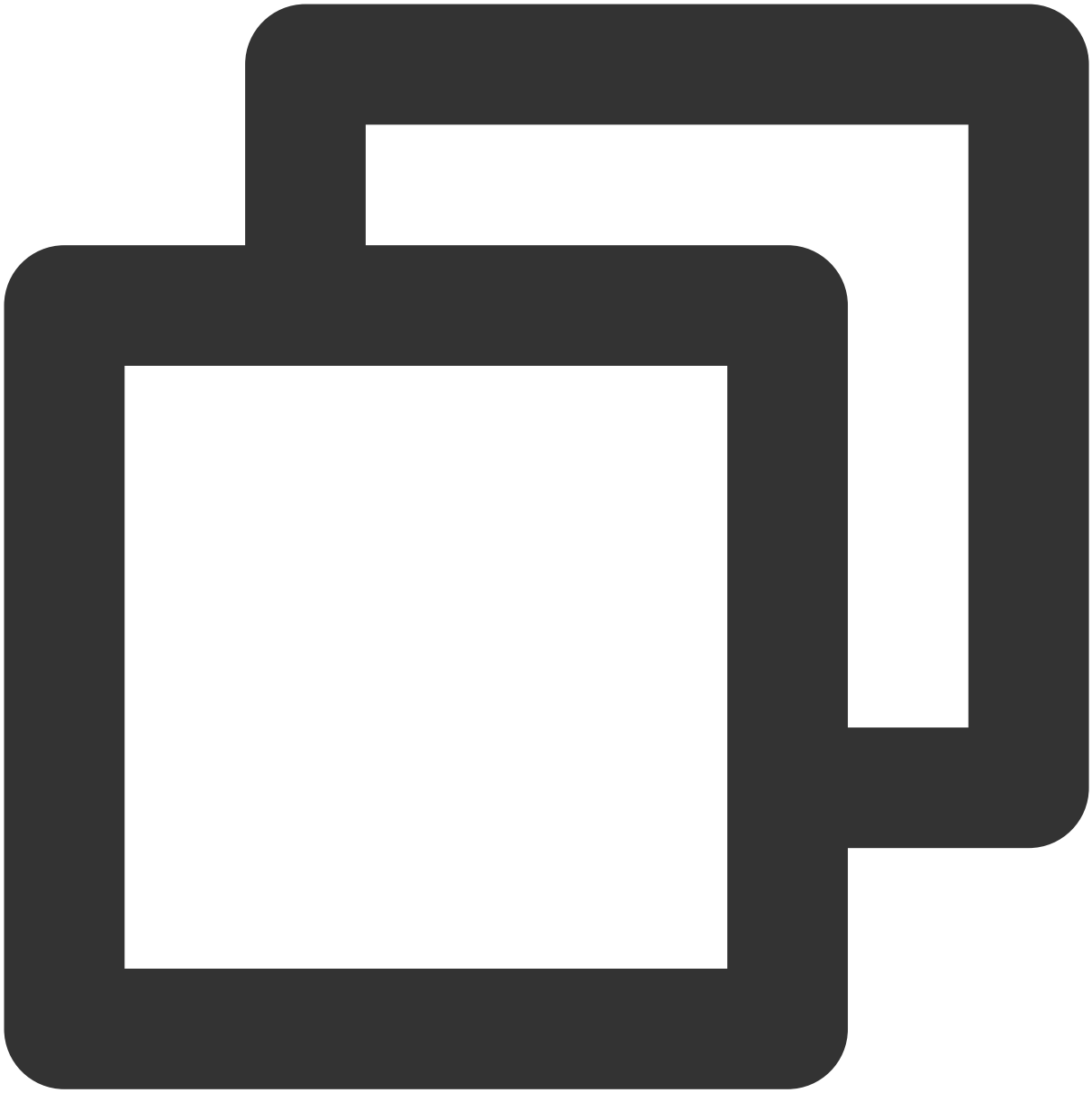
简介

本文档提供关于图片持久化处理的 API 概览以及 SDK 示例代码。

API	说明
图片持久化处理	对象存储（Cloud Object Storage，COS）提供的上传时处理功能可以帮助使用者在上传时实现图片处理。此外能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS

上传时处理

示例代码



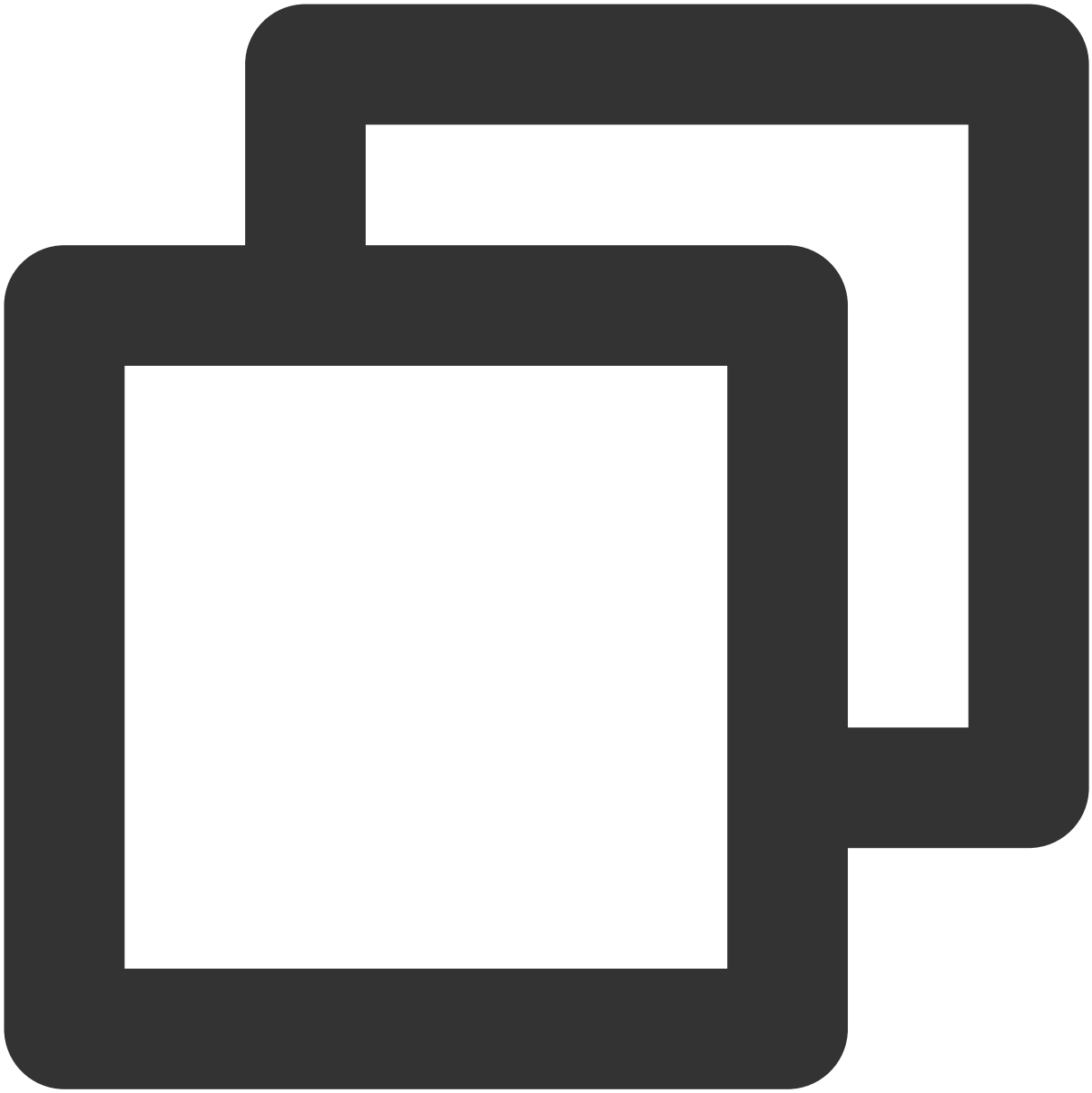
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $imageMogrTemplate = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate
        $imageMogrTemplate->thumbnailByScale(50);//指定图片的宽高为原图的 50%
        $picOperationsTemplate = new \\Qcloud\\Cos\\ImageParamTemplate\\PicOperatio
        $picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，
        $picOperationsTemplate->addRule($imageMogrTemplate, "resultobject");//设置图
        $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'Body' => fopen('path/to/localFile', 'rb'),
        'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
Body	File/String	上传的内容	是
PicOperations	Json/String	图片持久化处理信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [Body] =>
            [ETag] => "698d51a19d8a121ce581499d7b701668"
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 238186
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
```

```
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [Data] => Array
            (
                [OriginalInfo] => Array
                (
                    [Key] => exampleobject
                    [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.
                    [ETag] => "7037fb6fb4cca43b958a28789605e73d98088720"
                    [ImageInfo] => Array
                    (
                            [Format] => JPEG
                            [Width] => 600
                            [Height] => 500
                            [Quality] => 90
                            [Ave] => 0x46442e
                            [Orientation] => 0
                     )

                )
                [ProcessResults] => Array
                (
                    [Object] => Array
                    (
                        [0] => Array(
                            [Key] => resultobject
                            [Location] => examplebucket-1250000000.cos.ap-beijing.m
                            [Format] => JPEG
                            [Width] => 300
                            [Height] => 200
                            [Size] => 30000
                            [Quality] => 90
                            [ETag] => "87c153bc2909aa0ba111ca126b675c510d36b817"
                        )
                    )
                )
            )
    )
)
```

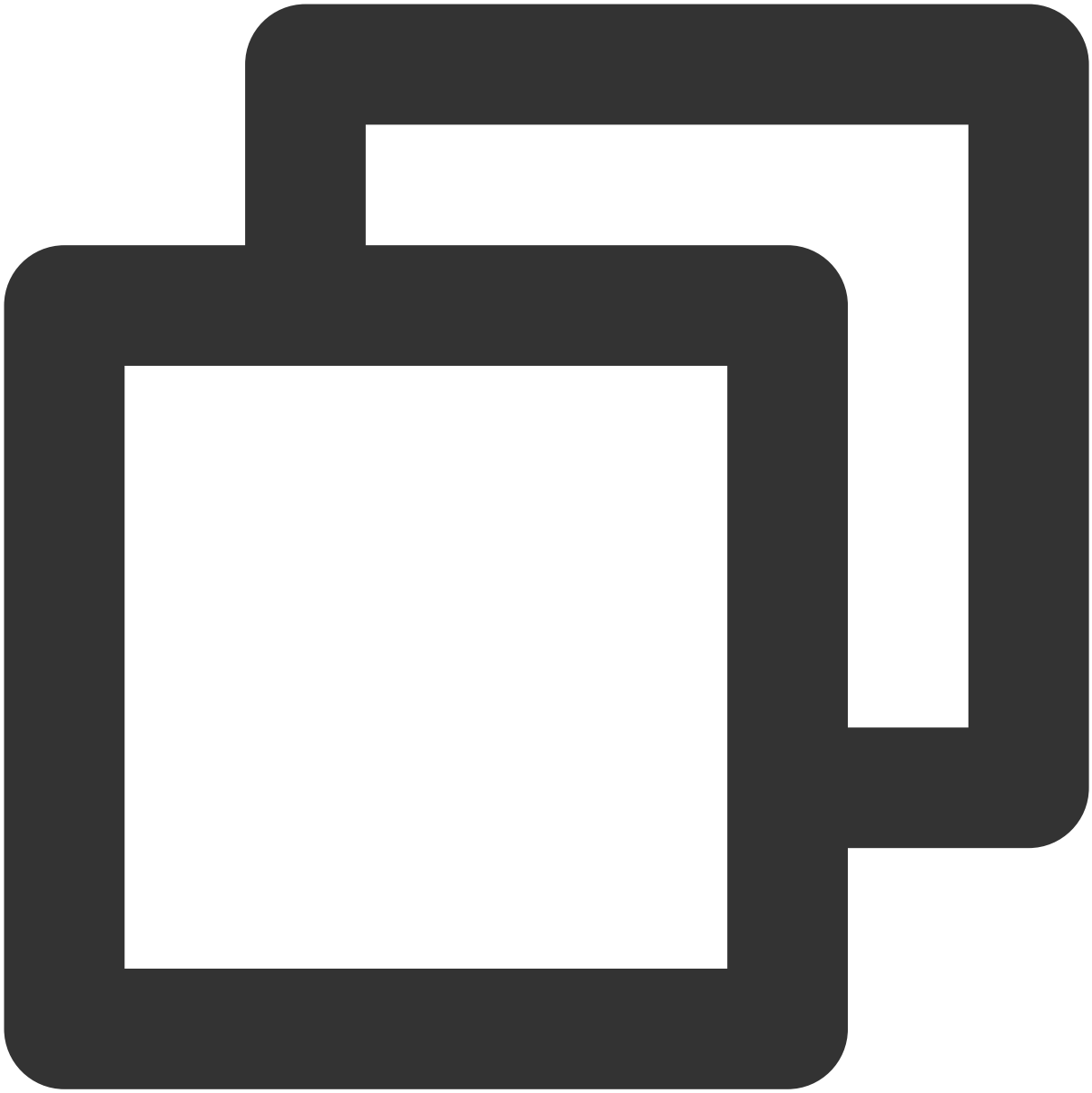
返回结果说明

参数名称	类型	描述	父节点
Body	File/String	返回体	无
ETag	String	文件的 MD5 值	无

RequestId	String	请求 ID 标识	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
Data	Array	图片处理结果信息	无

云上数据处理

示例代码



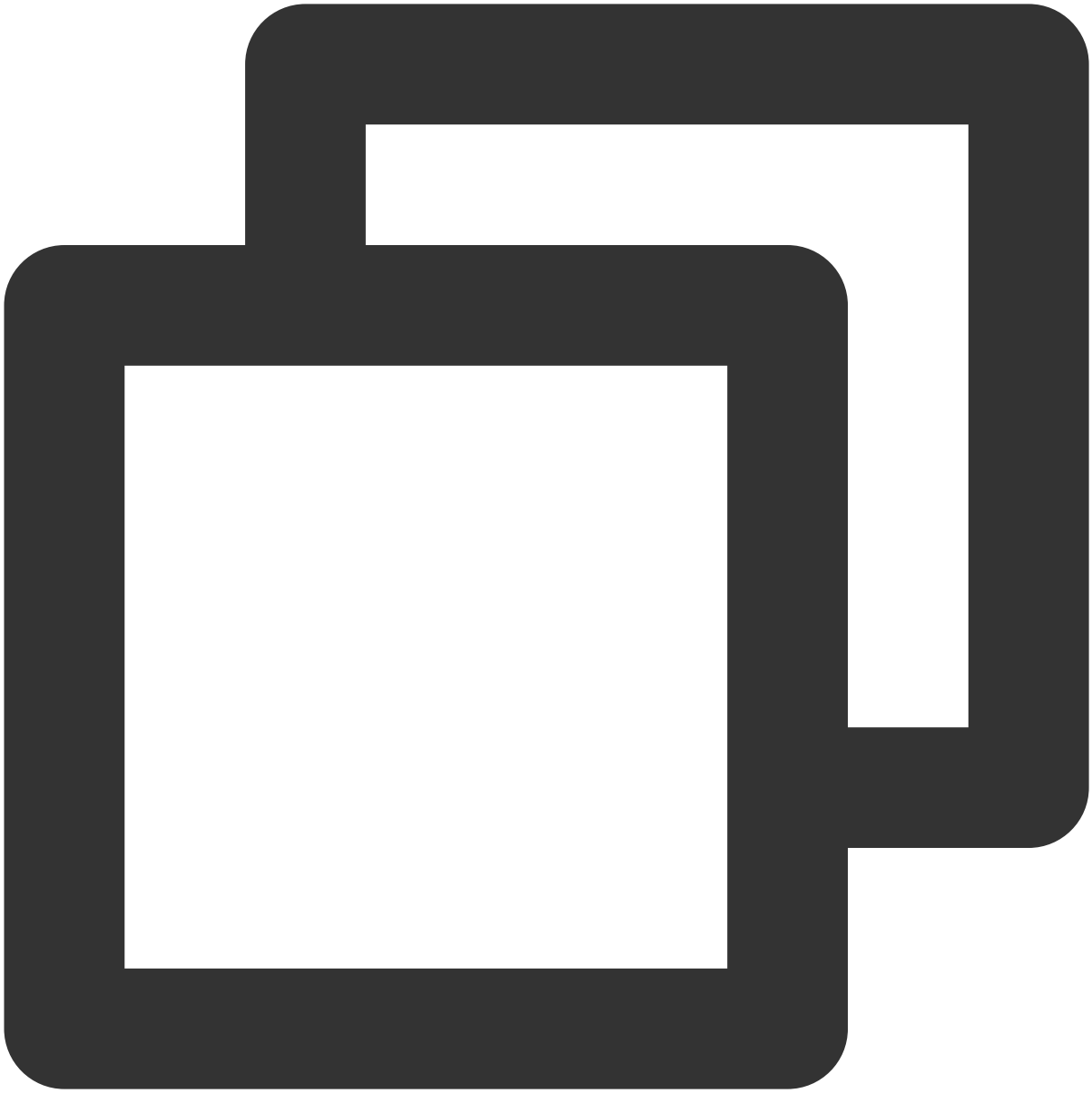
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $imageMogrTemplate = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate
        $imageMogrTemplate->thumbnailByScale(50);//指定图片的宽高为原图的 50%
        $picOperationsTemplate = new \\Qcloud\\Cos\\ImageParamTemplate\\PicOperatio
        $picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，
        $picOperationsTemplate->addRule($imageMogrTemplate, "resultobject");//设置图
        $result = $cosClient->ImageProcess(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
PicOperations	Json/String	图片持久化处理信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [OriginalInfo] => Array
            (
```

```
                [Key] => exampleobject
                [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/
                [ETag] => "7037fb6fb4cca43b958a28789605e73d98088720"
                [ImageInfo] => Array
                (
                        [Format] => JPEG
                        [Width] => 600
                        [Height] => 500
                        [Quality] => 90
                        [Ave] => 0x46442e
                        [Orientation] => 0
                    )

            )
            [ProcessResults] => Array
            (
                [Object] => Array
                (
                    [0] => Array(
                        [Key] => resultobject
                        [Location] => examplebucket-1250000000.cos.ap-beijing.myqcl
                        [Format] => JPEG
                        [Width] => 300
                        [Height] => 200
                        [Size] => 30000
                        [Quality] => 90
                        [ETag] => "87c153bc2909aa0ba111ca126b675c510d36b817"
                    )
                )
            )
    )
)
```

返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
OriginalInfo	Array	源图片信息	无

版权所有：腾讯云计算（北京）有限责任公司

ProcessResults	Array	图片处理结果信息	无

盲水印

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。

API	操作描述
盲水印	对本地图片添加或提取盲水印并上传至存储桶

添加盲水印

上传时添加

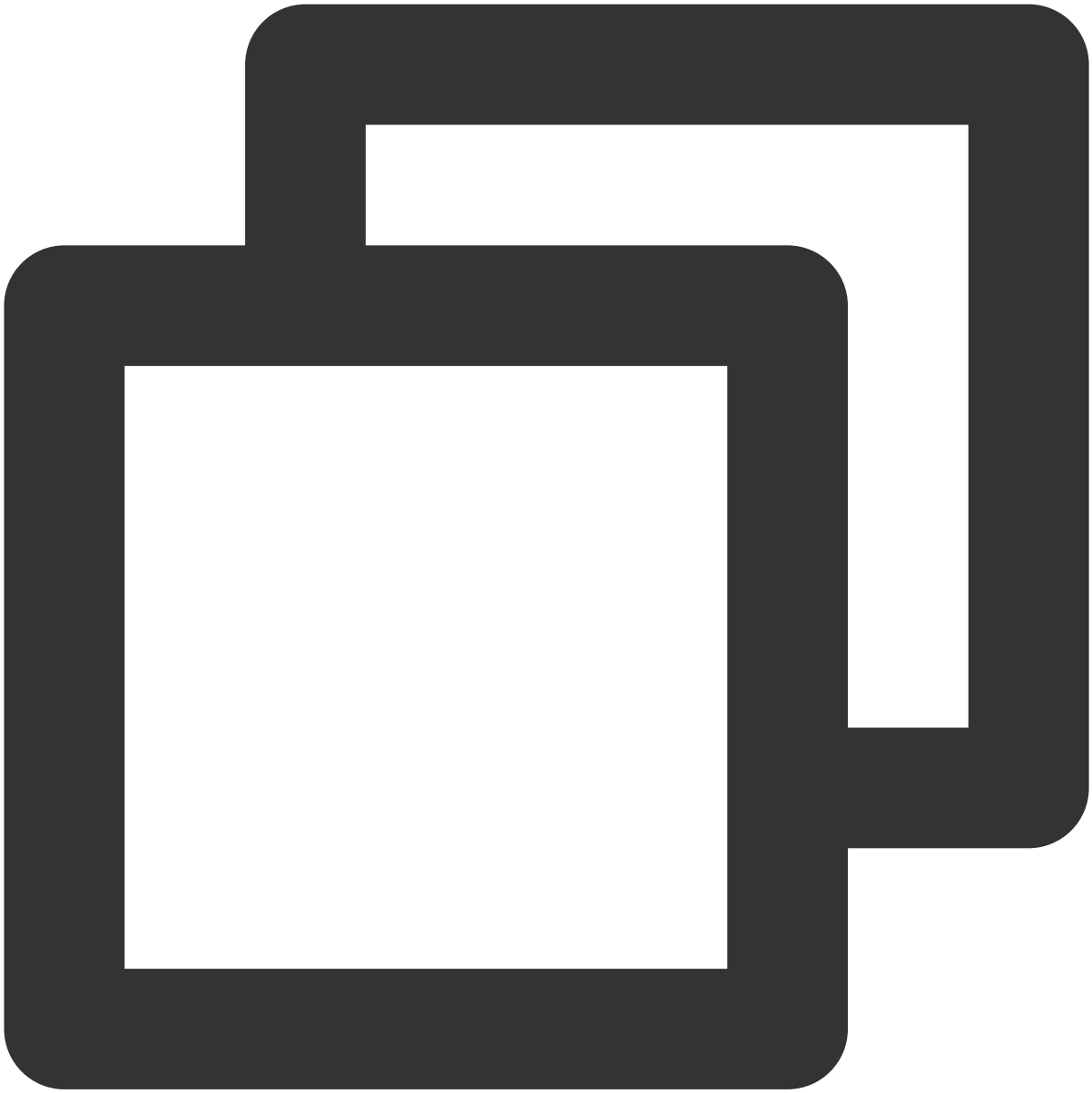
指定的水印图片必须同时满足如下3个条件：

1. 水印图片与源图片必须位于同一个存储桶下。

2. URL 需使用 COS 域名（不能使用 CDN 加速域名，例如 examplebucket-1250000000.file.myqcloud.com/shuiyin_2.png 不可用），且需保证水印图可访问。如果水印图读取权限为私有，则需要携带有效签名。

3. URL 必须以 `http://` 开始，不能省略 HTTP 头，也不能填 HTTPS 头，例如 `examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png`，`https://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` 为非法的水印 URL。

示例代码



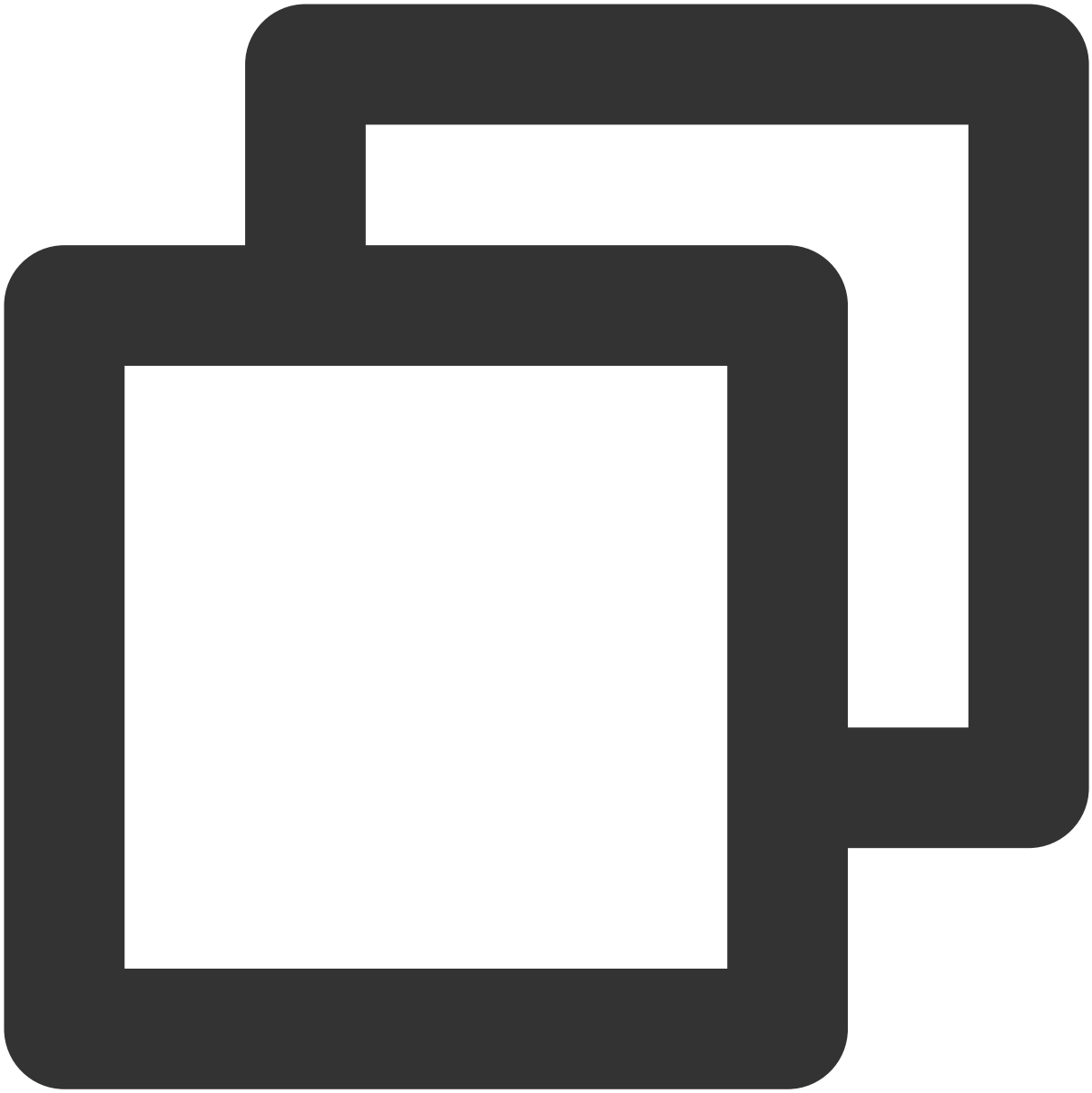
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $blindWatermarkTemplate = new Qcloud\\Cos\\ImageParamTemplate\\BlindWaterma
        $blindWatermarkTemplate->setType(3);//盲水印类型，有效值：1 半盲；2 全盲；3 文字
        $blindWatermarkTemplate->setImage("imageUrl");//设置盲水印图片地址
        $blindWatermarkTemplate->setText("Test");//设置盲水印文字
        $blindWatermarkTemplate->setLevel(3);//只对全盲水印（type=2）有效。level 的取值
        $picOperationsTemplate = new \\Qcloud\\Cos\\ImageParamTemplate\\PicOperatio
        $picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，
        $picOperationsTemplate->addRule($blindWatermarkTemplate, "resultobject");//
        $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'Body' => fopen('path/to/localFile', 'rb'),
        'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
Body	File/String	上传的内容	是
PicOperations	Json/String	图片持久化处理信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [Body] =>
            [ETag] => "698d51a19d8a121ce581499d7b701668"
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 10000
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
```

```
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [Data] => Array
            (
                [OriginalInfo] => Array
                (
                    [Key] => exampleobject
                    [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.
                    [ETag] => "7037fb6fb4cca43b958a28789605e73d98088720"
                    [ImageInfo] => Array
                    (
                            [Format] => JPEG
                            [Width] => 600
                            [Height] => 500
                            [Quality] => 90
                            [Ave] => 0x46442e
                            [Orientation] => 0
                     )

                )
                [ProcessResults] => Array
                (
                    [Object] => Array
                    (
                        [0] => Array(
                            [Key] => resultobject
                            [Location] => examplebucket-1250000000.cos.ap-beijing.m
                            [Format] => JPEG
                            [Width] => 300
                            [Height] => 200
                            [Size] => 30000
                            [Quality] => 90
                            [ETag] => "87c153bc2909aa0ba111ca126b675c510d36b817"
                        )
                    )
                )
            )
    )
)
```

返回结果说明

参数名称	类型	描述	父节点
Body	File/String	返回体	无
ETag	String	文件的 MD5 值	无

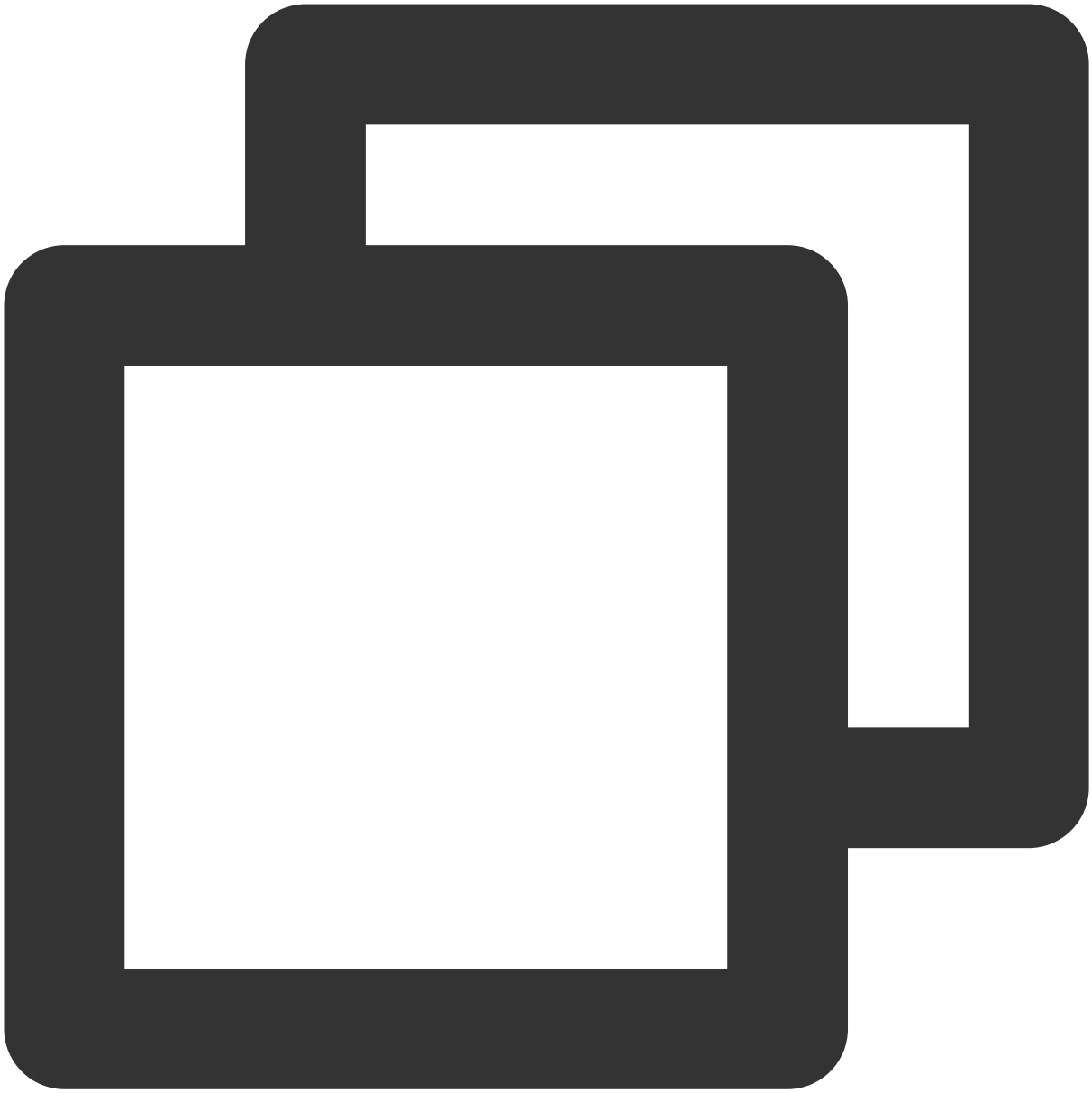
RequestId	String	请求 ID 标识	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
Data	Array	图片处理结果信息	无

下载时添加

指定的水印图片必须同时满足如下3个条件：

1. 水印图片与源图片必须位于同一个存储桶下。
2. URL 需使用 COS 域名（不能使用 CDN 加速域名，例如 examplebucket-1250000000.file.myqcloud.com/shuiyin_2.png 不可用），且需保证水印图可访问。如果水印图读取权限为私有，则需要携带有效签名。
3. URL 必须以 `http://` 开始，不能省略 HTTP 头，也不能填 HTTPS 头，例如 `examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png`，`https://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` 为非法的水印 URL。

示例代码



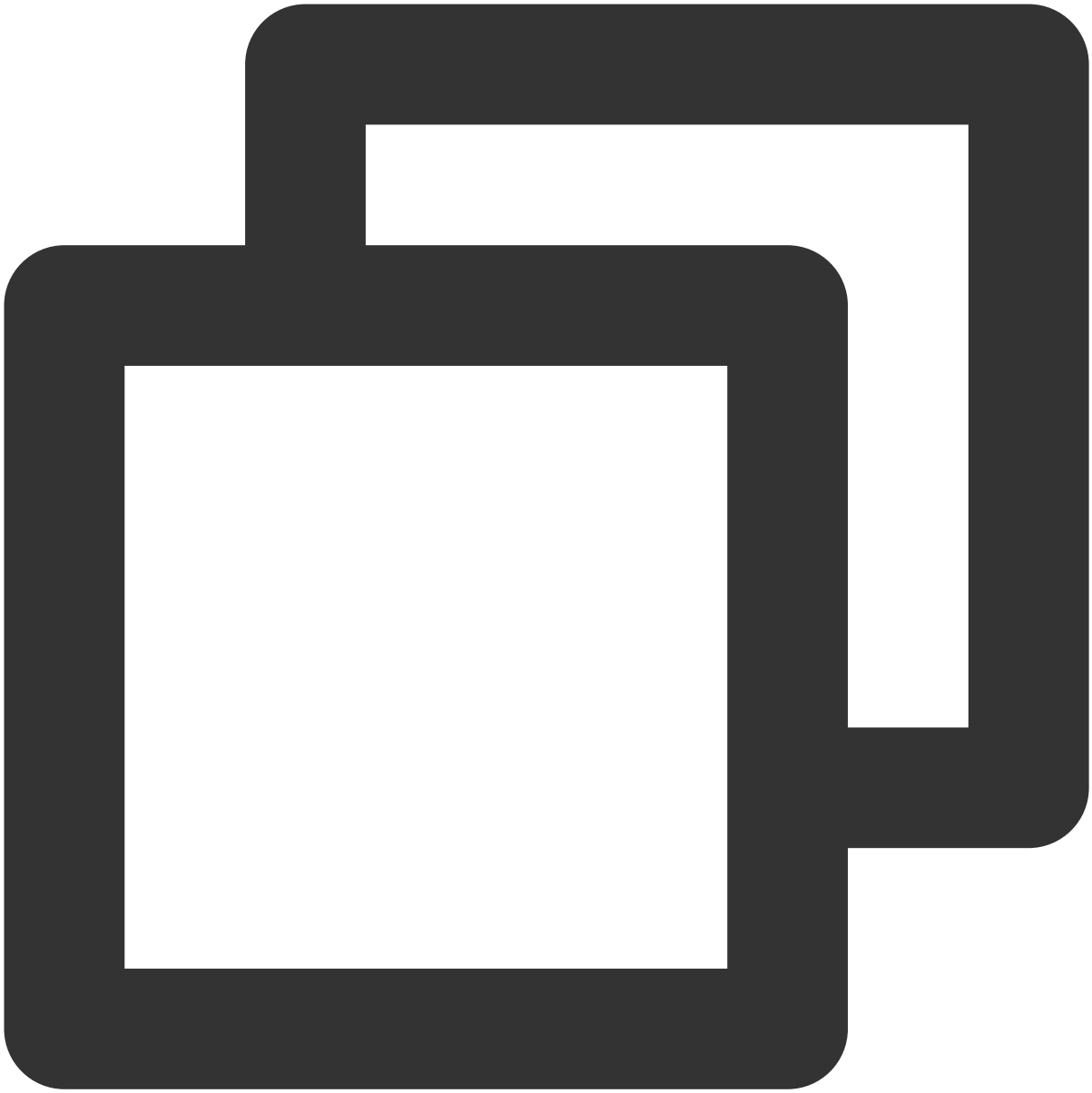
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $blindWatermarkTemplate = new Qcloud\\Cos\\ImageParamTemplate\\BlindWaterma
        $blindWatermarkTemplate->setType(3);//盲水印类型，有效值：1 半盲；2 全盲；3 文字
        $blindWatermarkTemplate->setImage("imageUrl");//设置盲水印图片地址
        $blindWatermarkTemplate->setText("Test");//设置盲水印文字
        $blindWatermarkTemplate->setLevel(3);//只对全盲水印（type=2）有效。level 的取值
        $result = $cosClient->getObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成,
        'Key' => 'exampleobject',
        'ImageHandleParam' => $blindWatermarkTemplate->queryString(),//生成盲水印参数
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
ImageHandleParam	String	盲水印参数	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [Body] =>
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 100
            [CacheControl] => max-age=2592000
            [ContentType] => image/jpeg
            [Key] => exampleobject
```

```
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
        )

)
```

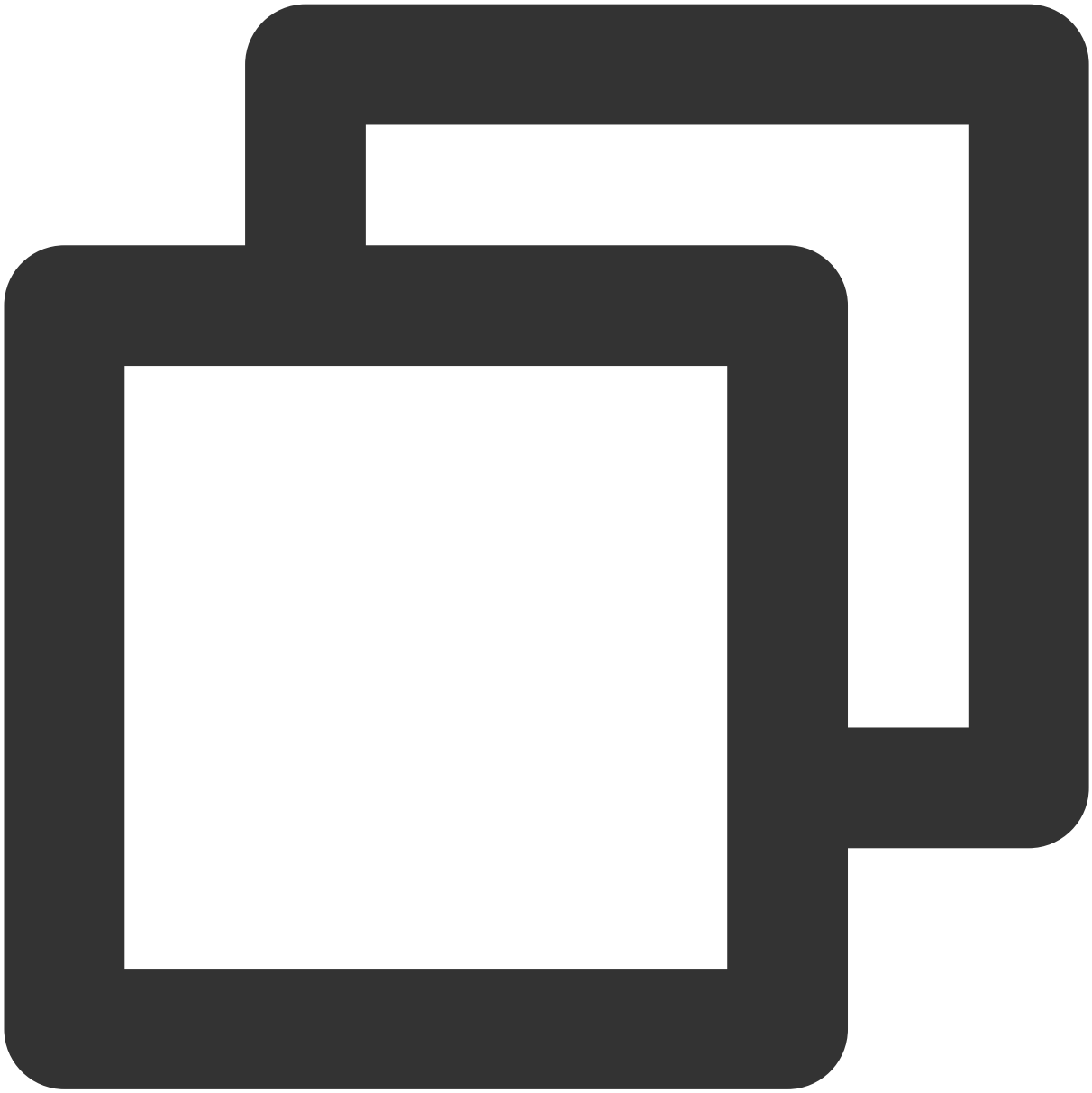
返回结果说明

参数名称	类型	描述	父节点
Body	File/String	下载内容	无
RequestId	String	请求 ID 标识	无
CacheControl	String	缓存策略，设置 Cache-Control	无
ContentType	String	内容类型，设置 Content-Type	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

提取盲水印

图片地址，根据盲水印类型填写：
1.当半盲类型时，则盲水印图片地址必填，且为原图图片地址。
2.当全盲类型时，则盲水印图片地址必填，且为水印图地址。
3.当文字类型时，则盲水印图片地址无需填写（无效）。
指定的水印图片必须同时满足如下3个条件：
1. 水印图片与源图片必须位于同一个存储桶下。
2. URL 需使用 COS 域名（不能使用 CDN 加速域名，例如 examplebucket-1250000000.file.myqcloud.com/shuiyin_2.png 不可用），且需保证水印图可访问。如果水印图读取权限为私有，则需要携带有效签名。
3. URL 必须以 `http://` 开始，不能省略 HTTP 头，也不能填 HTTPS 头，例如 `examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png`，`https://examplebucket-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png` 为非法的水印 URL。

示例代码



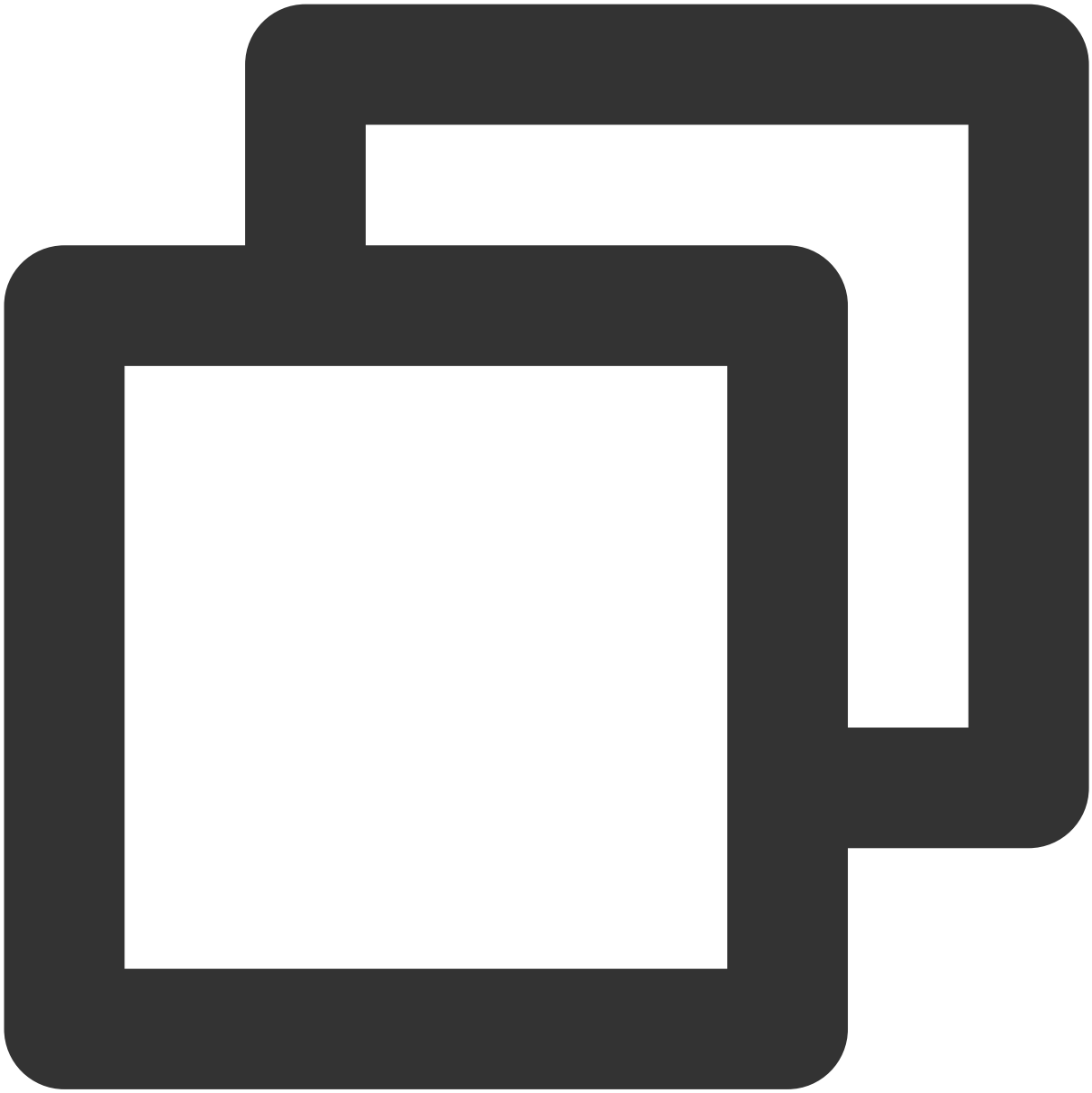
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
```

```
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $blindWatermarkTemplate = new Qcloud\\Cos\\ImageParamTemplate\\BlindWaterma
        $blindWatermarkTemplate->setPick();//设置为提取盲水印
        $blindWatermarkTemplate->setType(3);//盲水印类型，有效值：1 半盲；2 全盲；3 文字
        $blindWatermarkTemplate->setImage("imageUrl");//设置盲水印图片地址
        $blindWatermarkTemplate->setText("Test");//设置盲水印文字
        $blindWatermarkTemplate->setLevel(3);//只对全盲水印（type=2）有效。level 的取值
        $picOperationsTemplate = new \\Qcloud\\Cos\\ImageParamTemplate\\PicOperatio
        $picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，
        $picOperationsTemplate->addRule($blindWatermarkTemplate, "resultobject");//
        $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'Body' => fopen('path/to/localFile', 'rb'),
        'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
Body	File/String	上传的内容	是
PicOperations	Json/String	图片持久化处理信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [Body] =>
            [ETag] => "698d51a19d8a121ce581499d7b701668"
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 10000
```

```
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [Data] => Array
            (
                [OriginalInfo] => Array
                (
                    [Key] => exampleobject
                    [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.
                    [ETag] => "7037fb6fb4cca43b958a28789605e73d98088720"
                    [ImageInfo] => Array
                    (
                            [Format] => JPEG
                            [Width] => 600
                            [Height] => 500
                            [Quality] => 90
                            [Ave] => 0x46442e
                            [Orientation] => 0
                     )

                )
                [ProcessResults] => Array
                (
                    [Object] => Array
                    (
                        [0] => Array(
                            [Key] => resultobject
                            [Location] => examplebucket-1250000000.cos.ap-beijing.m
                            [Format] => JPEG
                            [Width] => 300
                            [Height] => 200
                            [Size] => 30000
                            [Quality] => 90
                            [WatermarkStatus] => 90
                            [ETag] => "87c153bc2909aa0ba111ca126b675c510d36b817"
                        )
                    )
                )
            )
    )
)
```

返回结果说明

参数名称	类型	描述	父节点

Body	File/String	返回体	无
ETag	String	文件的 MD5 值	无
RequestId	String	请求 ID 标识	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
Data	Array	图片处理结果信息	无

版权所有：腾讯云计算（北京）有限责任公司

图片高级压缩

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于图片高级压缩的 API 概览以及 SDK 示例代码。

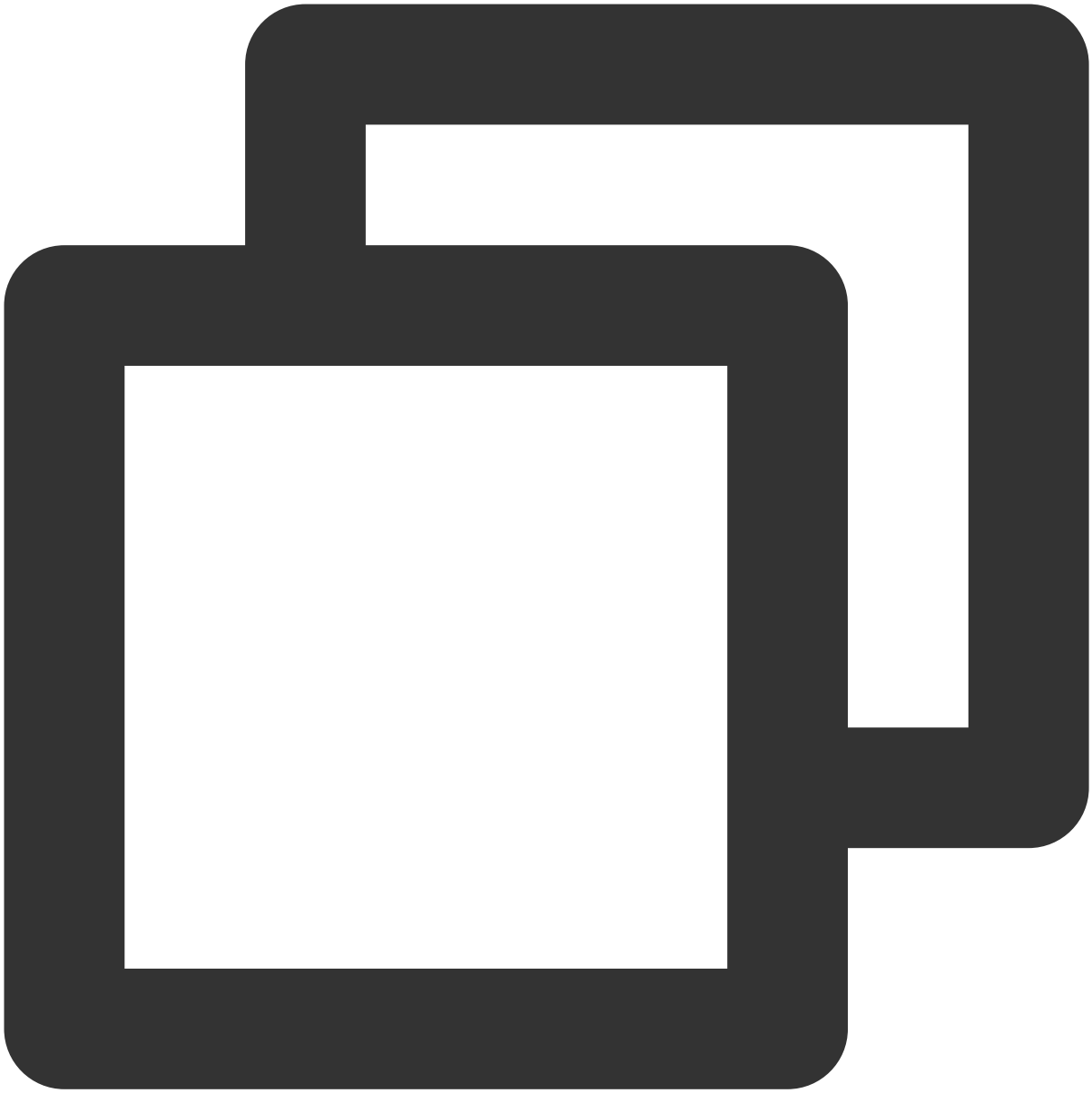
API	操作描述
图片高级压缩	图片高级压缩可以更加高效地将图片转换为 TPG 或 HEIF 等高压缩比格式，有效降低图片传输链路及加载耗时，降低带宽及流量成本

图片高级压缩

功能说明

通过数据万象 imageMogr2 接口提供图片高级压缩功能。

示例代码



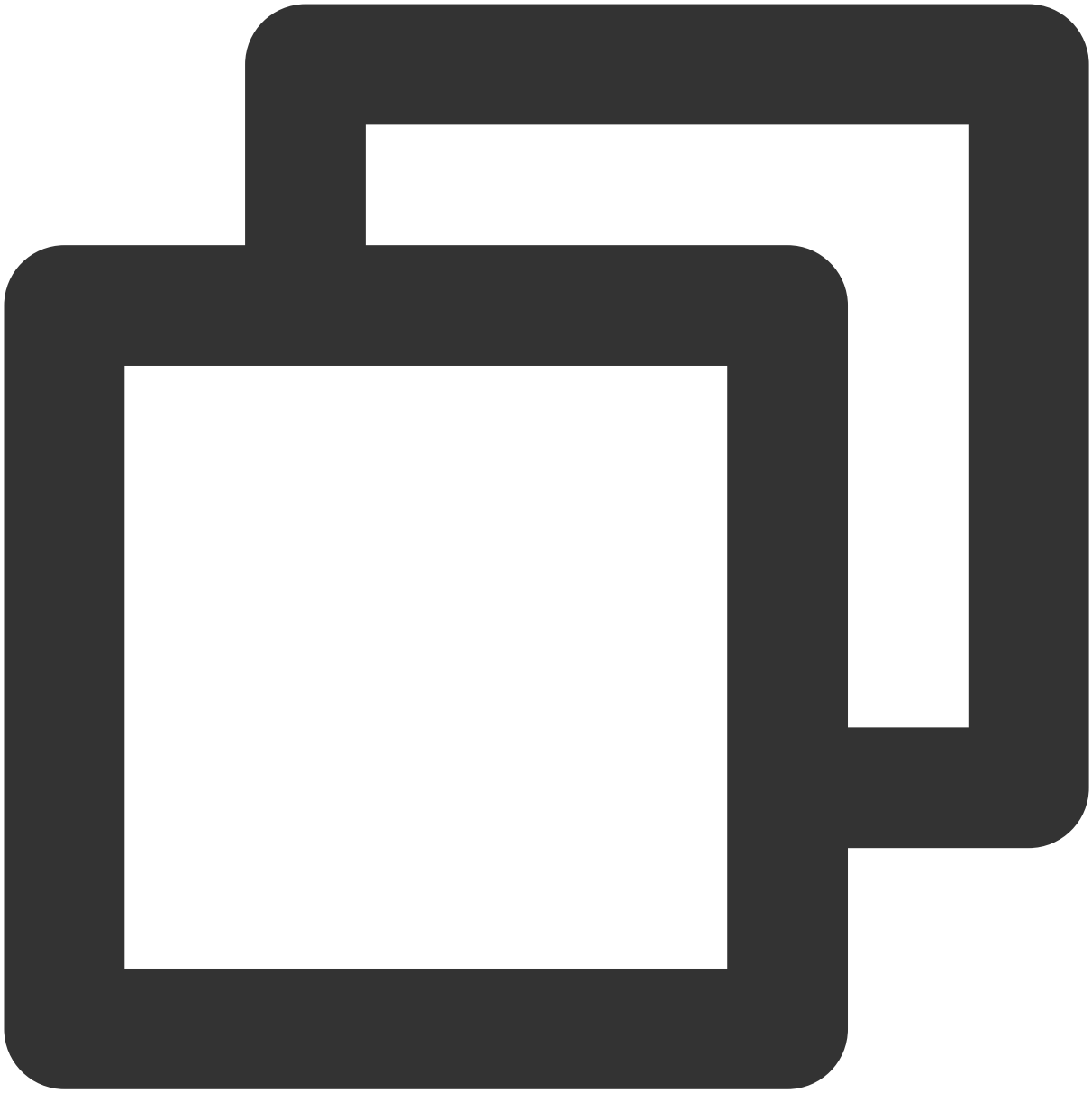
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $imageMogrTemplate = new Qcloud\\Cos\\ImageParamTemplate\\ImageMogrTemplate
        $imageMogrTemplate->format('tpg');//将原图转换为 TPG 格式,
        $imageMogrTemplate->format('heif');//将原图转换为 HEIF 格式
        $result = $cosClient->getObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成,
        'Key' => 'exampleobject',
        'ImageHandleParam' => $imageMogrTemplate->queryString(),//生成基础图片处理参数
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
ImageHandleParam	String	万象图片处理参数，例如：imageMogr2/format/tpg	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [Body] =>
            [ETag] => "698d51a19d8a121ce581499d7b701668"
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 100
            [ContentType] => image/jpeg
            [Key] => exampleobject
```

```
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
        )

)
```

返回结果说明

参数名称	类型	描述	父节点
Body	File/String	下载内容	无
ETag	String	文件的 MD5 值	无
ContentLength	Int	响应体长度	无
ContentType	String	内容类型，设置 Content-Type	无
RequestId	String	请求 ID 标识	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

版权所有：腾讯云计算（北京）有限责任公司

Guetzli 压缩

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于 Guetzli 压缩的相关的 API 概览以及 SDK 示例代码。

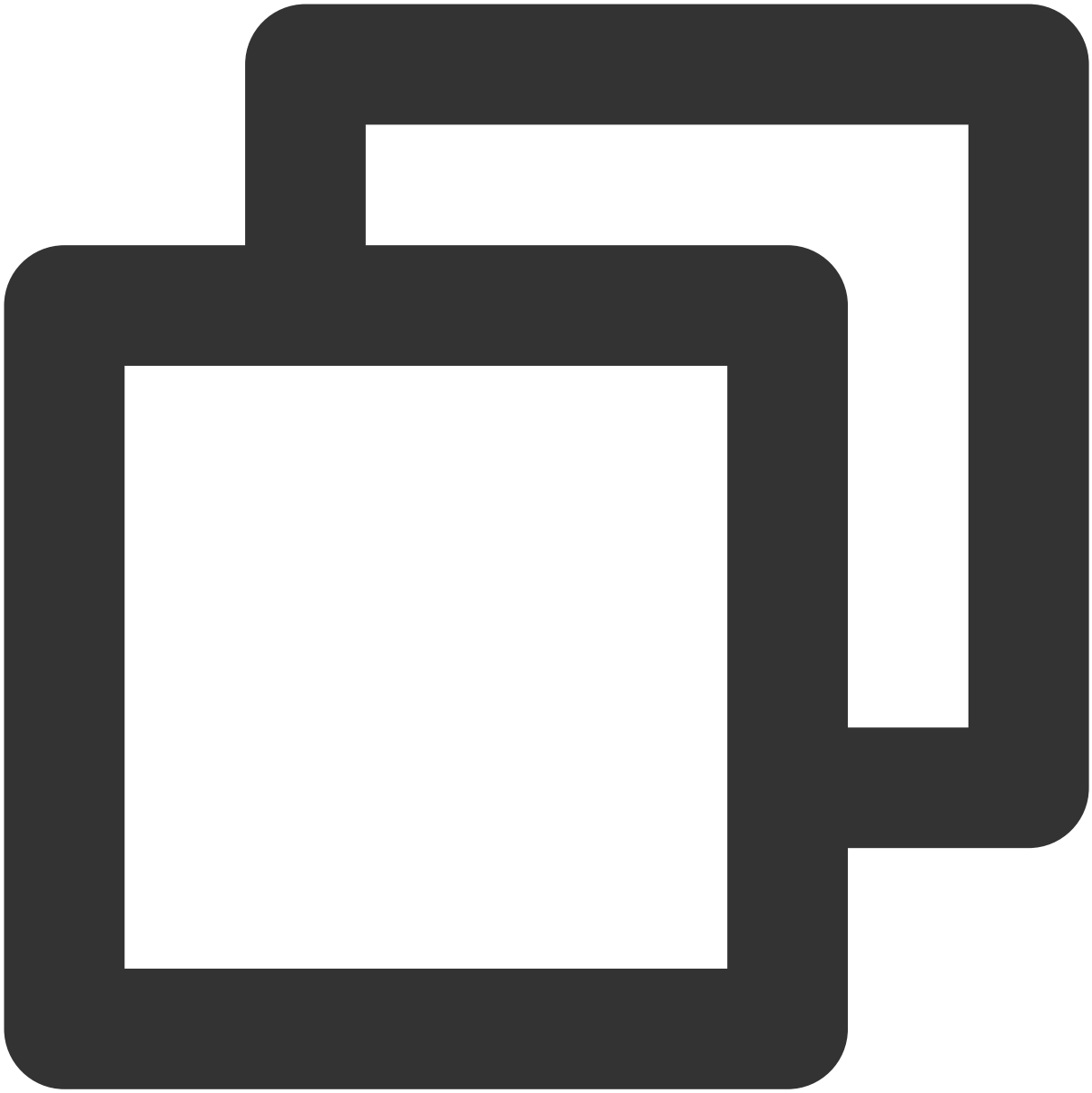
API	操作描述
开通 Guetzli 压缩	对 Bucket 开通 Guetzli 压缩功能
查询 Guetzli 状态	用于查询 Guetzli 压缩功能是否开启
关闭 Guetzli 压缩	用于关闭 Guetzli 压缩功能

开通 Guetzli 压缩

功能说明

对 Bucket 开通 Guetzli 压缩功能。

示例代码



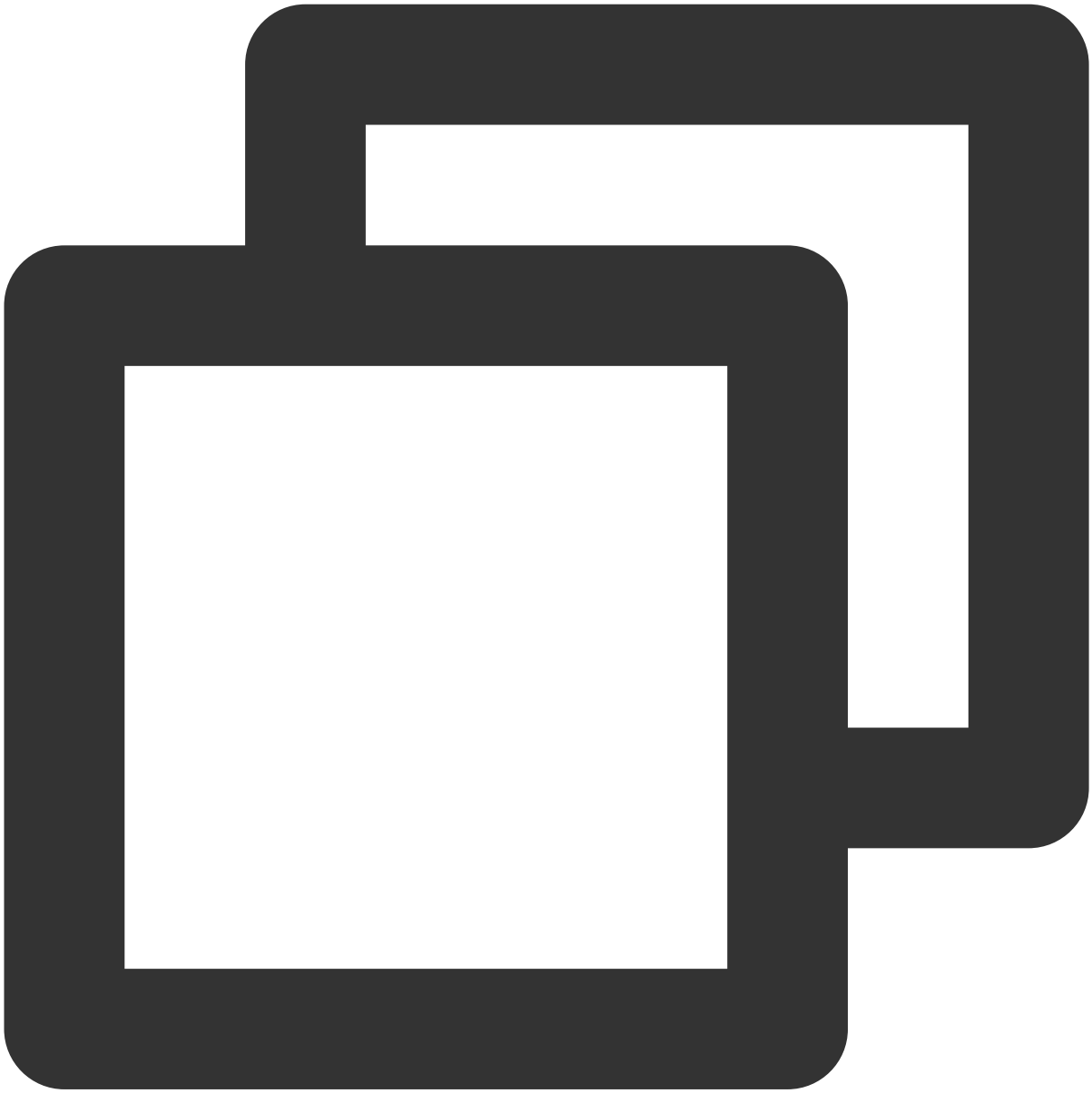
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->PutBucketGuetzli(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
        )
)
```

返回结果说明

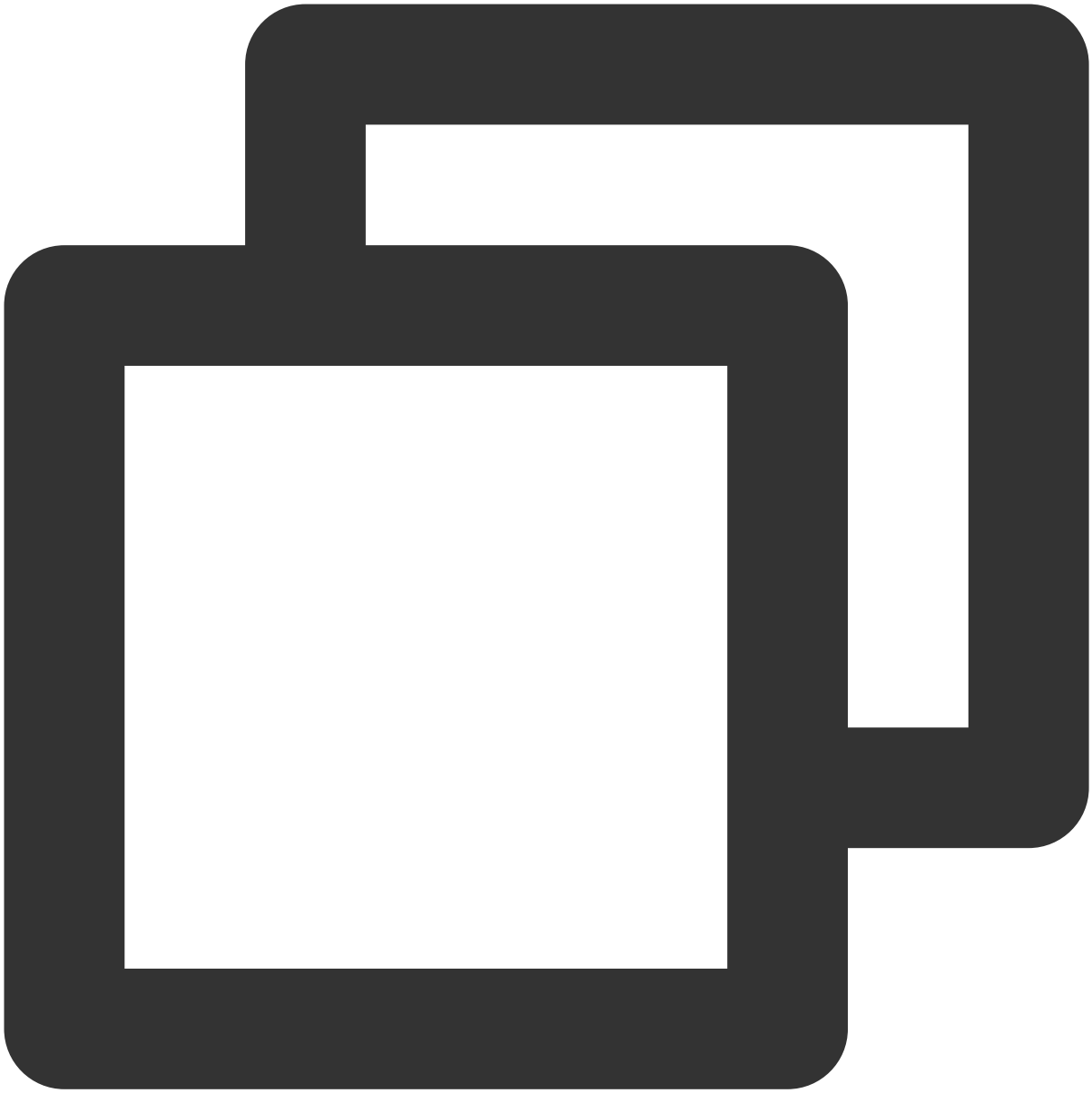
参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

查询 Guetzli 状态

功能说明

用于查询 Guetzli 压缩功能是否开启。

示例代码



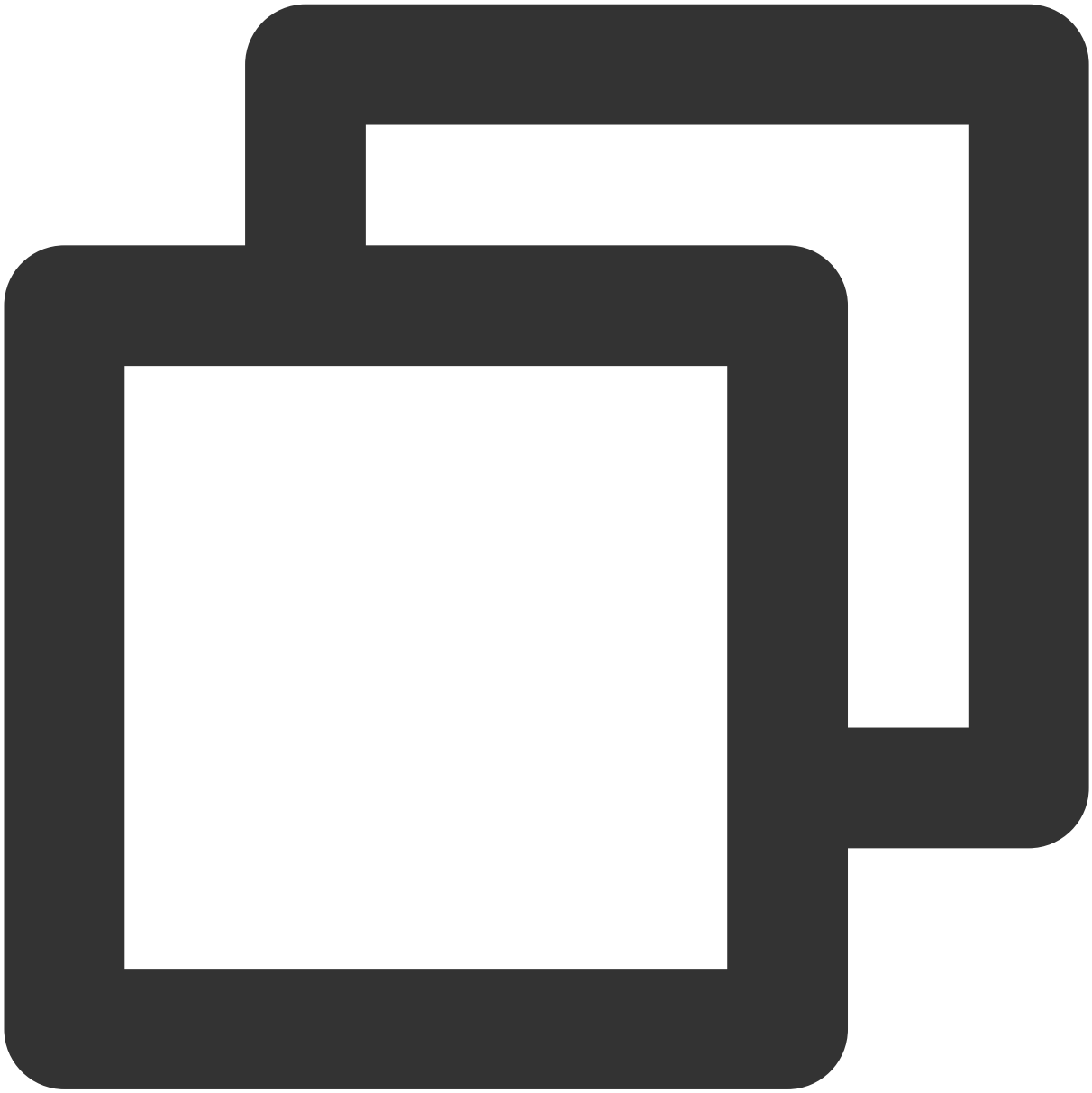
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->GetBucketGuetzli(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成,
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
            [GuetzliStatus] => on
        )
)
```

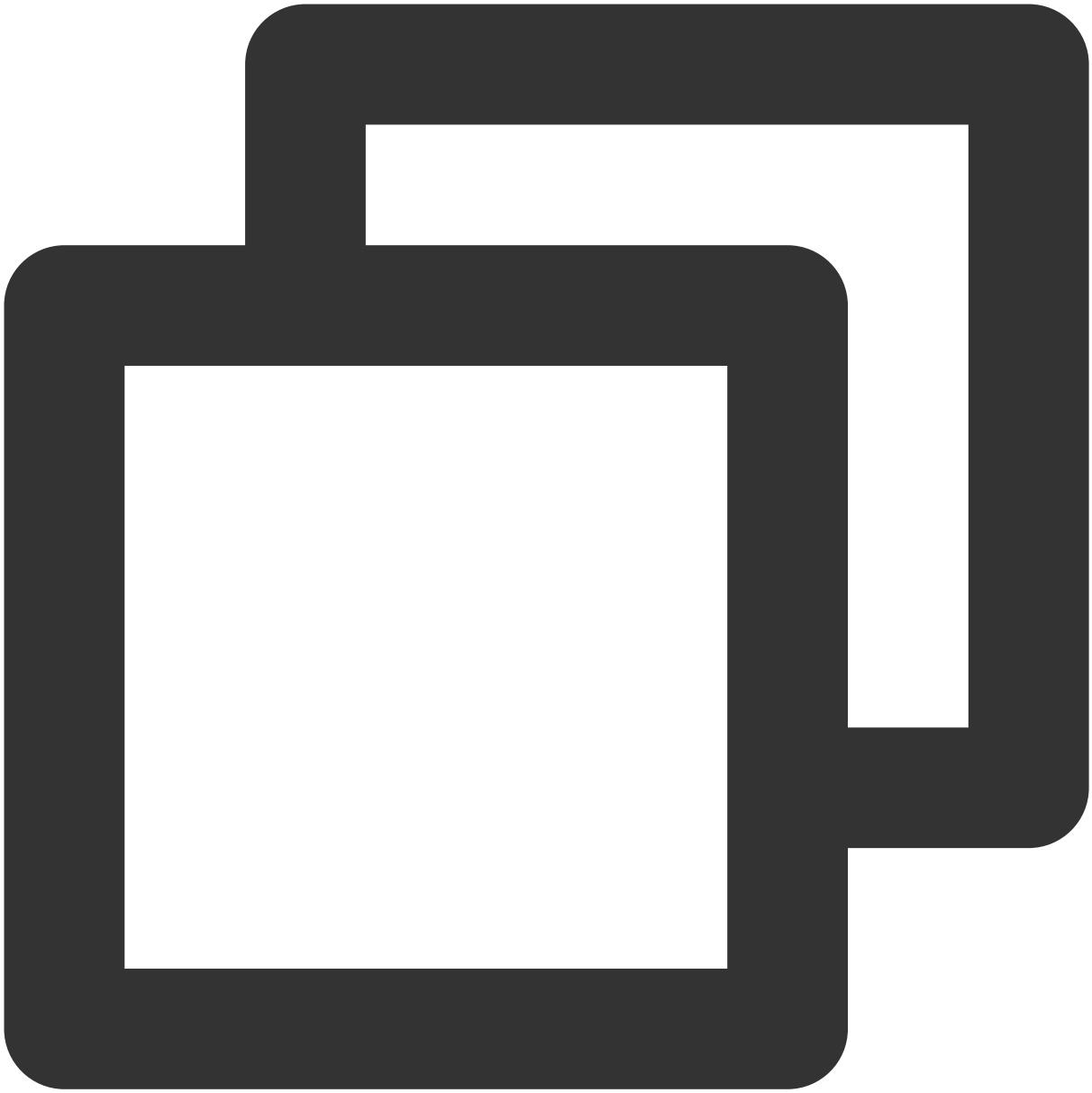
返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
GuetzliStatus	String	Guetzli 状态，on：开启，off：关闭	无

关闭 Guetzli 压缩

用于关闭 Guetzli 压缩功能。

示例代码



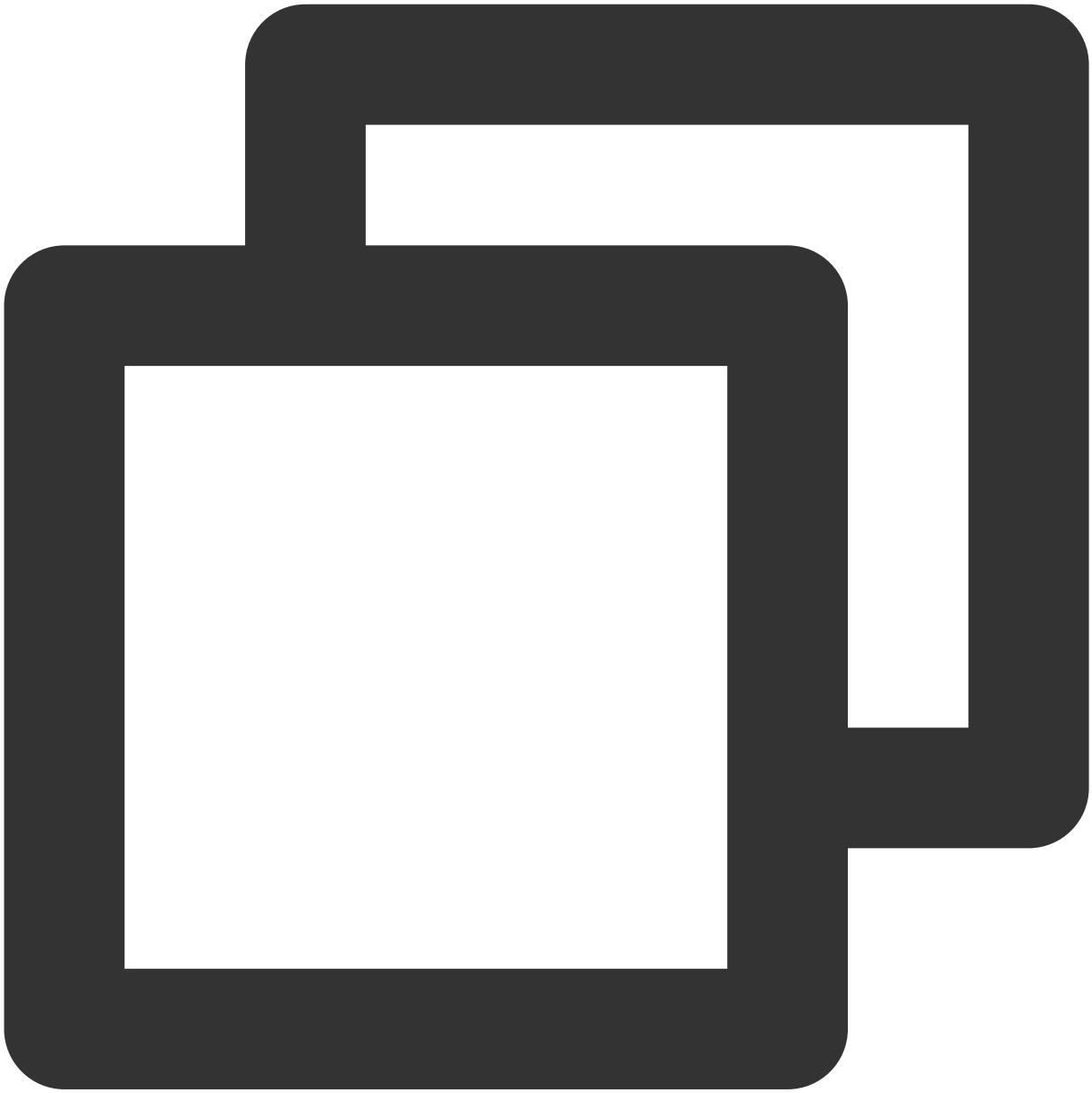
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->DeleteBucketGuetzli(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成,
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
        (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.pic.ap-beijing.myqcloud.com/
        )
)
```

返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于查询媒体处理开通状态的 API 概览和 SDK 示例代码。

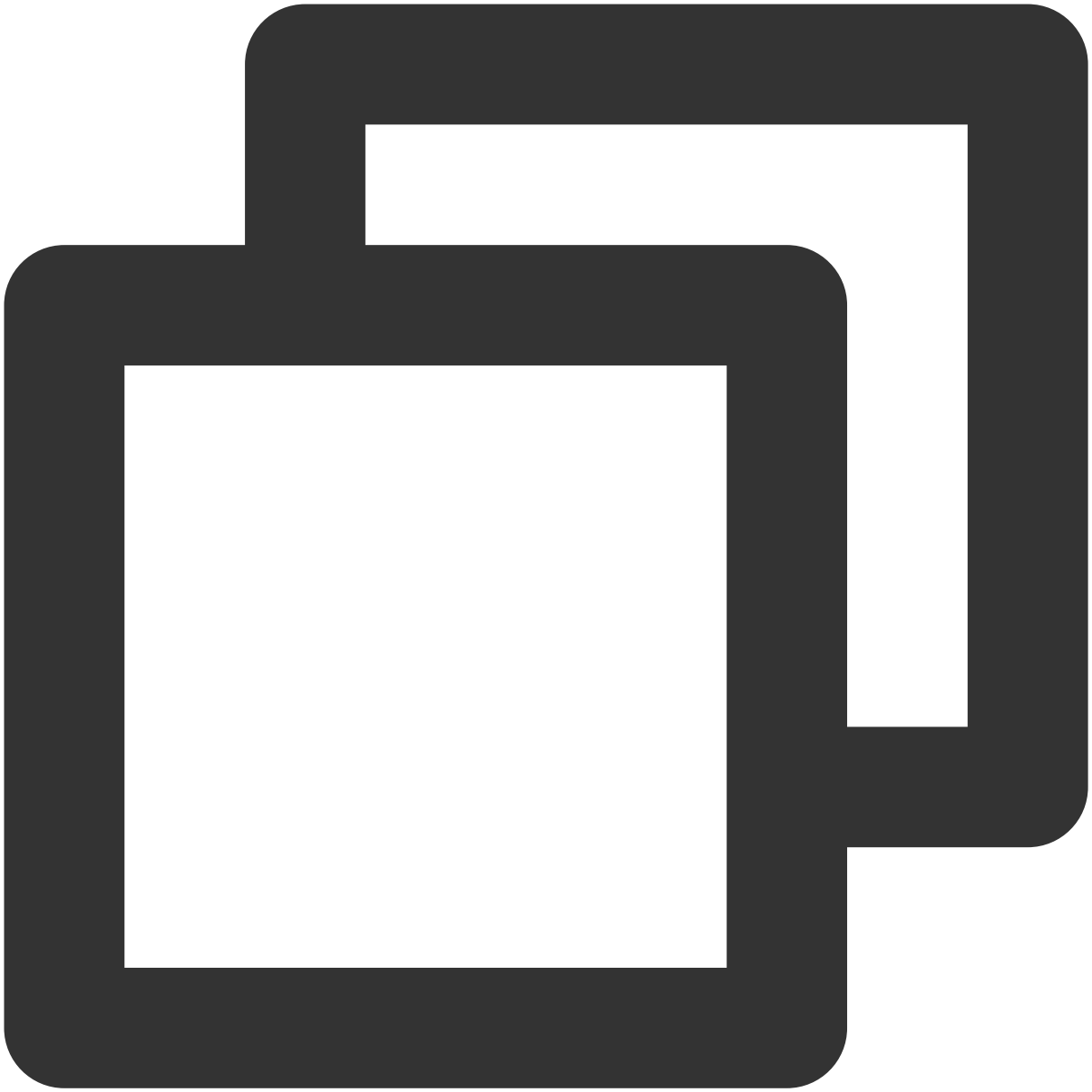
API	操作描述
查询媒体处理开通状态	查询媒体处理开通状态

查询媒体处理开通状态

功能说明

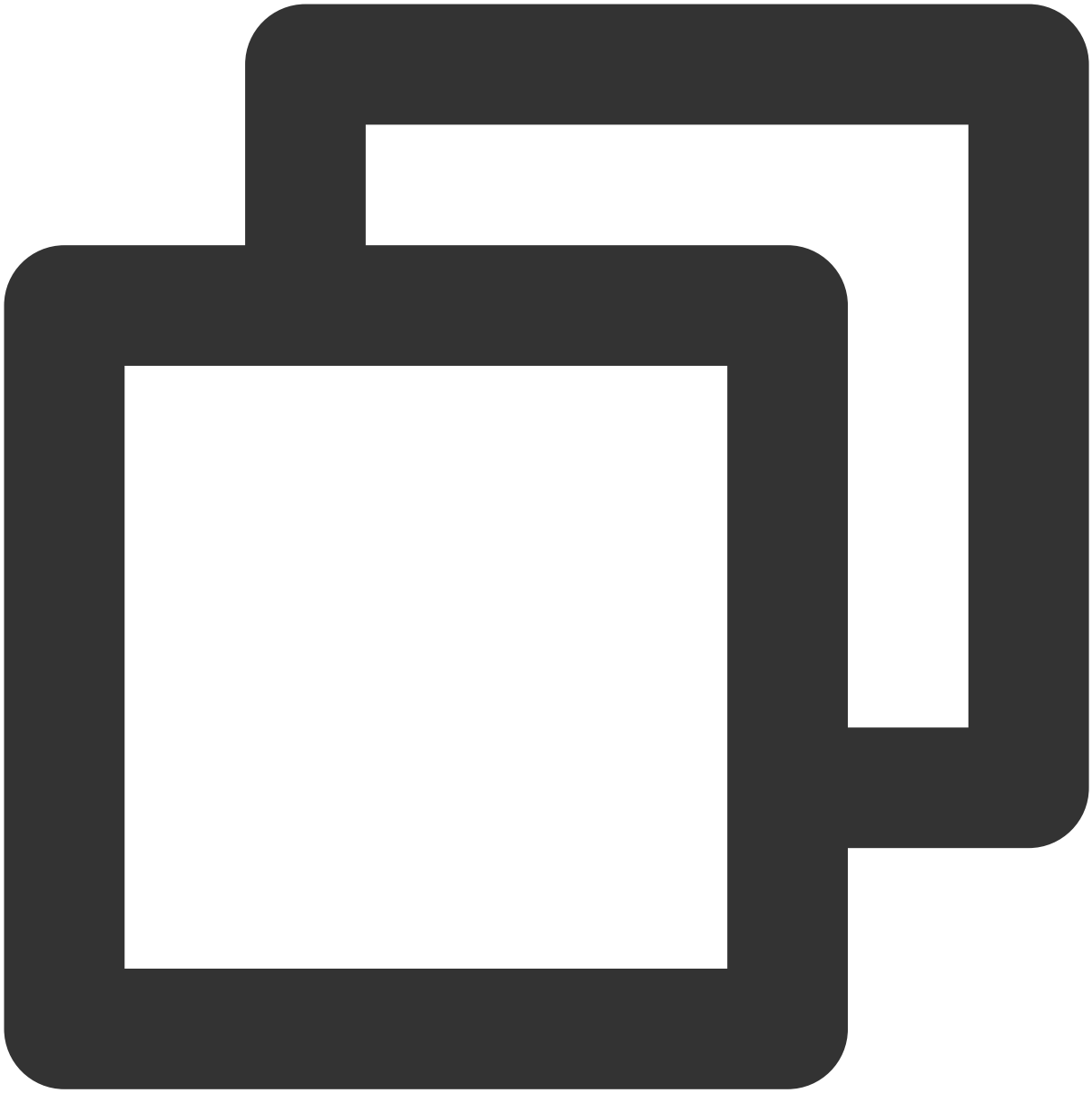
查询已经开通媒体处理功能的存储桶。

方法原型



```
public Guzzle\\Service\\Resource\\Model describeMediaBuckets(array $args = array())
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

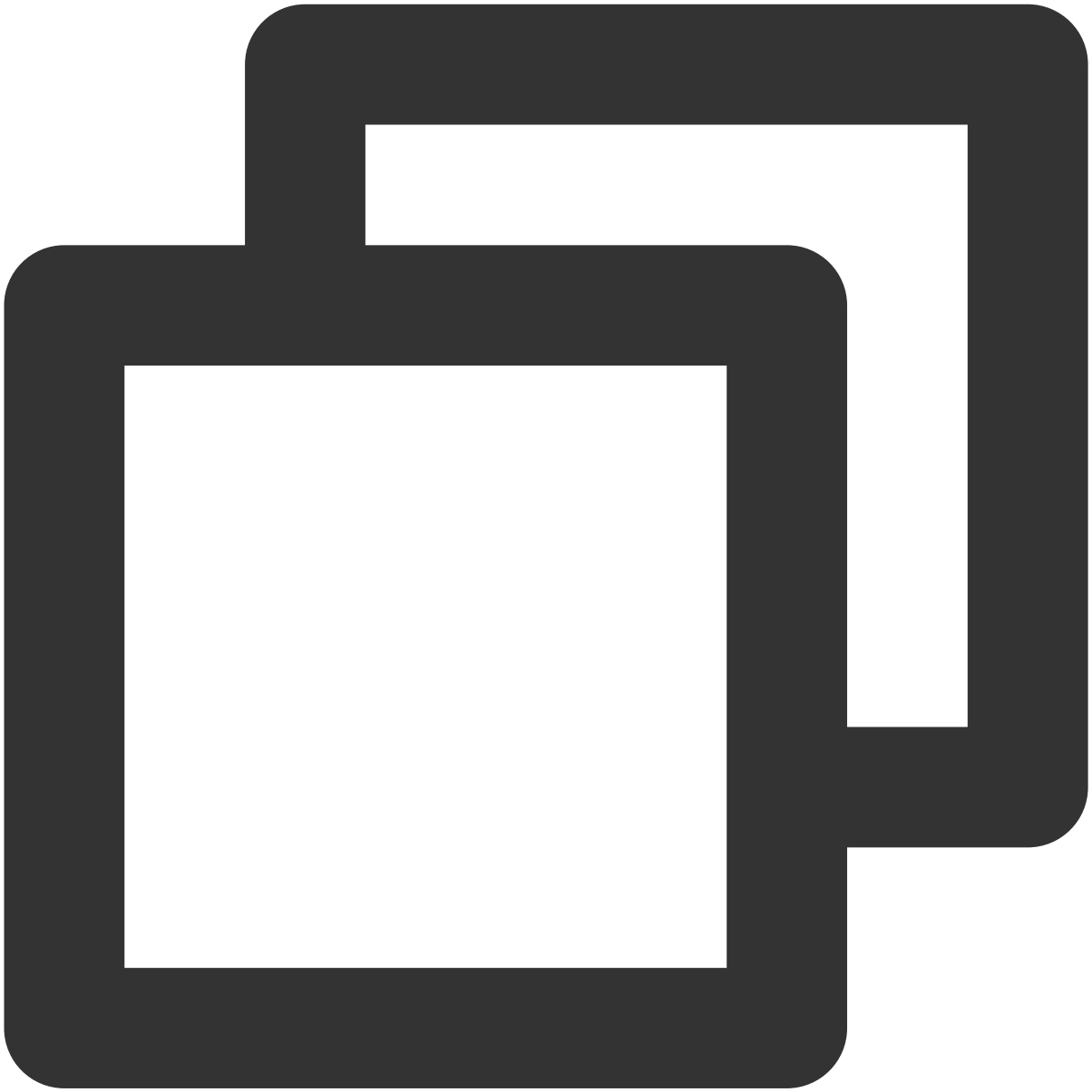
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
    $result = $cosClient->describeMediaBuckets(array(
        'Regions' => '', // 可选 地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以‘
        'BucketNames' => '', // 可选 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
        'BucketName' => '', // 可选 存储桶名称前缀，前缀搜索
        'PageNumber' => 1, // 可选 第几页
        'PageSize' => 20, // 可选 每页个数
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

名称	描述	类型	是否必选
regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	string	否
bucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string	否
bucketName	存储桶名称前缀，前缀搜索	string	否
pageNumber	第几页	string	否
pageSize	每页个数	string	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjI2MTE5OTlfNzgSxDxIHF0ZF8xZjk0MmQ=
    [ContentType] => application/xml
    [ContentLength] => 4744
    [TotalCount] => 51
    [PageNumber] => 1
    [PageSize] => 5
    [MediaBucketList] => Array
        (
            [0] => Array
```

```
                (
                    [BucketId] => examplebucket-1250000000
                    [Name] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [CreateTime] => 2022-04-07T22:06:57+0800
                )

            [1] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Name] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [CreateTime] => 2022-04-07T22:06:57+0800
                )

            [2] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Name] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [CreateTime] => 2022-04-07T22:06:57+0800
                )

            [3] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Name] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [CreateTime] => 2022-04-07T22:06:57+0800
                )

            [4] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Name] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [CreateTime] => 2022-04-07T22:06:57+0800
                )

        )

    [Location] => ci.ap-guangzhou.myqcloud.com/mediabucket
)
```

媒体信息接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供用于获取媒体文件的信息的代码示例。

注意：

需要 COS PHP SDK v2.3.2 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

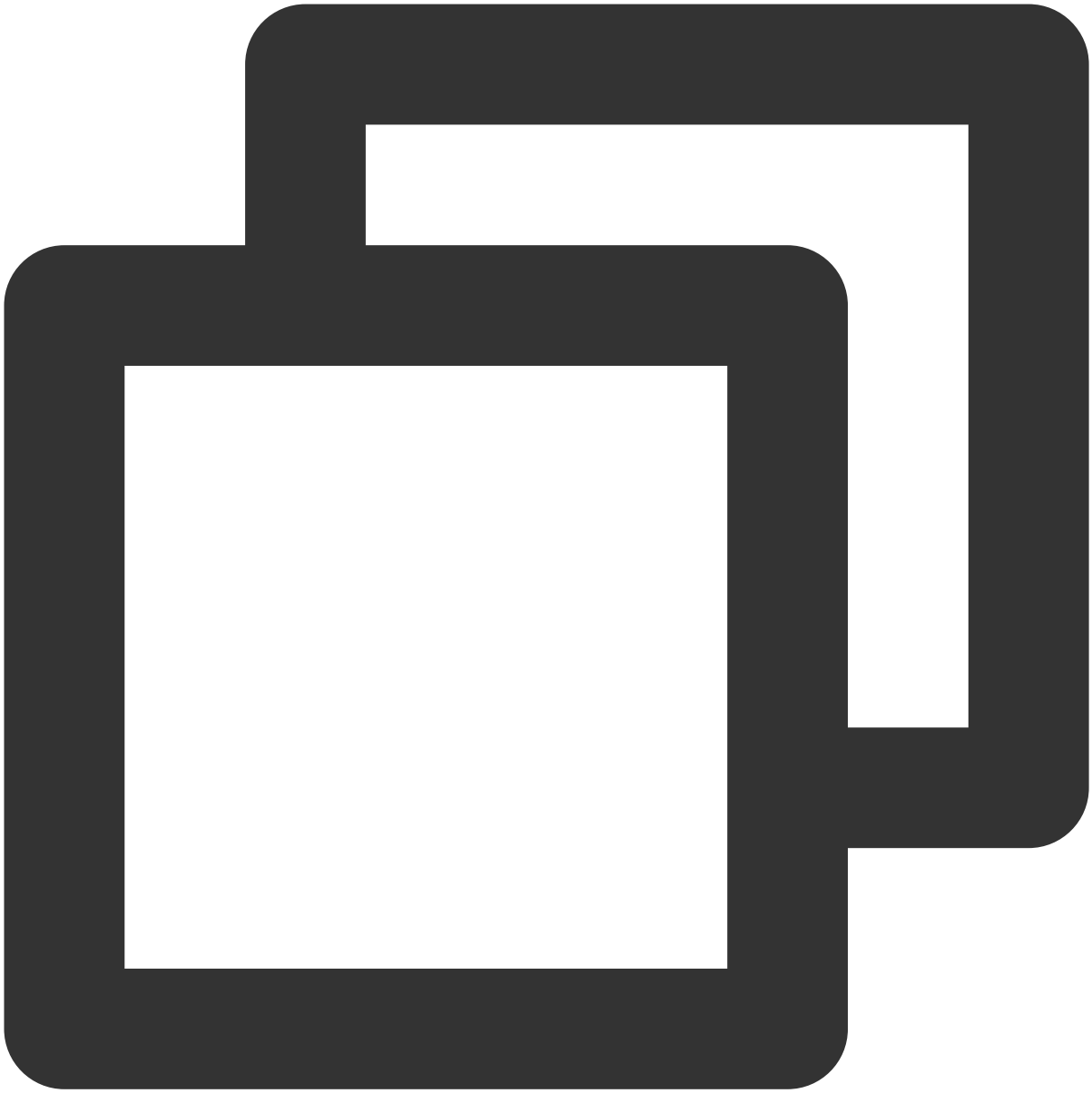
功能说明

用于查询媒体文件的信息。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



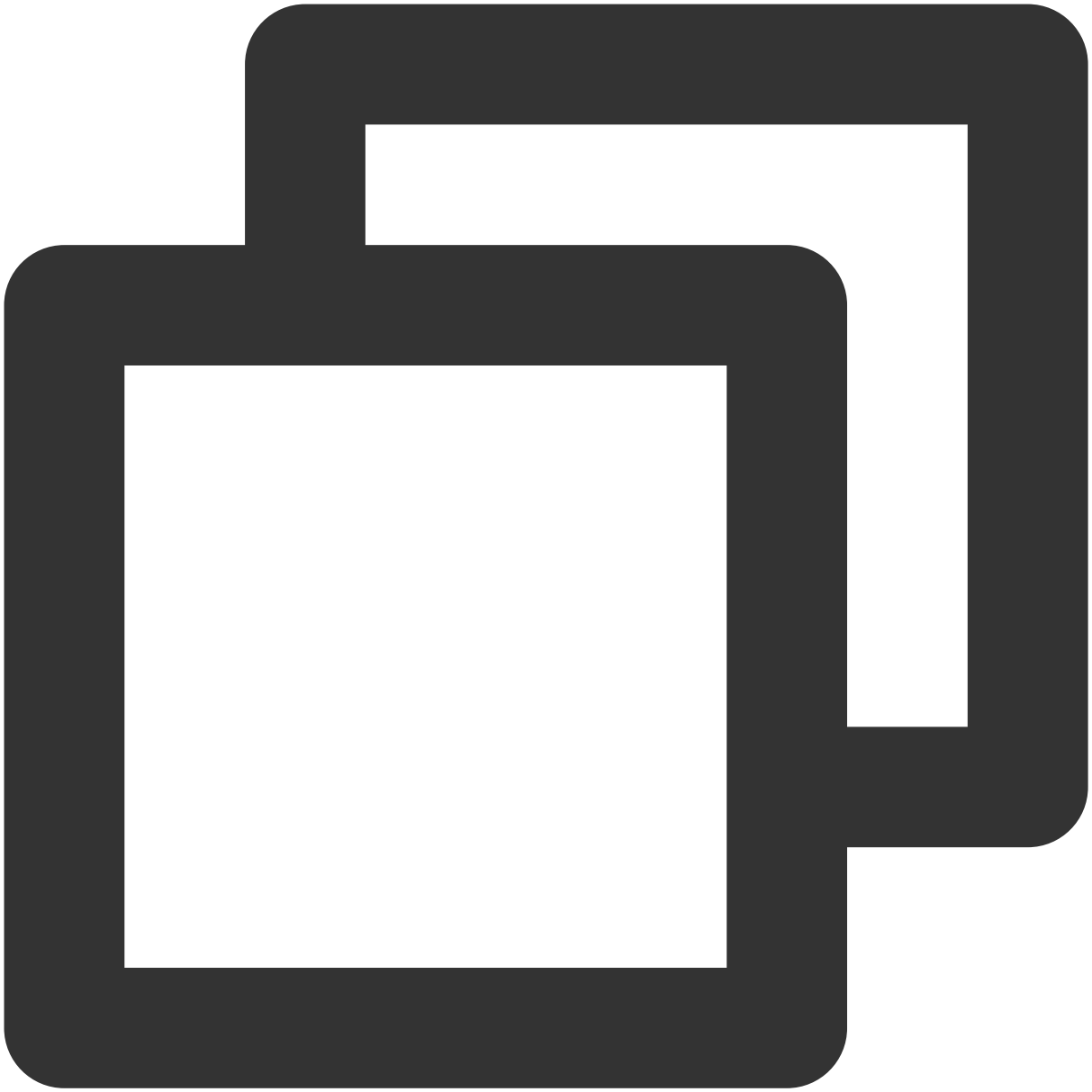
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
    $result = $cosClient->GetMediaInfo(
        array(
            'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组
            'Key' => 'exampleobject', //媒体文件对象路径,如folder/movie.mp4
            'ci-process' => 'videoinfo' //操作类型，固定使用 videoinfo
        )
    );
    // 请求成功
    echo($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名	参数描述	类型	是否必填
Bucket	存储桶的名称，命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Key	对象键（Object 的名称），这里的形式为媒体文件。对象在存储桶中的唯一标识，详情请参见 对象概述	String	是
ci-process	操作类型，固定使用 videoinfo	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjE0NTkxMWNfYTkyZTJjMGJfNTgyZF84ZDY2NWY=
    [ContentType] => application/xml
    [ContentLength] => 1787
    [MediaInfo] => Array
        (
            [Stream] => Array
                (
                    [Audio] => Array
                        (
```

```
                [Bitrate] => 383.727000
                [Channel] => 6
                [ChannelLayout] => 5.1
                [CodecLongName] => AAC (Advanced Audio Coding)
                [CodecName] => aac
                [CodecTag] => 0x6134706d
                [CodecTagString] => mp4a
                [CodecTimeBase] => 1/48000
                [Duration] => 62.314667
                [Index] => 1
                [Language] => und
                [SampleFmt] => fltp
                [SampleRate] => 48000
                [StartTime] => 0.000000
                [Timebase] => 1/48000
            )

        [Subtitle] => Array
            (
            )

        [Video] => Array
            (
                [CodecTagString] => avc1
                [Rotation] => 0.000000
                [Index] => 0
                [CodecName] => h264
                [CodecLongName] => H.264 / AVC / MPEG-4 AVC / MPEG-4 pa
                [CodecTimeBase] => 1/12800
                [CodecTag] => 0x31637661
                [Profile] => Main
                [Height] => 720
                [Width] => 1280
                [HasBFrame] => 0
                [RefFrames] => 1
                [Sar] => 1:1
                [Dar] => 16:9
                [PixFormat] => yuv420p
                [Level] => 31
                [Fps] => 25.500000
                [AvgFps] => 25/1
                [Timebase] => 1/12800
                [StartTime] => 0.000000
                [Duration] => 62.280000
                [Bitrate] => 959.963000
                [NumFrames] => 1557
                [Language] => und
```

```
                        )

                )

            [Format] => Array
                (
                    [NumStream] => 2
                    [NumProgram] => 0
                    [FormatName] => mov,mp4,m4a,3gp,3g2,mj2
                    [FormatLongName] => QuickTime / MOV
                    [StartTime] => 0.000000
                    [Duration] => 62.315000
                    [Bitrate] => 1347.820000
                    [Size] => 10498677
                )

        )

    [Key] => exampleobject
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobj
)
```

媒体截图接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供获取媒体文件某个时间的截图的代码示例。

注意：

需要 COS PHP SDK v2.3.2 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

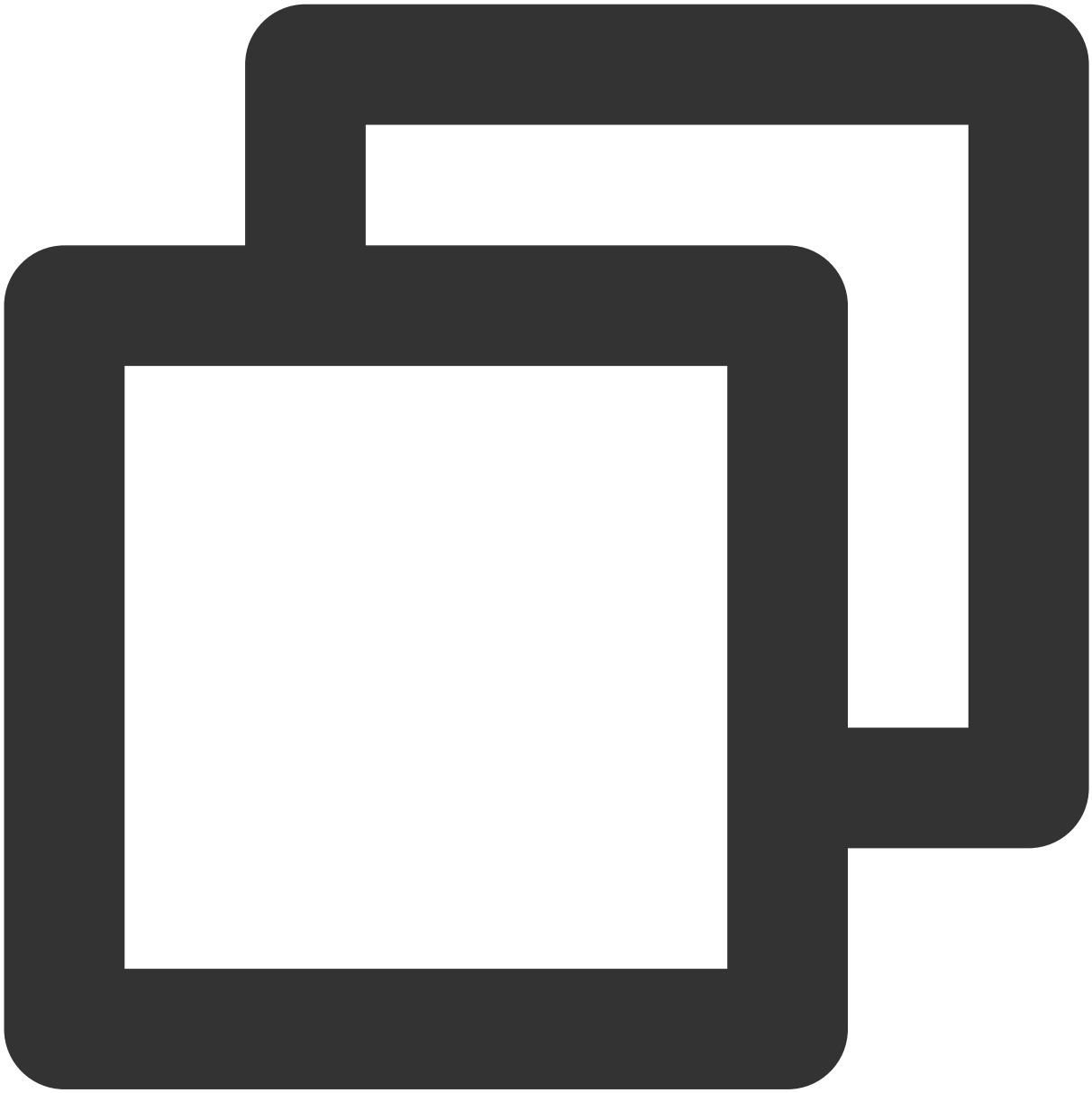
API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

功能说明

用于查询媒体文件在某个时间的截图。

使用示例



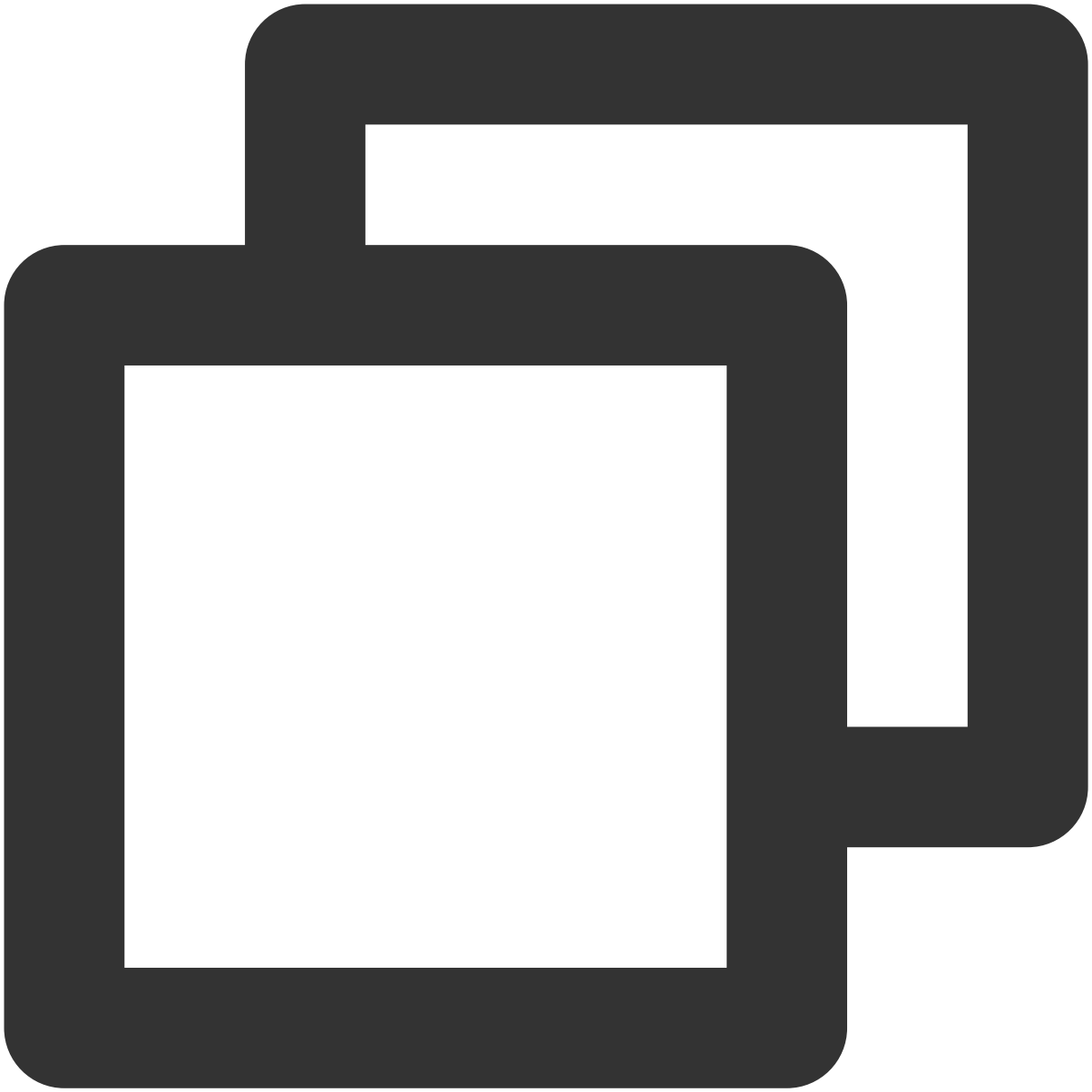
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

$time = 3.14; //视频截帧时间点
try {
    $result = $cosClient->getSnapshot(
        array(
            'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组
            'Key' => 'exampleobject', //媒体文件对象路径,如folder/movie.mp4
            'ci-process' => 'snapshot', //操作类型，固定使用 snapshot
            'Time' => $time, //视频截帧时间点
            'SaveAs' => '/data/test.jpg', //截图保存到本地的路径
        )
    );
    // 请求成功
    echo($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名	参数描述	类型	是否必填
Bucket	存储桶的名称，命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Key	对象键（Object 的名称），这里的形式为媒体文件。对象在存储桶中的唯一标识，详情请参见 对象概述	String	是
ci-process	操作类型，固定使用 snapshot	String	是
Time	视频截帧时间点	float	是
SaveAs	截图保存到本地的路径	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #91
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

```
                    (
                    )

            )

        [ContentLength] => 153613
        [ContentType] => image/jpeg
        [RequestId] => NjE0NThmNGNfNWIyZjJjMGJfNTE1MV84YmJlODI=
        [Key] => exampleobject
        [Bucket] => examplebucket-1250000000
        [Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobj
)
```

私有 M3U8 接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于私有 M3U8 接口的 API 概览和 SDK 示例代码。

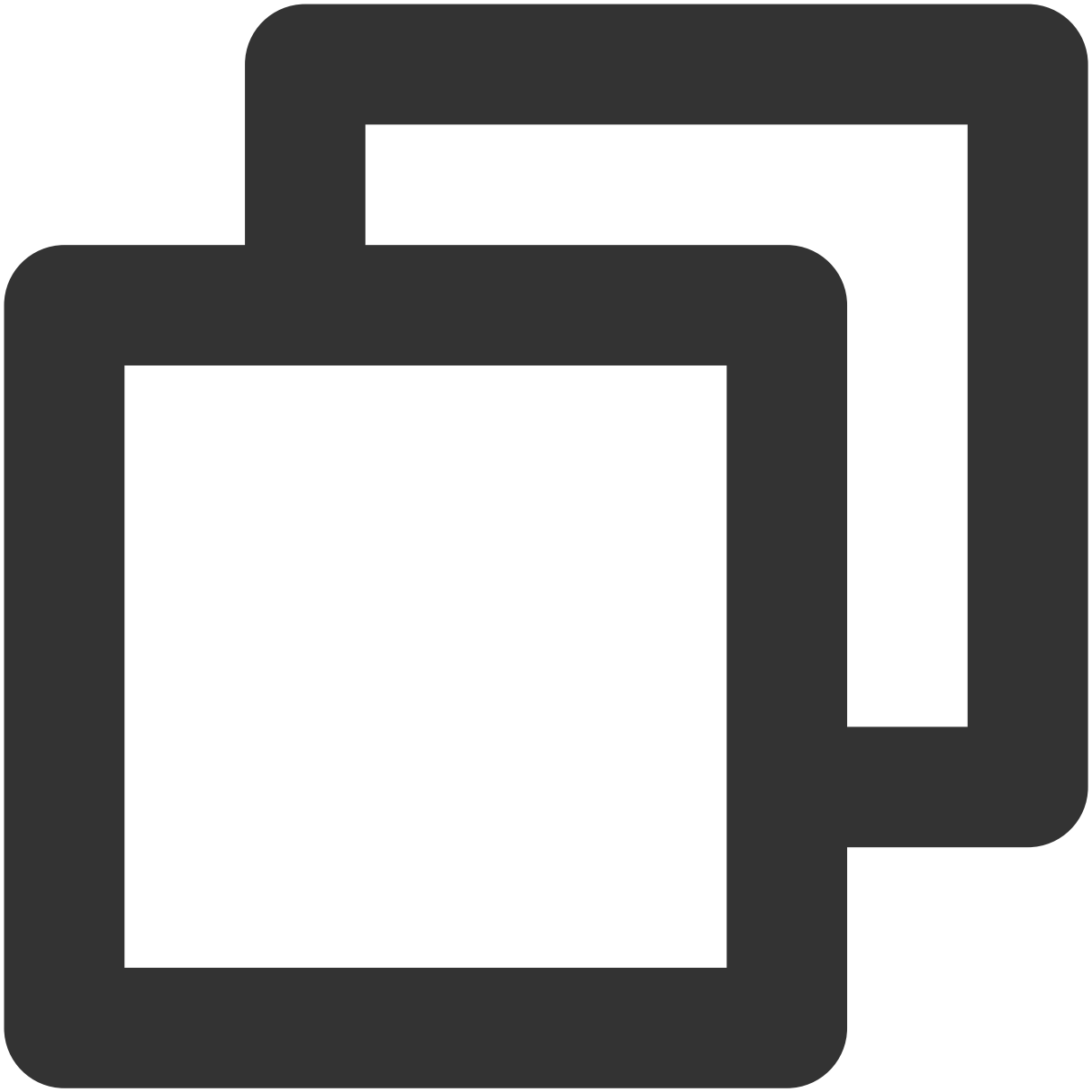
API	操作描述
GetPrivateM3U8	用于获取私有 M3U8 ts 资源的下载授权。（此方式通过对象存储转发请求至数据万象）。

GetPrivateM3U8

功能说明

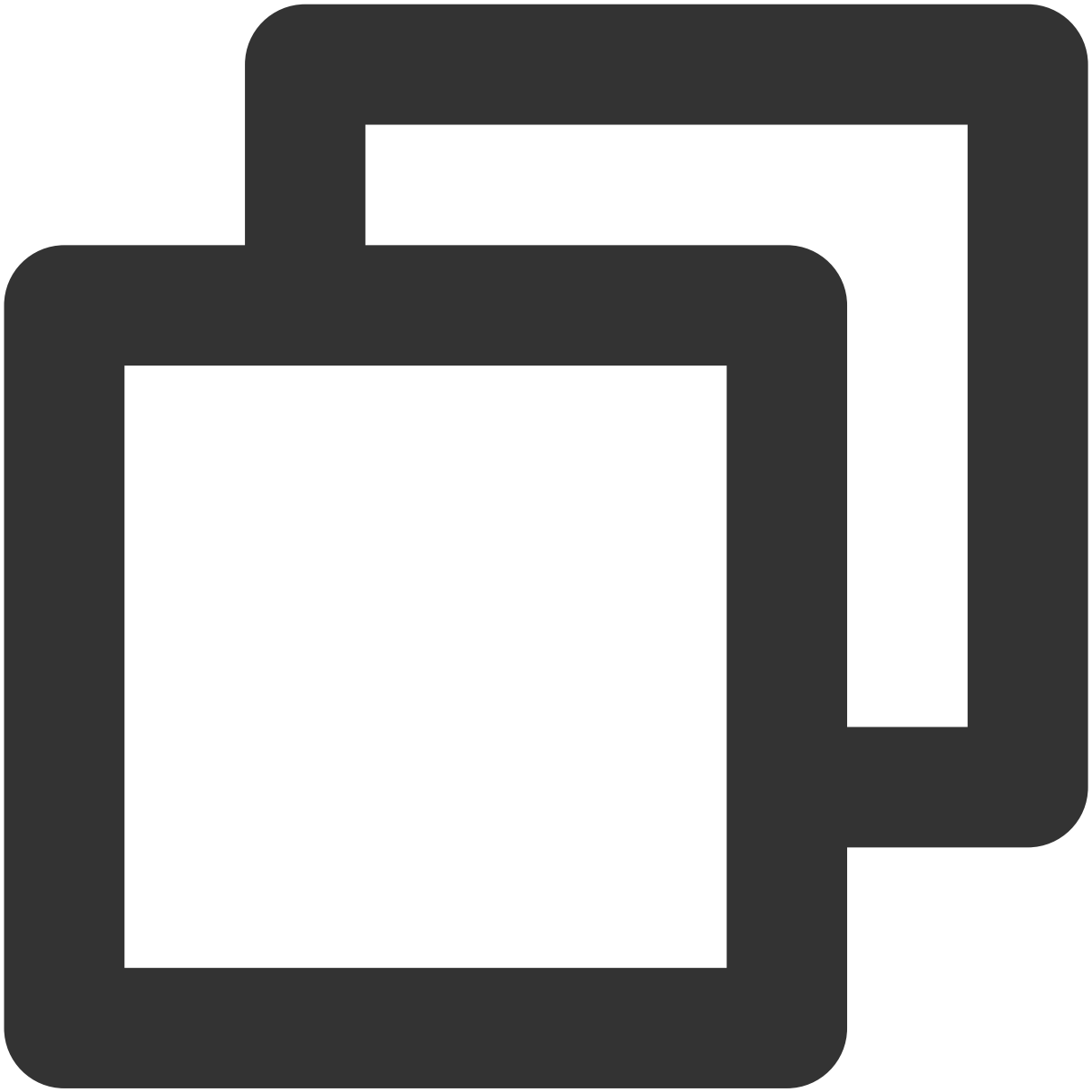
用于获取私有 M3U8 ts 资源的下载授权。（此方式通过对象存储转发请求至数据万象）。

方法原型



```
public Guzzle\\Service\\Resource\\Model getPrivateM3U8(array $args = array());
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

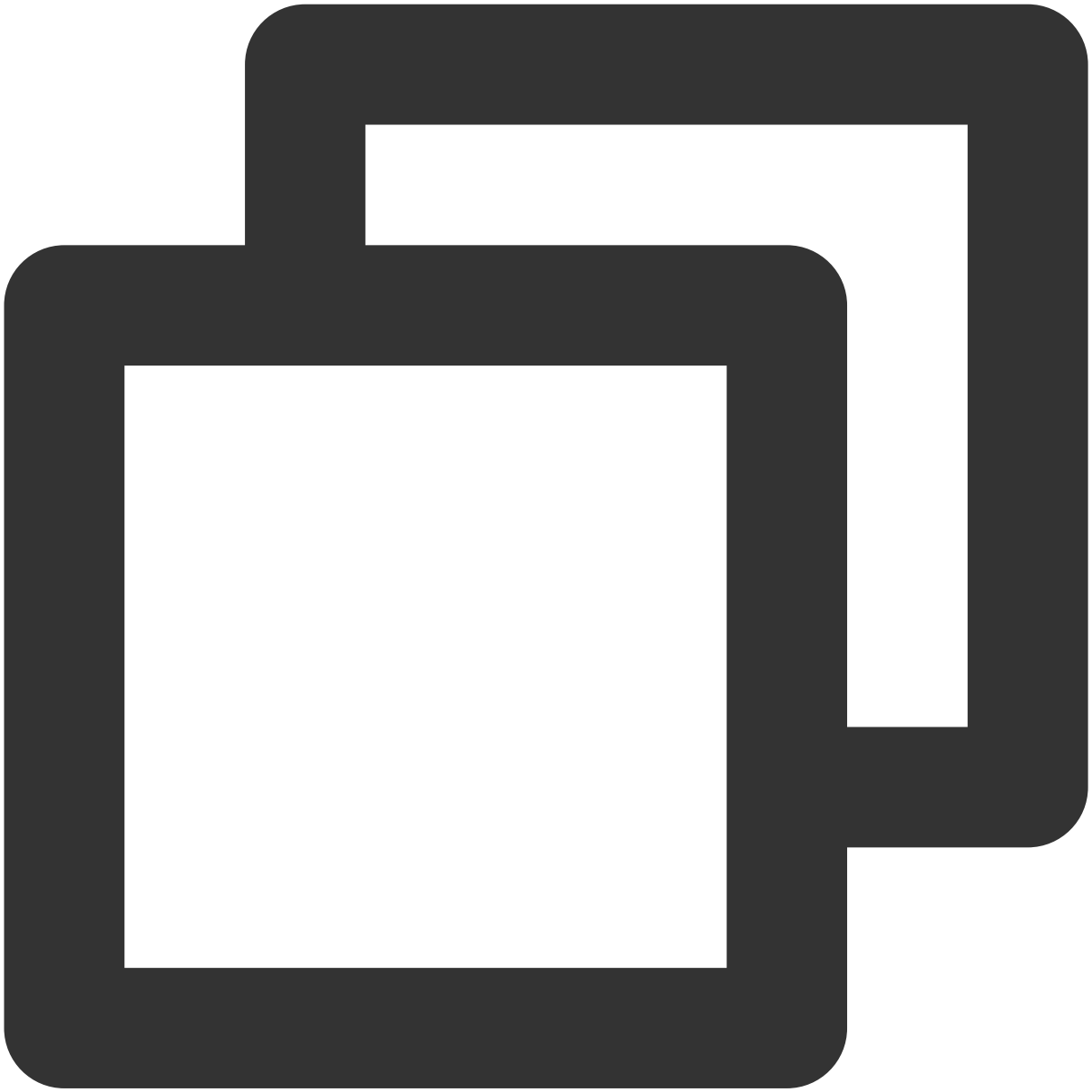
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
    $result = $cosClient->GetPrivateM3U8(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'xxx.m3u8', // 桶文件
        'ci-process' => 'pm3u8', // 操作类型，固定使用 pm3u8
        'expires' => '3600', // 私有 ts 资源 url 下载凭证的相对有效期，单位为秒，范围为[360
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	描述	类型	是否必选
ci-process	操作类型，固定使用 pm3u8	String	是
expires	私有 ts 资源 url 下载凭证的相对有效期，单位为秒，范围为[3600, 43200]	String	是

返回结果示例



```
 GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #89
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                    (
                    )

            )

        [RequestId] => NjI2MTFjYSDHIUsdsdSJDOIDTRfMjVjYzcw
        [ContentType] => application/x-mpegURL
        [ContentLength] => 4310
        [Key] => xxx.m3u8
        [Bucket] => examplebucket-1250000000
        [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/xxx.m3u8
        [Data] => m3u8文件内容
)
```

队列接口

最近更新时间：2024-03-14 09:46:17

简介

本文档提供关于媒体处理队列接口的 API 概览和 SDK 示例代码。

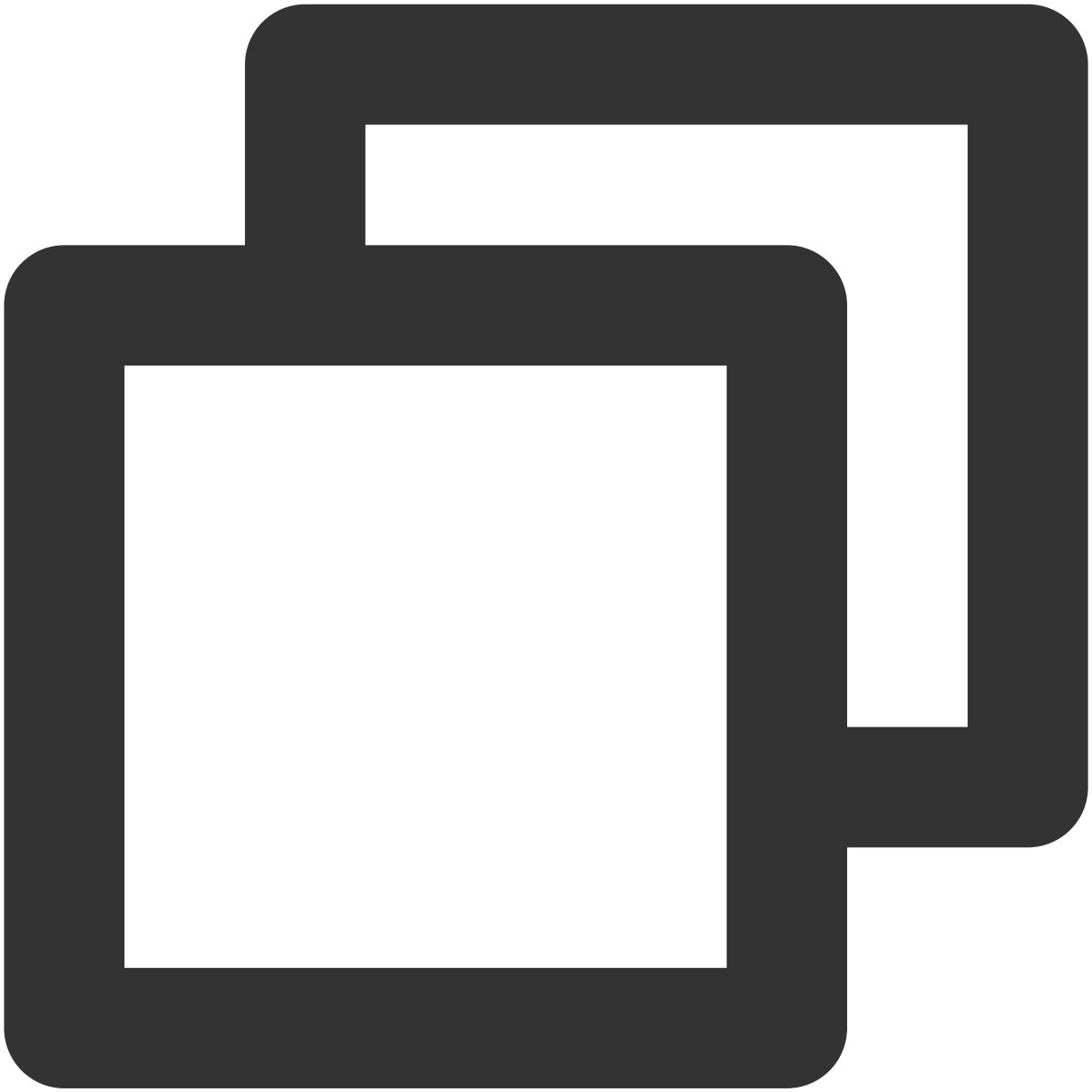
API	操作描述
搜索媒体处理队列	搜索媒体处理队列。
更新媒体处理队列	更新媒体处理队列。

搜索媒体处理队列

功能说明

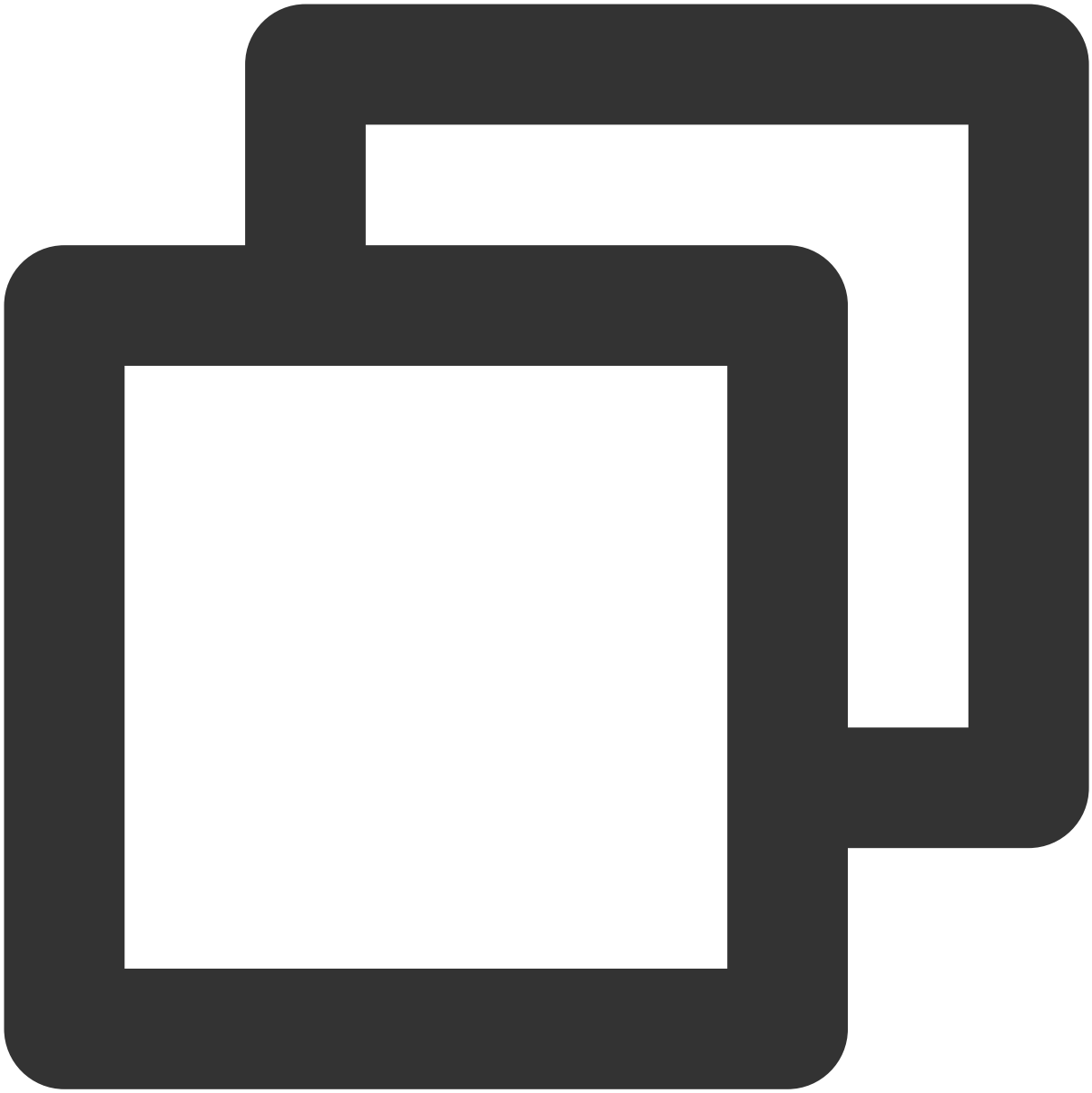
搜索媒体处理队列。

方法原型



```
public Guzzle\\Service\\Resource\\Model describeMediaQueues(array $args = array());
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

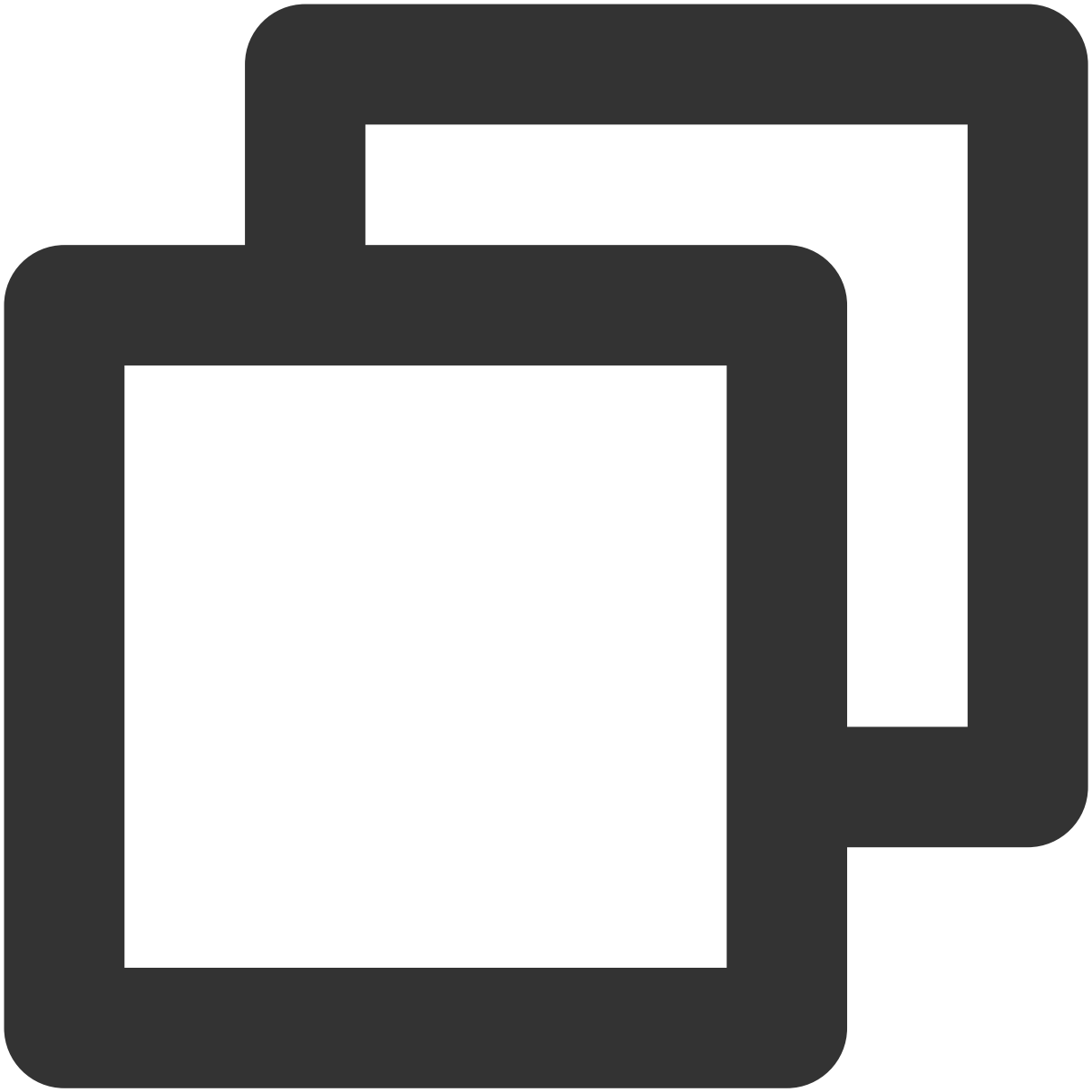
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {

    $result = $cosClient->describeMediaQueues(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
//        'QueueIds' => '', // 可选 队列 ID，以“,”符号分割字符串
//        'Category' => 'Transcoding', // 可选 CateAll：所有类型；Transcoding：媒体处理|
//        'State' => 'Paused', // 可选 1. Active 表示队列内的作业会被媒体转码服务调度转码执
//        'PageNumber' => '1', // 可选 第几页
//        'PageSize' => '2', // 可选 每页个数
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

请求参数描述如下：

参数名称（关键字）	描述	类型	是否必选
QueueIds	队列 ID，以“,”符号分割字符串。	String	否
State	Active 表示队列内的作业会被媒体处理服务调度执行。 Paused 表示队列暂停，作业不再会被媒体处理调度执行，队列内的所有作业状态维持在暂停状态，已经执行中的任务不受影响。	String	否
Category	CateAll：所有类型。 Transcoding：媒体处理队列。 SpeedTranscoding：媒体处理倍速转码队列。 默认为 Transcoding。	String	否
PageNumber	第几页，默认值1。	String	否
PageSize	每页个数，默认值10。	String	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 751
    [TotalCount] => 1
    [PageNumber] => 1
    [PageSize] => 10
    [QueueList] => Array
        (
            [0] => Array
```

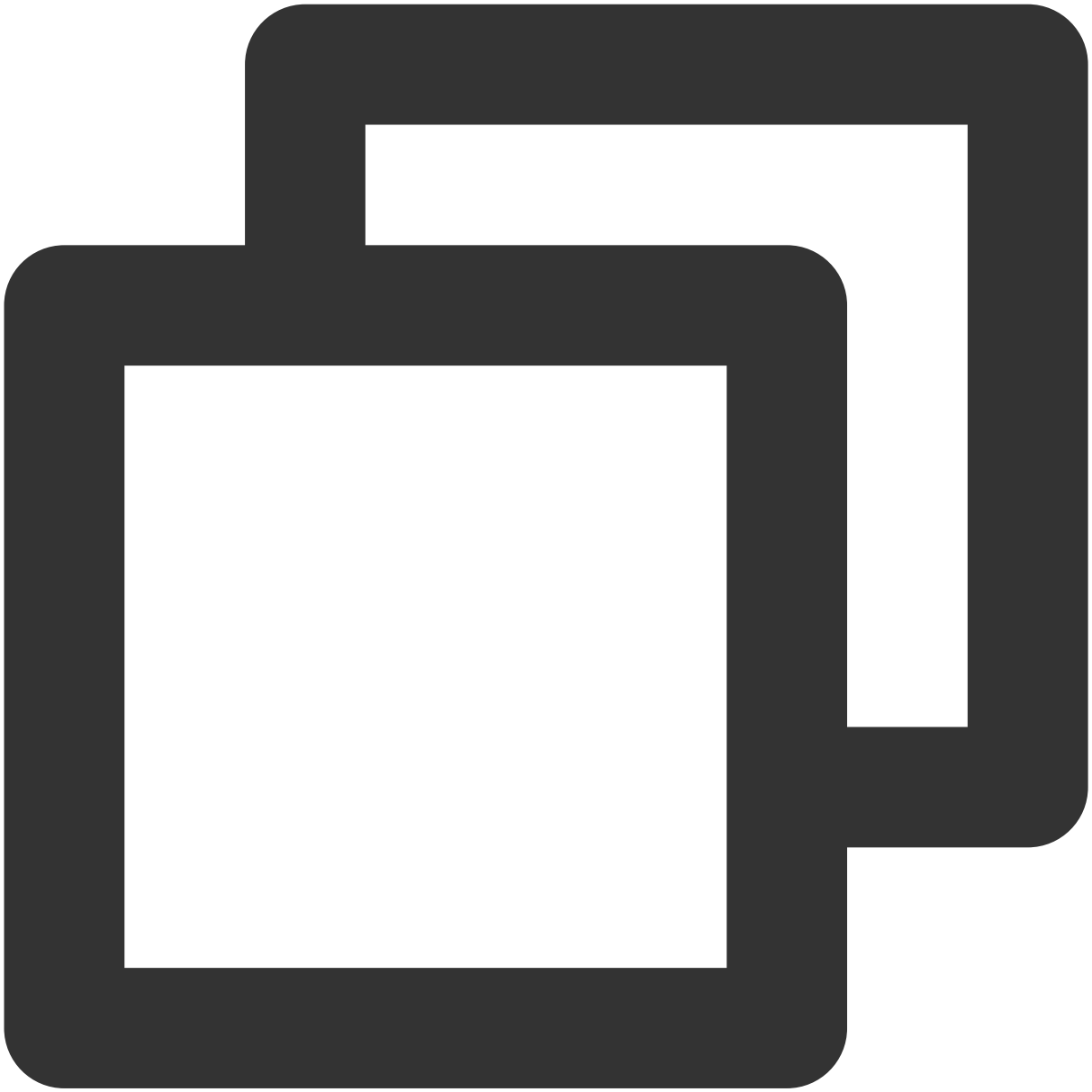
```
                (
                    [BucketId] => examplebucket-1250000000
                    [QueueId] => pcc3ae89sa9d807fs89dg789sdg
                    [Name] => queue-1
                    [State] => Active
                    [MaxSize] => 10000
                    [MaxConcurrent] => 10
                    [Category] => Transcoding
                    [UpdateTime] => 2022-07-13T10:41:34+0800
                    [CreateTime] => 2022-07-13T10:41:34+0800
                    [NotifyConfig] => Array
                        (
                            [Url] =>
                            [State] => Off
                            [Type] =>
                            [Event] =>
                            [ResultFormat] => XML
                        )

                )

        )

    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/queue
)
```

更新媒体处理队列

功能说明

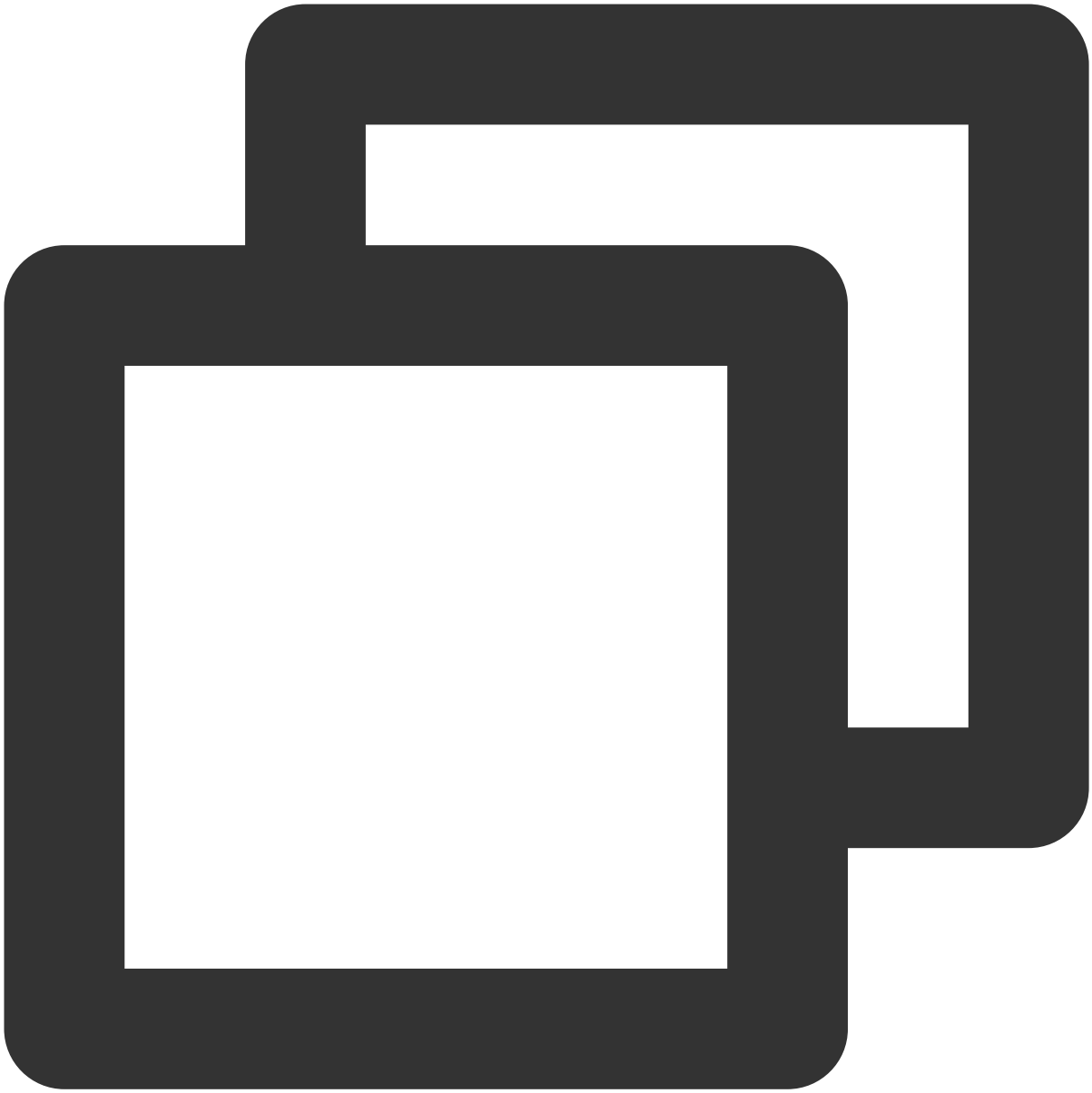
更新媒体处理队列。

方法原型



```
public Guzzle\\Service\\Resource\\Model updateMediaQueue(array $args = array());
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
    $result = $cosClient->updateMediaQueue(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'xxx', // queueId
        'Name' => '', // 模板名称，长度限制100字符
        'State' => 'Active', // 管道状态
        'NotifyConfig' => array(
            'State' => 'Off',
//              'Event' => '',
//              'ResultFormat' => '',
//              'Type' => '',
//              'Url' => '',
//              'MqMode' => '',
//              'MqRegion' => '',
//              'MqName' => '',
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

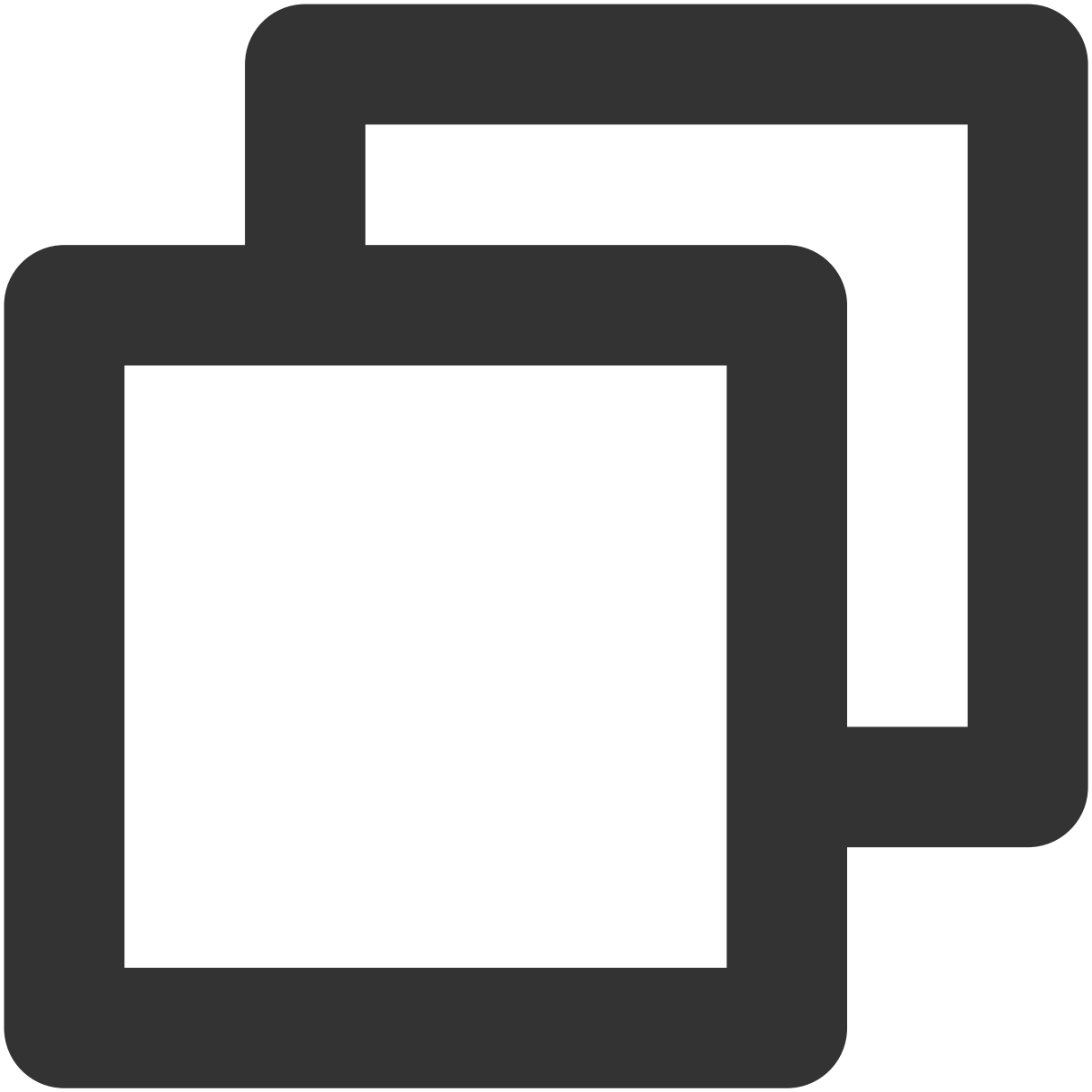
Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Name	Request	队列名称，长度不超过128	String	是	无	无
State	Request	Active 表示队列内的作业会被媒体处理服务调度执行 Paused 表示队列暂停，作业不再会被媒体处理调度执行，队列内的所有作业状态维持在暂停状态，已经执行中的任务不受影响	String	是	无	无
NotifyConfig	Request	回调配置	Container	是	无	无

Container 节点 NotifyConfig 的内容：

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
State	Request.NotifyConfig	回调开关，Off/On	String	否	Off	On/Off
Event	Request.NotifyConfig	回调事件	String	当 State=On 时，必选	无	任务完成：TaskFinish；工作流完成：WorkflowFinish
ResultFormat	Request.NotifyConfig	回调格式	String	否	XML	JSON/XML
Type	Request.NotifyConfig	回调类型	String	当 State=On 时，必选	无	Url 或 TDMQ
Url	Request.NotifyConfig	回调地址	String	当 State=On，且 Type=Url 时，必选	无	不能为内网地址
MqMode	Request.NotifyConfig	TDMQ 使用模式	String	当 State=On，且 Type=TDMQ 时，必选	Queue	主题订阅：Topic 队列服务：Queue
MqRegion	Request.NotifyConfig	TDMQ 所属园区	String	当 State=On，且 Type=TDMQ 时，必选	无	目前支持园区 sh（上海）、bj（北京）、gz（广州）、cd（成都）、hk（中国香港）
MqName	Request.NotifyConfig	TDMQ 主题名称	String	当 State=On，且 Type=TDMQ 时，必选	无	无

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 668
    [Key] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/queue/fcd32
    [Response] => Array
        (
            [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
            [Queue] => Array
                (
                    [QueueId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
                    [Name] => media-queue-1
                    [State] => Active
                    [NotifyConfig] => Array
                        (
                            [Url] =>
                            [Event] =>
                            [Type] =>
                            [State] => Off
                            [ResultFormat] => XML
                            [MqMode] =>
                            [MqName] =>
                            [MqRegion] =>
                        )

                    [MaxSize] => 10000
                    [MaxConcurrent] => 10
                    [CreateTime] => 2022-07-13T10:41:34+0800
                    [UpdateTime] => 2023-01-31T17:32:18+0800
                    [BucketId] => examplebucket-1250000000
                    [Category] => Transcoding
                )

        )

)
```

文档处理
查询文档预览开通状态

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于查询文档预览开通状态的 API 概览和 SDK 示例代码。

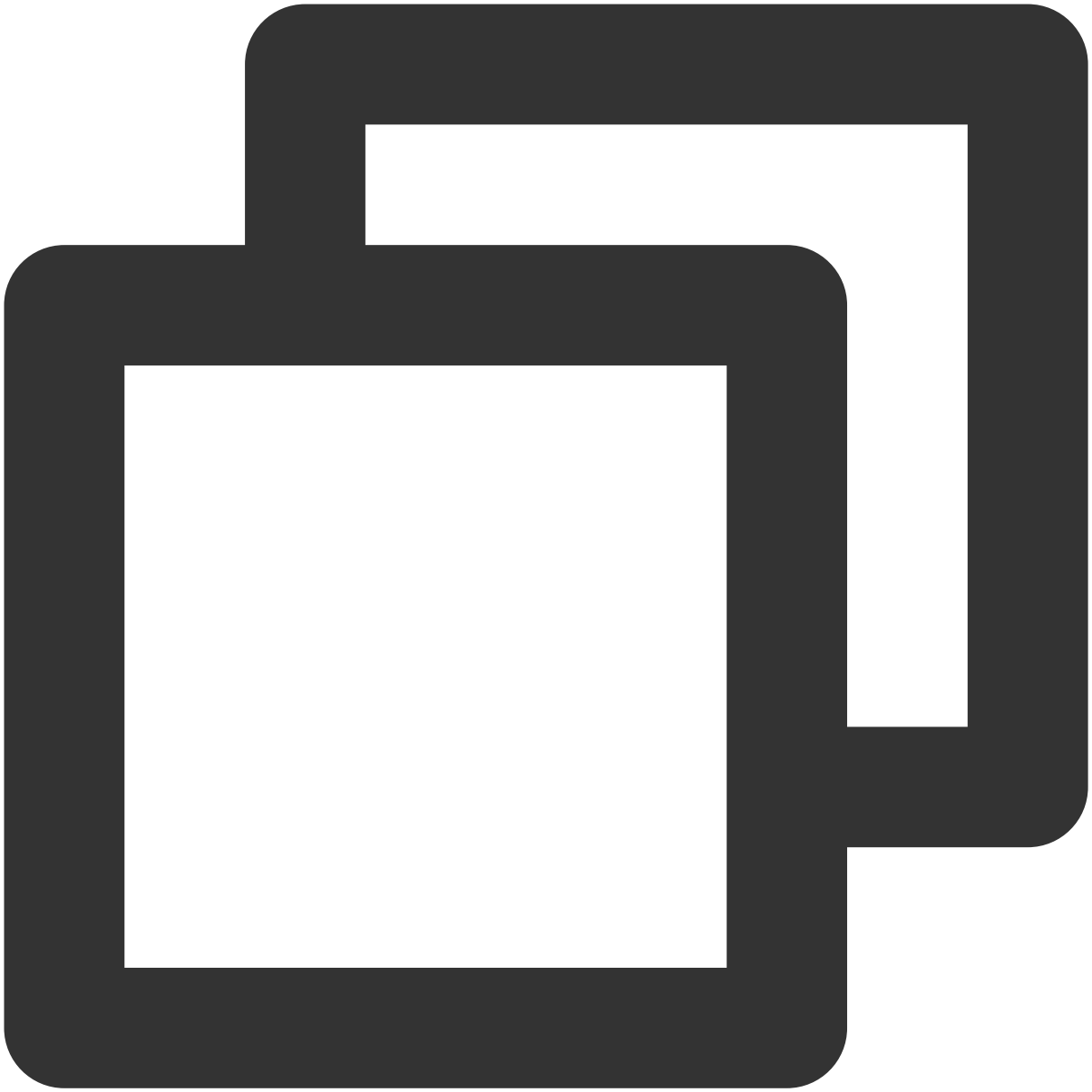
API	操作描述
查询文档预览开通状态	用于查询已经开通文档预览功能的 Bucket

查询文档预览开通状态

功能说明

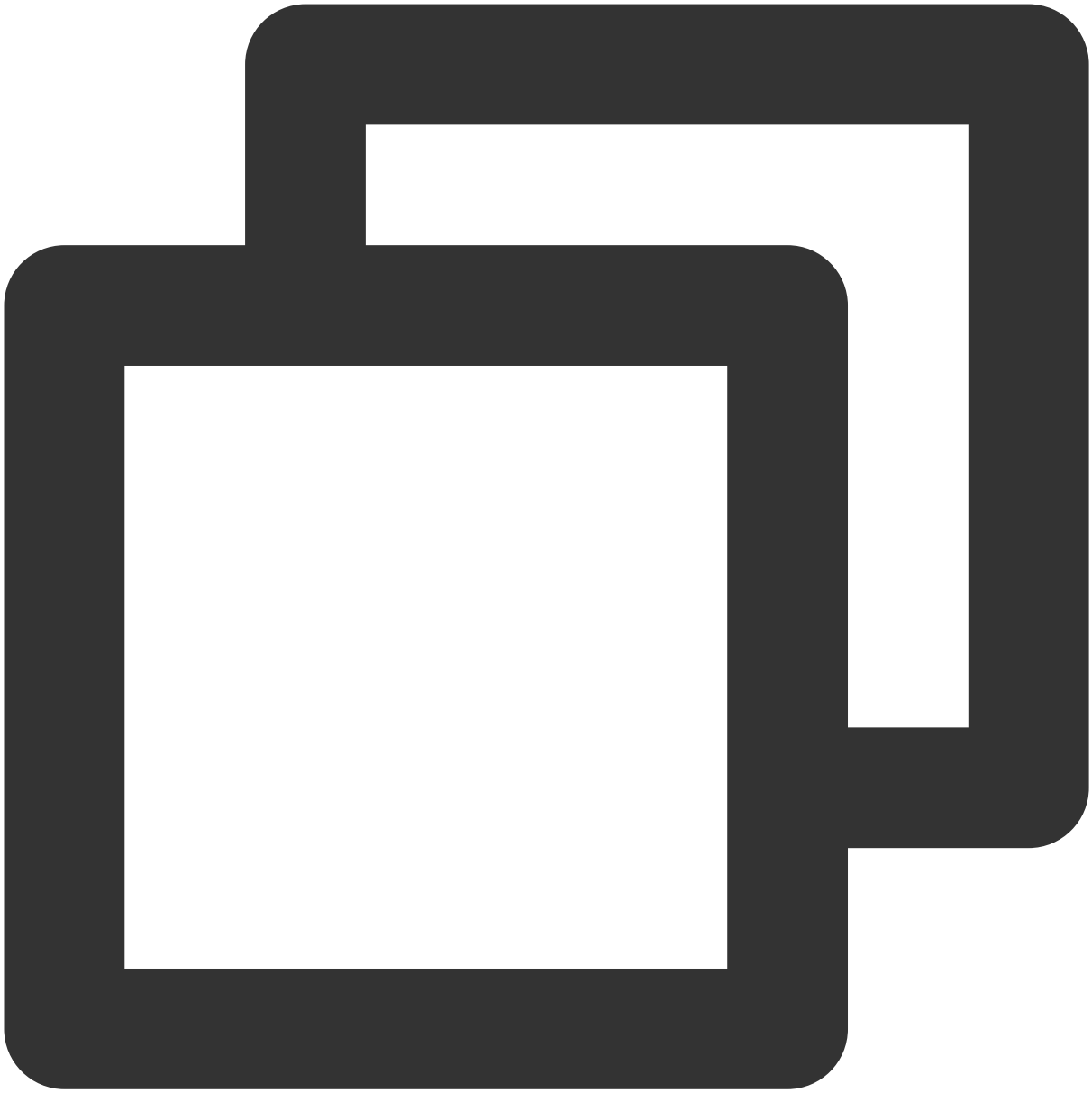
用于查询已经开通文档预览功能的 Bucket。

方法原型



```
public Guzzle\\Service\\Resource\\Model describeDocProcessBuckets(array $args = arr
```

请求示例



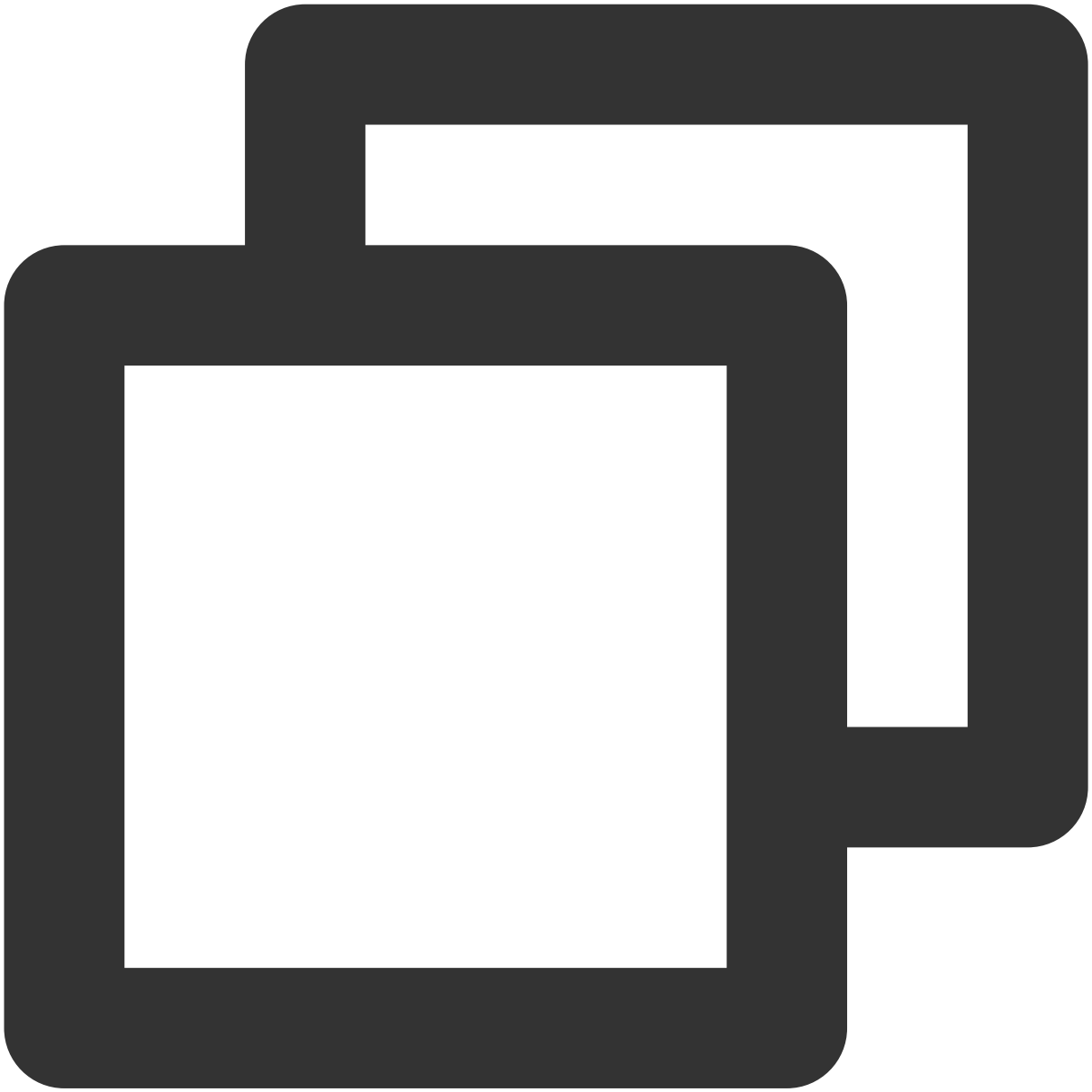
```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
    // 查询文档预览开通状态 https://cloud.tencent.com/document/product/460/46945
    $result = $cosClient->describeDocProcessBuckets(array(
        'Regions' => '', // 可选 地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“
        'BucketNames' => '', // 可选 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
        'BucketName' => '', // 可选 存储桶名称前缀，前缀搜索
        'PageNumber' => 1, // 可选 第几页
        'PageSize' => 20, // 可选 每页个数
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

名称	描述	类型	是否必选
Regions	地域信息，以“,”分隔字符串，支持 All、ap-shanghai、ap-beijing	string	否
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string	否
BucketName	存储桶名称前缀，前缀搜索	string	否
PageNumber	第几页	string	否
PageSize	每页个数	string	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjMxMDcwZDlfAHDKAJHSDKJASHDKY3
    [ContentType] => application/xml
    [ContentLength] => 4600
    [TotalCount] => 59
    [PageNumber] => 1
    [PageSize] => 3
    [DocBucketList] => Array
        (
            [0] => Array
```

```
                (
                    [BucketId] => xxx-123
                    [Name] => xxx-123
                    [Region] => ap-chongqing
                    [CreateTime] => 2022-08-30T10:41:31+0800
                    [AliasBucketId] =>
                )

            [1] => Array
                (
                    [BucketId] => xxx-123
                    [Name] => xxx-123
                    [Region] => ap-chongqing
                    [CreateTime] => 2022-08-30T10:41:31+0800
                    [AliasBucketId] =>
                )

            [2] => Array
                (
                    [BucketId] => xxx-123
                    [Name] => xxx-123
                    [Region] => ap-chongqing
                    [CreateTime] => 2022-08-30T10:41:31+0800
                    [AliasBucketId] =>
                )

        )

    [Location] => ci.ap-guangzhou.myqcloud.com/docbucket
)
```

文档转码
同步请求接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于文档预览的相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
文档转码	文档转码同步请求	获取文档转码的请求 URL

文档转码同步请求

功能说明

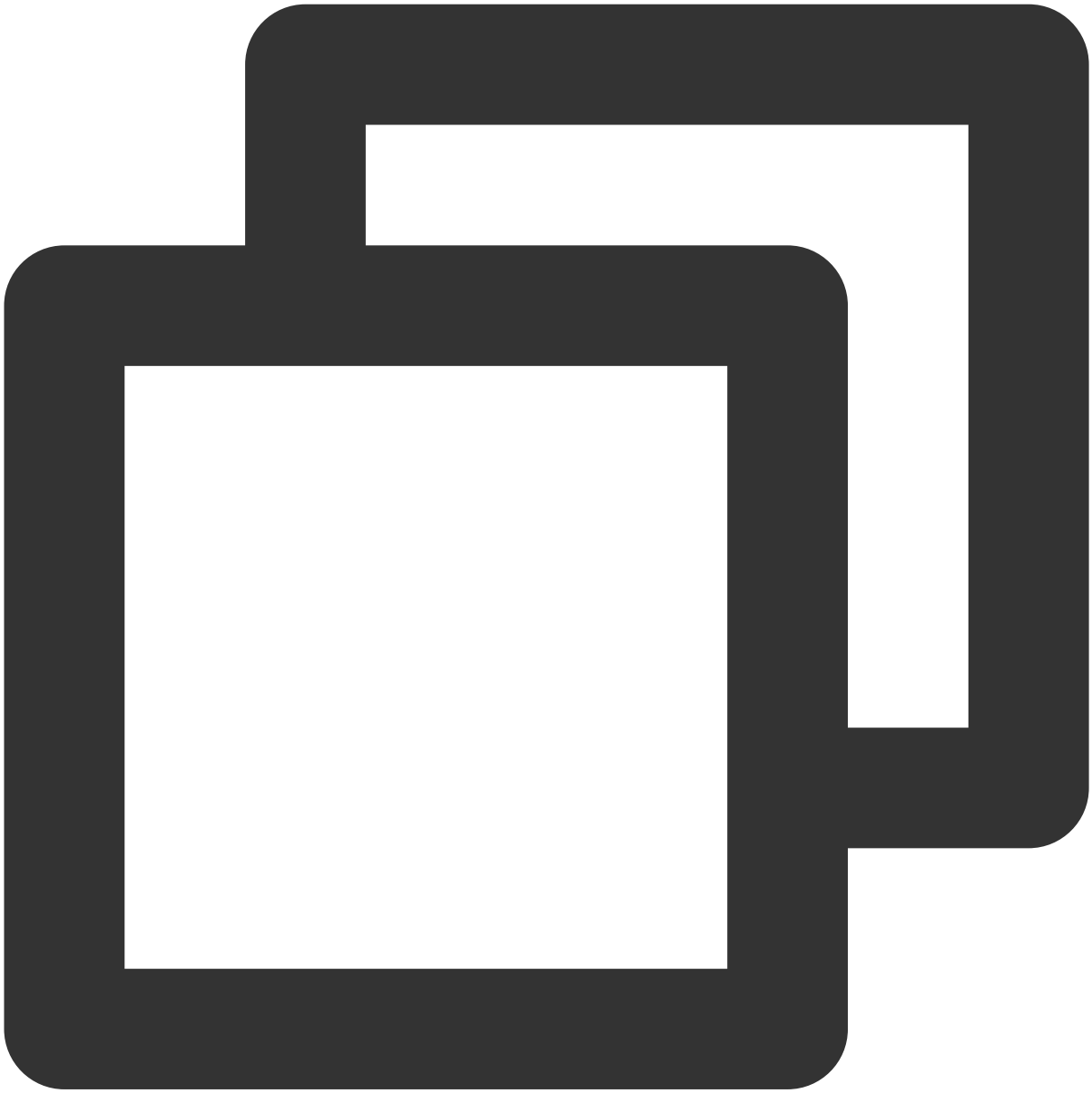
文档转码同步请求，获取文档转码的请求 URL。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的文档处理开关，否则会报错 `doc bucket unbinded, bucket's host is unavailable` 。

如需使用文档转 HTML 功能，可直接将请求参数置于 URL 后方，SDK 仅负责签名计算操作。

示例代码



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 1. 文档转码 https://cloud.tencent.com/document/product/460/47074
    $bucket = 'examplebucket-1250000000';
    $key = 'exampleobject';
    $url = $cosClient->getObjectUrl($bucket, $key);
    $params = array(
        'ci-process' => 'doc-preview',
//        'srcType' => '',
        'page' => 3,
        'dstType' => 'png',
//        'password' => '',
//        'comment' => '',
//        'sheet' => '',
//        'excelPaperDirection' => '',
//        'excelRow' => '',
//        'excelCol' => '',
//        'excelPaperSize' => '',
//        'txtPagination' => '',
        'ImageParams' => 'imageMogr2/thumbnail/!50p',
//        'quality' => '',
//        'scale' => '',
//        'imageDpi' => '',
    );
    $query = http_build_query($params);
    echo $url . $query; // 生成的可访问链接
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

通用请求参数

名称	描述	类型	是否必选
Key	对象文件名，例如 folder/document.pdf	String	是
ci-process	数据万象处理能力，文档预览固定为 doc-preview	String	是

srcType	源数据的后缀类型，当前文档转换根据 COS 对象的后缀名来确定源数据类型。当 COS 对象没有后缀名时，可以设置该值	String	否
page	需转换的文档页码，默认从1开始计数；表格文件中 page 表示转换的第 X 个 sheet 的第 X 张图	Int	否
dstType	转换输出目标文件类型： png，转成 png 格式的图片文件；jpg，转成 jpg 格式的图片文件； pdf，转成 pdf 格式文件；无法选择页码，page 参数不生效；txt，转成 txt 格式文件。如果传入的格式未能识别，默认使用 jpg 格式	String	否
password	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String	否
comment	是否隐藏批注和应用修订，默认为0。0：隐藏批注，应用修订 1：显示批注和修订	Int	否

适用于表格文件（Excel）的参数

名称	描述	类型	是否必选
sheet	表格文件参数，转换第 X 个表，默认为1	Int	否
excelPaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	Int	否
excelRow	值为 `1` 表示将所有列放到1页进行排版，默认值为0	Int	否
excelCol	值为 `1` 表示将所有行放到1页进行排版，默认值为0	Int	否
excelPaperSize	设置纸张（画布）大小，对应信息为： `0 → A4` 、 `1 → A2` 、 `2 → A0` ，默认 A4 纸张 （需配合 `excelRow` 或 `excelCol` 一起使用）	Int	否

适用于转码成 txt 的参数

名称	描述	类型	是否必选
txtPagination	是否转换成长文本，设置为 `true` 时，可以将需要导出的页中的文字合并导出，分页范围可以通过 Ranges 控制。默认值为 `false` ，按页导出 txt。（ `ExportType="txt"` 时生效）	Bool	否

适用于转码成 png/jpg 图片的参数

名称	描述	类型	是否必选
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String	否
quality	生成预览图的图片质量，取值范围为 [1, 100]，默认值100。 例如取值为100，代表生成图片质量为100%	Int	否
scale	预览图片的缩放参数，取值范围为 [10, 200]， 默认值100。 例如取值为200，代表图片缩放比例为200% 即放大两倍	Int	否
imageDpi	按指定 dpi 渲染图片，该参数与 `scale` 共同作用，取值范围 `96-600` ，默认值为 `96` 。转码后的图片单边宽度需小于65500像素	Int	否

注意：

目前支持的输入文件类型包含如下格式：

演示文件：pptx、ppt、pot、potx、pps、ppsx、dps、dpt、pptm、potm、ppsm。

文字文件：doc、dot、wps、wpt、docx、dotx、docm、dotm。

表格文件：xls、xlt、et、ett、xlsx、xltx、csv、xlsb、xlsm、xltm、ets。

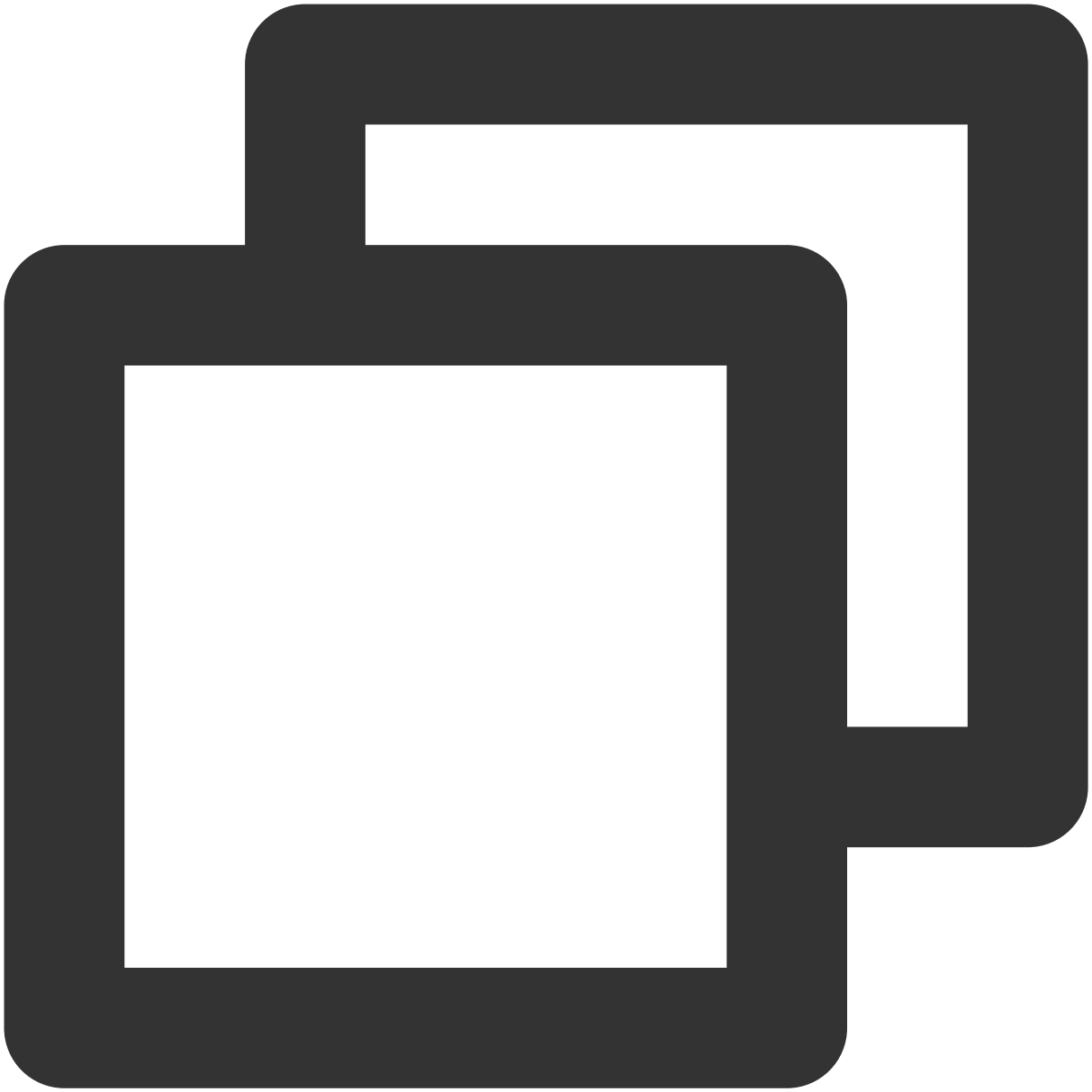
表格文件，一张表可能分割为多页转换，生成多张图片。

其他格式文件：pdf、lrc、c、cpp、h、asm、s、java、asp、bat、bas、prg、cmd、rtf、txt、log、xml、htm、html。

输入文件大小限制在200MB之内。

输入文件页数限制在5000页之内。

返回结果示例



同步请求URL

```
https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/example.ppt?sign=q-s
```

异步处理任务接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于异步处理任务接口的相关的 API 概览以及 SDK 示例代码。

注意：

需要 COS PHP SDK v2.4.0 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

API	操作名	操作描述
CreateDocProcessJobs	提交文档预览任务	用于提交一个文档预览任务
DescribeDocProcessJob	查询文档预览任务	查询指定的文档预览任务
DescribeDocProcessJobs	拉取文档预览任务	拉取符合条件的文档预览任务

提交文档预览任务

功能说明

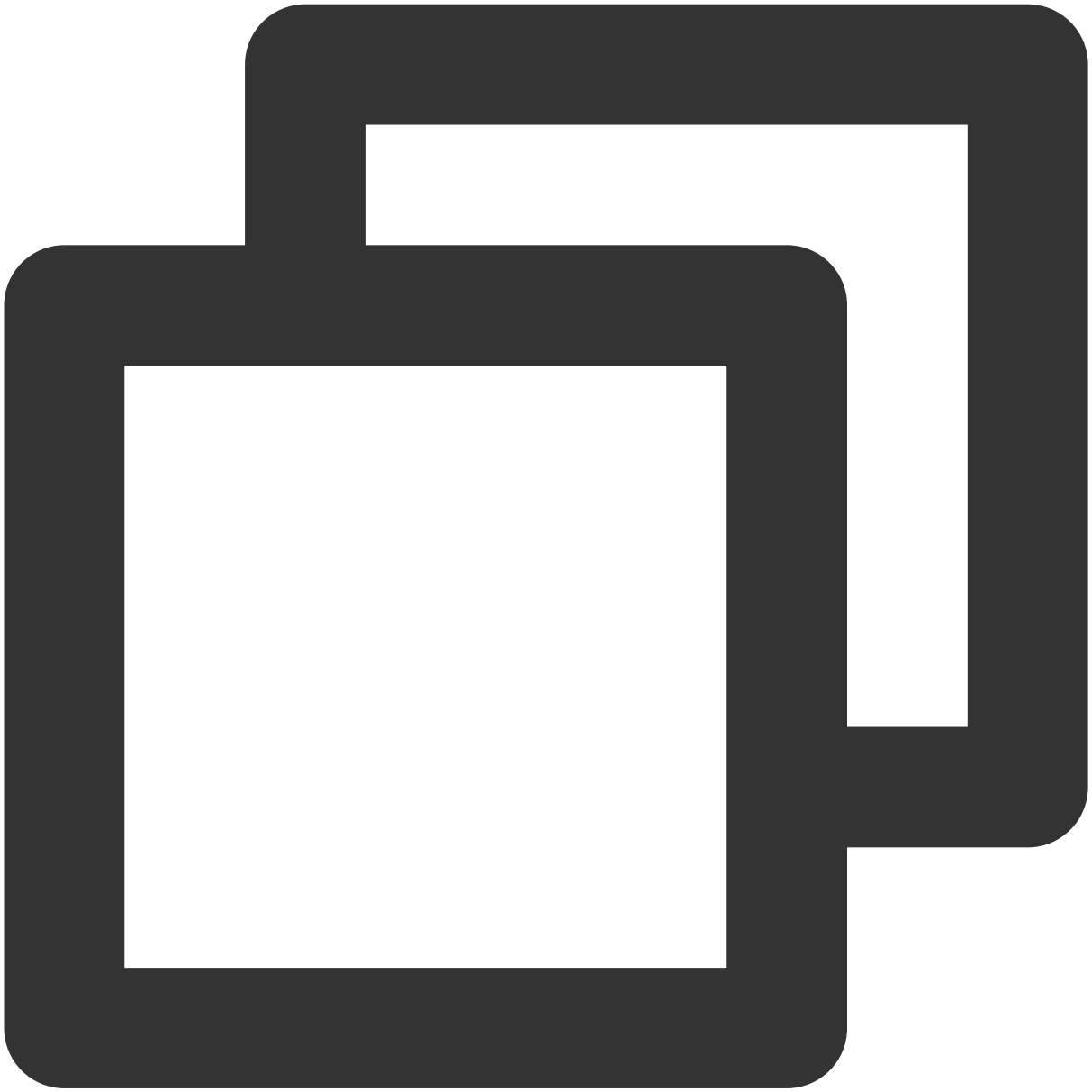
CreateDocProcessJobs 接口用于提交一个文档预览任务。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的文档处理开关，否则会报错 `doc bucket unbinded, bucket's host is unavailable` 。

如需使用文档转 HTML 功能，可直接将请求参数置于 URL 后方，SDK 仅负责签名计算操作。

示例代码



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createDocProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Tag' => 'DocProcess', //任务的 Tag：DocProcess 固定值
        'QueueId' => 'pd8e422a2ea134165a92f2012ea43****', //任务所在的队列 ID
        'Input' => array(
            'Object' => 'Append功能测试.pdf' //待操作的文件对象
        ),
        'Operation' => array(
            'DocProcess' => array(
                'SrcType' => 'pdf', //源数据的后缀类型
                'TgtType' => 'png', //转换输出目标文件类型
                'SheetId' => 0, //表格文件参数，转换第 X 个表，默认为1
                'StartPage' => 1, //从第 X 页开始转换，默认为1
                'EndPage' => 3, //转换至第 X 页，默认为-1，即转换全部页
                'ImageParams' => '', //转换后的图片处理参数
                'DocPassword' => '', //Office 文档的打开密码
                'Comments' => 0, //是否隐藏批注和应用修订，默认为 0
                'PaperDirection' => 0, //表格文件转换纸张方向，默认为0
                'Quality' => 100, //生成预览图的图片质量，取值范围 [1-100]，默认值100
                'Zoom' => 100, //预览图片的缩放参数，取值范围[10-200]， 默认值100
            ),
            'Output' => array(
                'Region' => $region, //存储桶的地域
                'Bucket' => 'examplebucket-1250000000', //存储结果的存储桶
                'Object' => 'pic-${Page}.jpg', //输出文件路径
            ),
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是

Operation	Array	操作规则	是
Tag	String	创建任务的 Tag，目前仅支持：DocProcess	是
Input	Array	待操作的文件对象	是
QueueId	String	任务所在的队列 ID	是

Input 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Object	String	文件在 COS 上的文件路径，Bucket 由 Host 指定	是

Operation 中的具体数据描述如下：

参数名称	类型	描述	是否必填
DocProcess	Array	当 Tag 为 DocProcess 时有效，指定该任务的参数	是
Output	Array	结果输出地址	是

DocProcess 中的具体数据描述如下：

参数名称	类型	描述	是否必填
srcType	String	源数据的后缀类型，当前文档转换根据 cos 对象的后缀名来确定源数据类型，当 cos 对象没有后缀名时，可以设置该值	否
tgtType	String	转换输出目标文件类型： jpg，转成 jpg 格式的图片文件；如果传入的格式未能识别，默认使用 jpg 格式 png，转成 png 格式的图片文件 pdf，转成 pdf 格式文件（暂不支持指定页数）	否
sheetId	Integer	表格文件参数，转换第 X 个表，默认为1；设置 SheetId 为0，即转换文档中全部表	否
startPage	Integer	从第 X 页开始转换；在表格文件中，一张表可能分割为多页转换，生成多张图片。StartPage 表示从指定 SheetId 的第 X 页开始转换。默认为1	否
endPage	Integer	转换至第 X 页；在表格文件中，一张表可能分割为多页转换，生成多张图片。EndPage 表示转换至指定 SheetId 的第 X 页。默认为-1，即转换全部页	否
imageParams	String	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数	否

版权所有：腾讯云计算（北京）有限责任公司

		可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	
docPassword	String	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	否
comments	Integer	是否隐藏批注和应用修订，默认为 0；0：隐藏批注，应用修订；1：显示批注和修订	否
paperDirection	Integer	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	否
quality	Integer	生成预览图的图片质量，取值范围 [1-100]，默认值100。 例：值为100，代表生成图片质量为100%	否
zoom	Integer	预览图片的缩放参数，取值范围[10-200]， 默认值100。 例：值为200，代表图片缩放比例为200% 即放大两倍	否

Output 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Region	String	存储桶的地域	是
Bucket	String	存储结果的存储桶	是
Object	String	输出文件路径	是

返回参数说明

参数名称	类型	描述
Code	String	错误码，只有 State 为 Failed 时有意义
Message	String	错误描述，只有 State 为 Failed 时有意义
JobId	String	新创建任务的 ID
Tag	String	新创建任务的 Tag：DocProcess
State	String	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个
CreationTime	String	任务的创建时间
QueueId	String	任务所属的队列 ID
Input	Array	该任务的输入文件路径
Operation	Array	该任务的规则

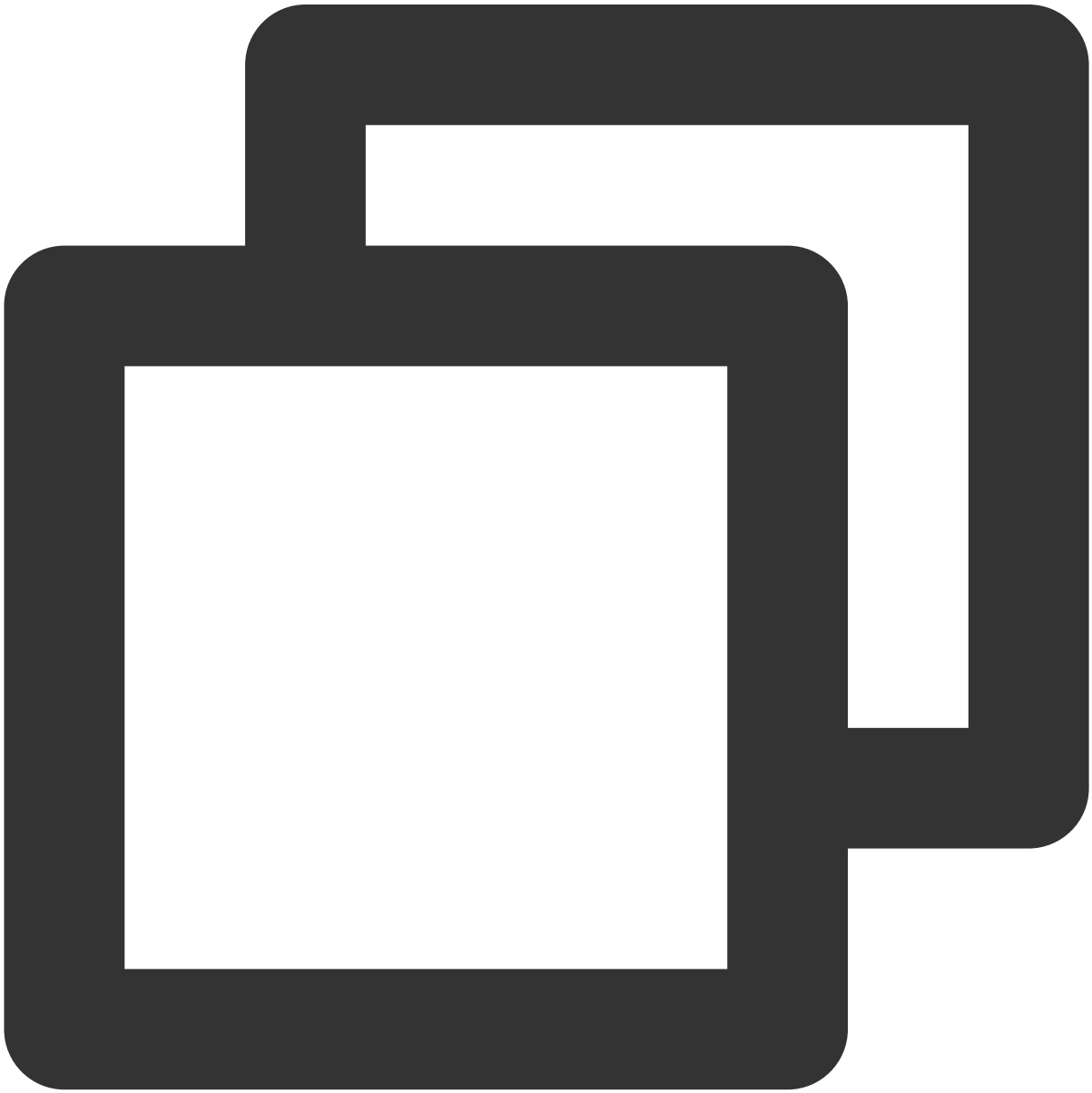
版权所有：腾讯云计算（北京）有限责任公司

查询文档预览任务

功能说明

DescribeDocProcessJob 用于查询指定的文档预览任务。

示例代码



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';
```

```
$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->describeDocProcessJob(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'examplejobid', // JobId
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述
Bucket	COS 存储桶名称	String
Key	创建任务的 ID	String

返回参数说明

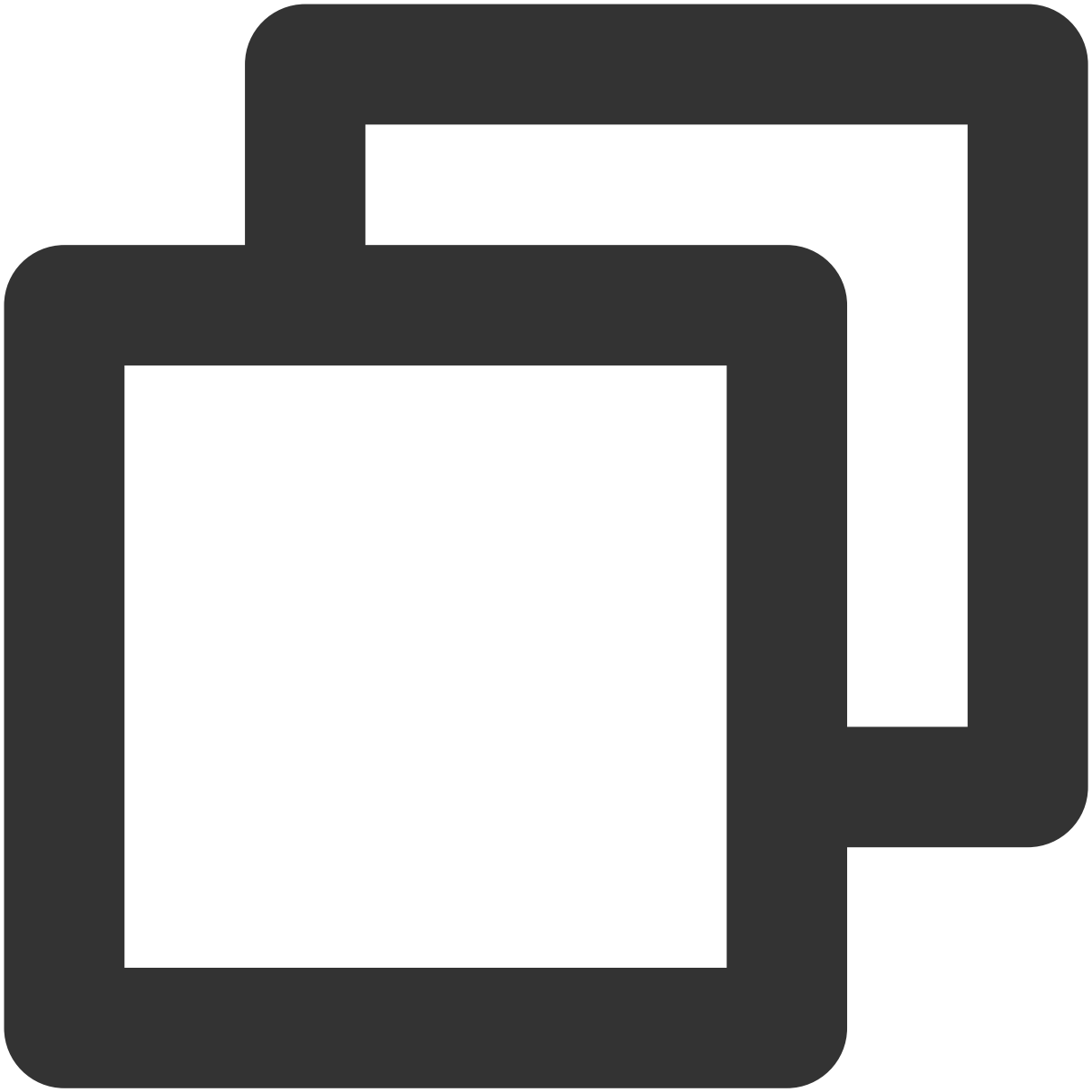
参数名称	类型	描述
jobsDetail	记录任务的详细信息	Array

拉取文档预览任务

功能说明

DescribeDocProcessJobs 用于拉取符合条件的文档预览任务。

示例代码



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->getDescribeDocProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'QueueId' => 'pd8e422a2ea134165a92f2012ea43****', //拉取该队列 ID 下的任务
        'Tag' => 'DocProcess', //任务的 Tag：DocProcess 固定值
//      'NextToken' => '143486', //请求的上下文，用于翻页。上次返回的值
//      'OrderByTime' => 'Desc', //Desc 或者 Asc。默认为 Desc
//      'Size' => 2, //拉取的最大任务数。默认为10。最大为100
//      'States' => 'All', //拉取该状态的任务，以,分割，支持多状态：All、Submitted、Runnin
//      'StartCreationTime' => '2021-10-10T16:20:07+0800', //拉取创建时间大于该时间的任
//      'EndCreationTime' => '2021-10-10T16:20:07+0800', //拉取创建时间小于该时间的任务
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	COS 存储桶名称	是
QueueId	String	拉取该队列 ID 下的任务	是
Tag	String	任务的 Tag：DocProcess	是
OrderByTime	String	Desc 或者 Asc，默认为 Desc	否
NextToken	String	请求的上下文，用于翻页，设置为上次返回的值	否
Size	Integer	拉取的最大任务数，默认为10，最大为100	否
States	String	拉取该状态的任务，以 `,` 分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel，默认为 All	否
StartCreationTime	String	拉取创建时间大于该时间的任务，格式为：`%Y-%m-%dT%H:%m:%S%z`	否

EndCreationTime	String	拉取创建时间小于该时间的任务，格式为：`%Y-%m-%dT%H:%m:%S%z`	否

返回参数说明

参数名称	类型	描述
JobDetail	Array	记录任务列表
nextToken	String	翻页的上下文 Token

异步处理队列接口

最近更新时间：2024-03-13 10:25:32

简介

本文档提供关于异步处理队列接口相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DescribeDocProcessQueues	查询文档转码队列	用于查询文档转码队列
UpdateDocProcessQueue	更新文档转码队列	用于更新文档转码队列

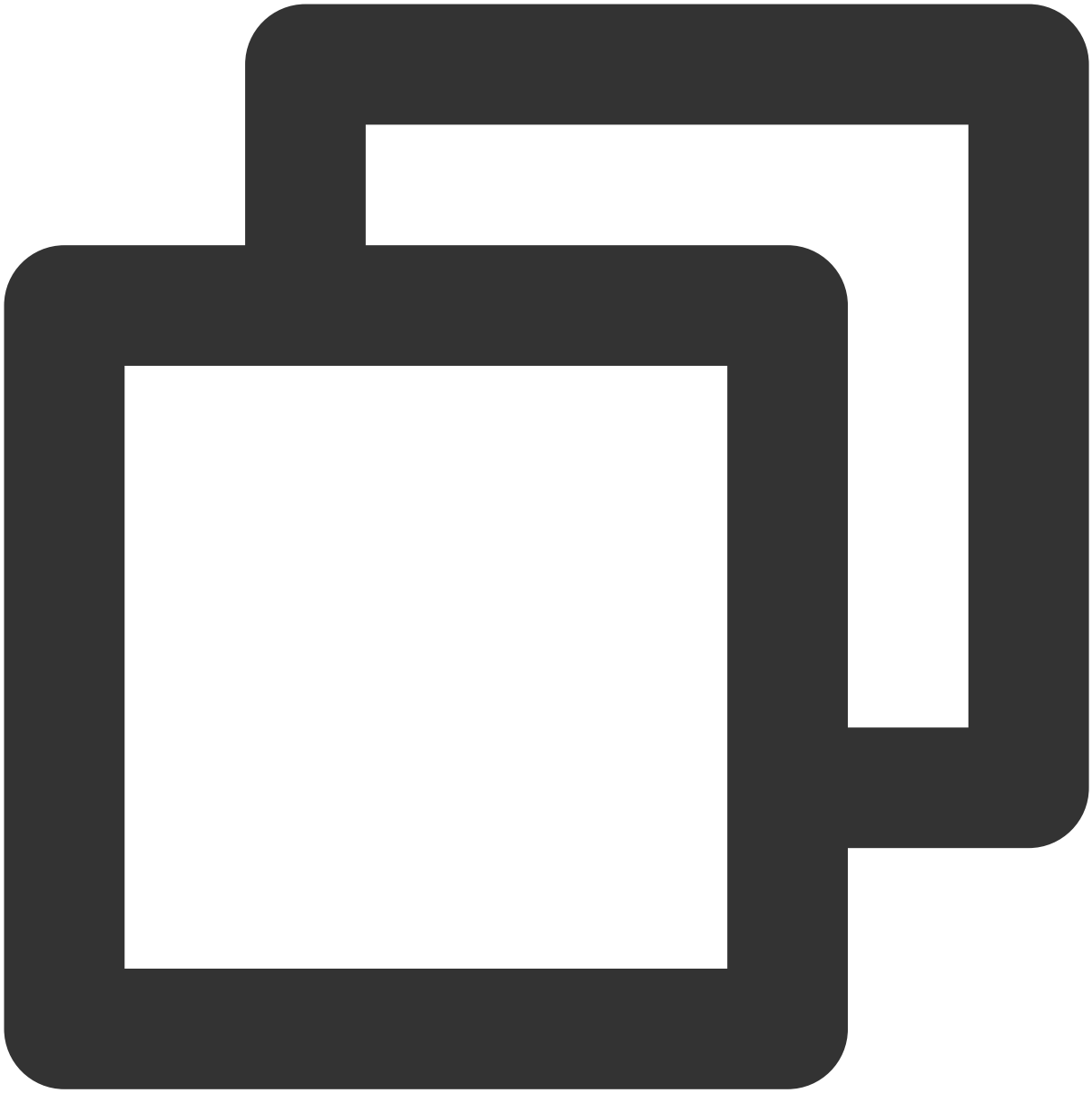
查询文档转码队列

功能说明

DescribeDocProcessQueues 接口用于查询文档转码队列。（可通过该接口获取桶对应的队列 ID）

示例代码

版权所有：腾讯云计算（北京）有限责任公司



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->describeDocProcessQueues(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述
Bucket	COS 存储桶名称	String

返回参数说明

Response 中的具体数据描述如下：

参数名称	类型	描述
RequestId	String	请求的唯一 ID
TotalCount	Integer	队列总数
PageNumber	Integer	当前页数，同请求中的 pageNumber
PageSize	Integer	每页个数，同请求中的 pageSize
QueueList	Array	队列数组

QueueList 中的具体数据描述如下：

参数名称	类型	描述
QueueId	String	队列 ID
Name	String	队列名字
State	String	当前状态，Active 或者 Paused
NotifyConfig	Array	回调配置

MaxSize	Integer	队列最大长度
MaxConcurrent	Integer	当前队列最大并行执行的任务数
UpdateTime	String	更新时间
CreateTime	String	创建时间

NotifyConfig 中的具体数据描述如下：

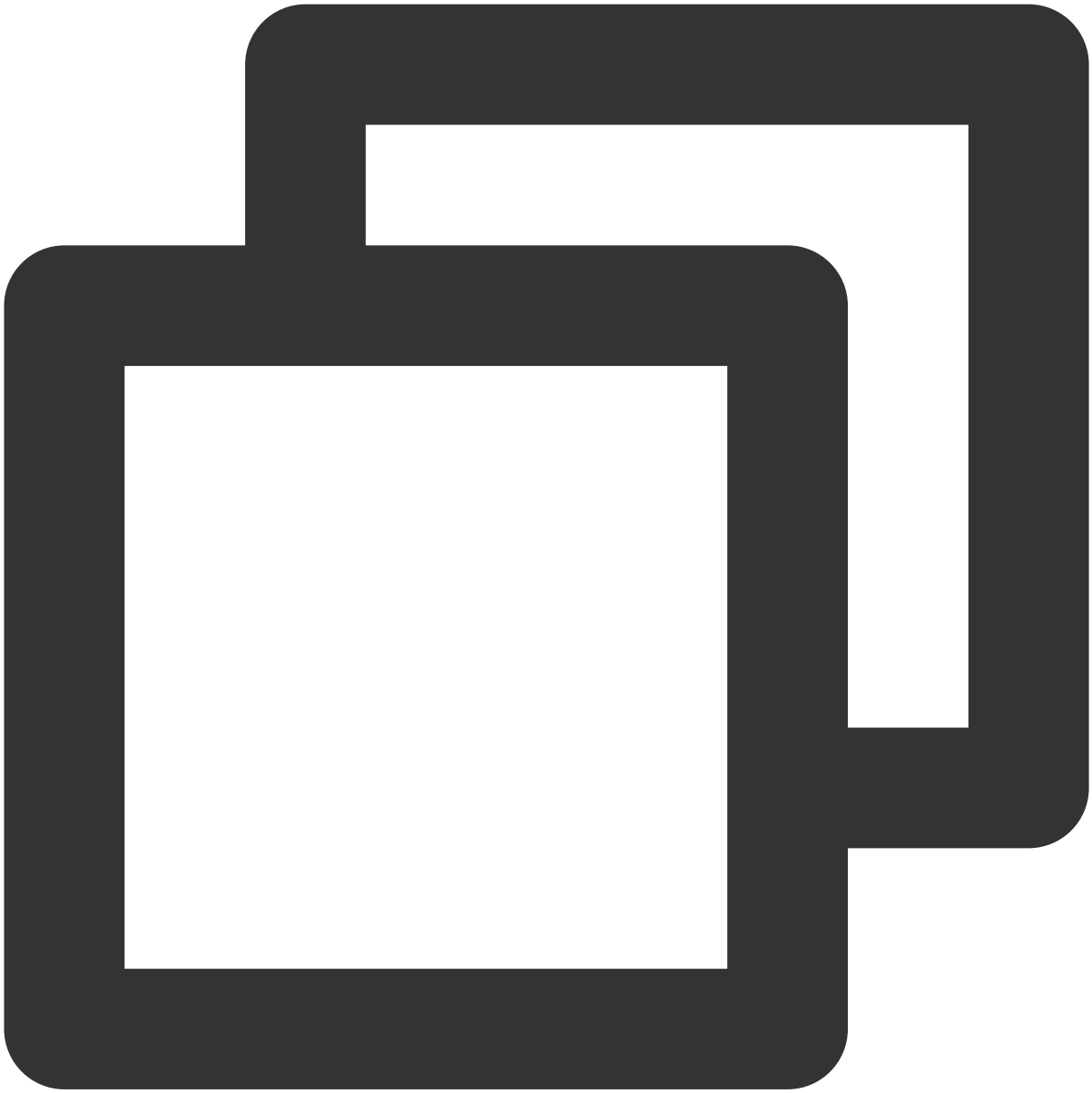
参数名称	类型	描述
Url	String	回调地址
State	String	开关状态，On 或者 Off
Type	String	回调类型，Url
Event	String	触发回调的事件

更新文档转码队列

功能说明

UpdateDocProcessQueue 接口用于更新文档转码队列。

示例代码



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 更新文档转码队列 https://cloud.tencent.com/document/product/460/46947
    $result = $cosClient->updateDocProcessQueue(array(
        'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => '', // QueueID
        'Name' => '',
        'QueueID' => '',
        'State' => '',
        'NotifyConfig' => array(
            'Url' => '',
            'Type' => '',
            'Event' => '',
            'State' => '',
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Name	Request	队列名称	String	是	长度限制100字符
QueueID	Request	队列ID	String	是	无
State	Request	队列状态	String	是	1. Active 表示队列内的作业会被文档预览服务调度转码执行 </br>2. Paused 表示队列暂停，作业不再会被文档预览服务调度执行，队列内的所有作业状态维持

					在已提交状态，已经处于执行中的任务将继续执行，不受影响
NotifyConfig	Request	通知渠道	Container	是	第三方回调 Url

Container 类型 NotifyConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Url	Request.NotifyConfig	回调配置	String	否	长度限制100字符
Type	Request.NotifyConfig	回调类型，普通回调：Url	String	否	长度限制100字符
Event	Request.NotifyConfig	回调事件，文档预览任务完成	String	否	长度限制100字符
State	Request.NotifyConfig	回调开关，Off，On	String	否	长度限制100字符

返回参数说明

Response 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型
RequestId	Response	请求的唯一 ID	String
Queue	Response	队列信息，详情请参见 DescribeDocProcessQueues 中的 Response.QueueList	Container

文档转 HTML

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于文档预览的相关的 API 概览以及 SDK 示例代码。

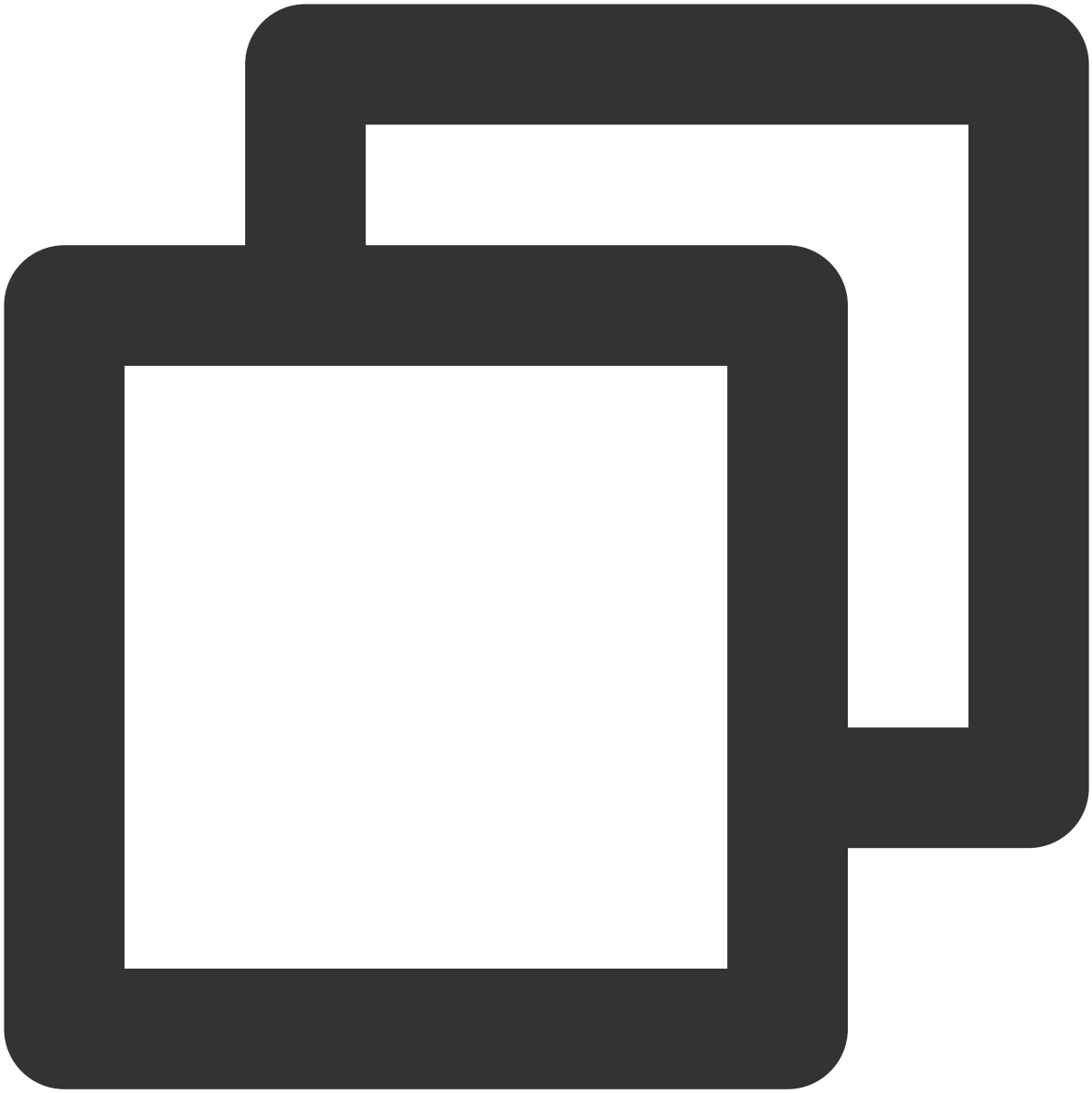
API	操作名	操作描述
文档转 HTML	文档转 HTML 同步请求	获取文档转 HTML 的请求 URL

文档转 HTML

功能说明

文档转 HTML 同步请求，获取文档转 HTML 的请求 URL。

示例代码



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 2. 文档转HTML https://cloud.tencent.com/document/product/460/52518
    $bucket = 'examplebucket-1250000000';
    $key = 'exampleobject';
    $url = $cosClient->getObjectUrl($bucket, $key, "+30 minutes");
    $params = array(
        'ci-process' => 'doc-preview',
//        'srcType' => '',
        'dstType' => 'html',
//        'sign' => '',
//        'copyable' => '',
//        'htmlParams' => '',
//        'htmlwaterword' => '',
//        'htmlfillstyle' => '',
//        'htmlfront' => '',
//        'htmlrotate' => '',
//        'htmlhorizontal' => '',
//        'htmlvertical' => '',
    );
    $query = http_build_query($params);
    echo $url . $query; // 生成的可访问链接
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

名称	描述	类型	是否必选
Key	对象文件名，例如 folder/document.pdf	String	是
ci-process	数据万象处理能力，文档 HTML 预览固定为 doc-preview	String	是
dstType	转换输出目标文件类型，文档 HTML 预览固定为 html（需为小写字母）	String	是
srcType	指定目标文件类型，支持的文件类型请见下方	String	否
sign	对象下载签名，如果预览的对象为私有读时，需要传入签名，详情请参见请求签名 文档</br>注意：需要进行 urlencode	String	否

copyable	是否可复制。默认为可复制，填入值为1；不可复制，填入值为0	String	否
htmlParams	自定义配置参数，JSON 结构，需要经过 URL 安全 的 Base64 编码，默认配置为：{ commonOptions: { isShowTopArea: true, isShowHeader: true } }，支持的配置参考 自定义配置项说明	String	否
htmlwaterword	水印文字，需要经过 URL 安全 的 Base64 编码，默认为空	String	否
htmlfillstyle	水印 RGBA（颜色和透明度），需要经过 URL 安全 的 Base64 编码，默认为：rgba(192,192,192,0.6)	String	否
htmlfront	水印文字样式，需要经过 URL 安全 的 Base64 编码，默认为：bold 20px Serif	String	否
htmlrotate	水印文字旋转角度，0 - 360，默认315度	String	否
htmlhorizontal	水印文字水平间距，单位 px，默认为50	String	否
htmlvertical	水印文字垂直间距，单位 px，默认为100	String	否

注意：

目前支持的输入文件类型包含如下格式：

演示文件：pptx、ppt、pot、potx、pps、ppsx、dps、dpt、pptm、potm、ppsm。

文字文件：doc、dot、wps、wpt、docx、dotx、docm、dotm。

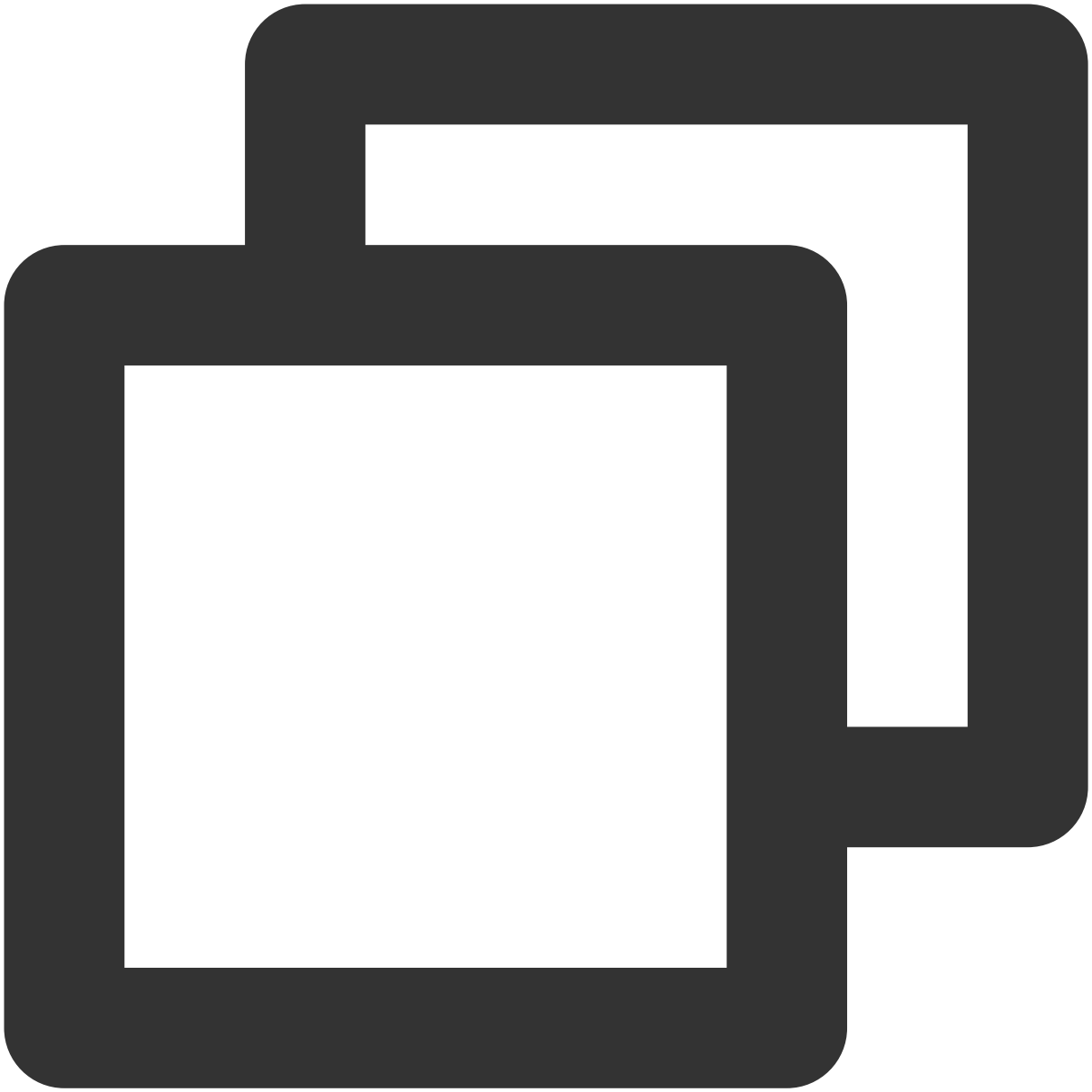
表格文件：xls、xlt、et、ett、xlsx、xltx、csv、xlsb、xlsm、xltm、ets。

其他格式文件：pdf、lrc、c、cpp、h、asm、s、java、asp、bat、bas、prg、cmd、rtf、txt、log、xml、htm、html。

输入文件大小限制在200MB之内。

输入文件页数限制在5000页之内。

返回结果示例



同步请求URL

```
https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/example.ppt?sign=q-s
```

内容识别
图片标签

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于图片标签相关的 API 概览以及 SDK 示例代码。

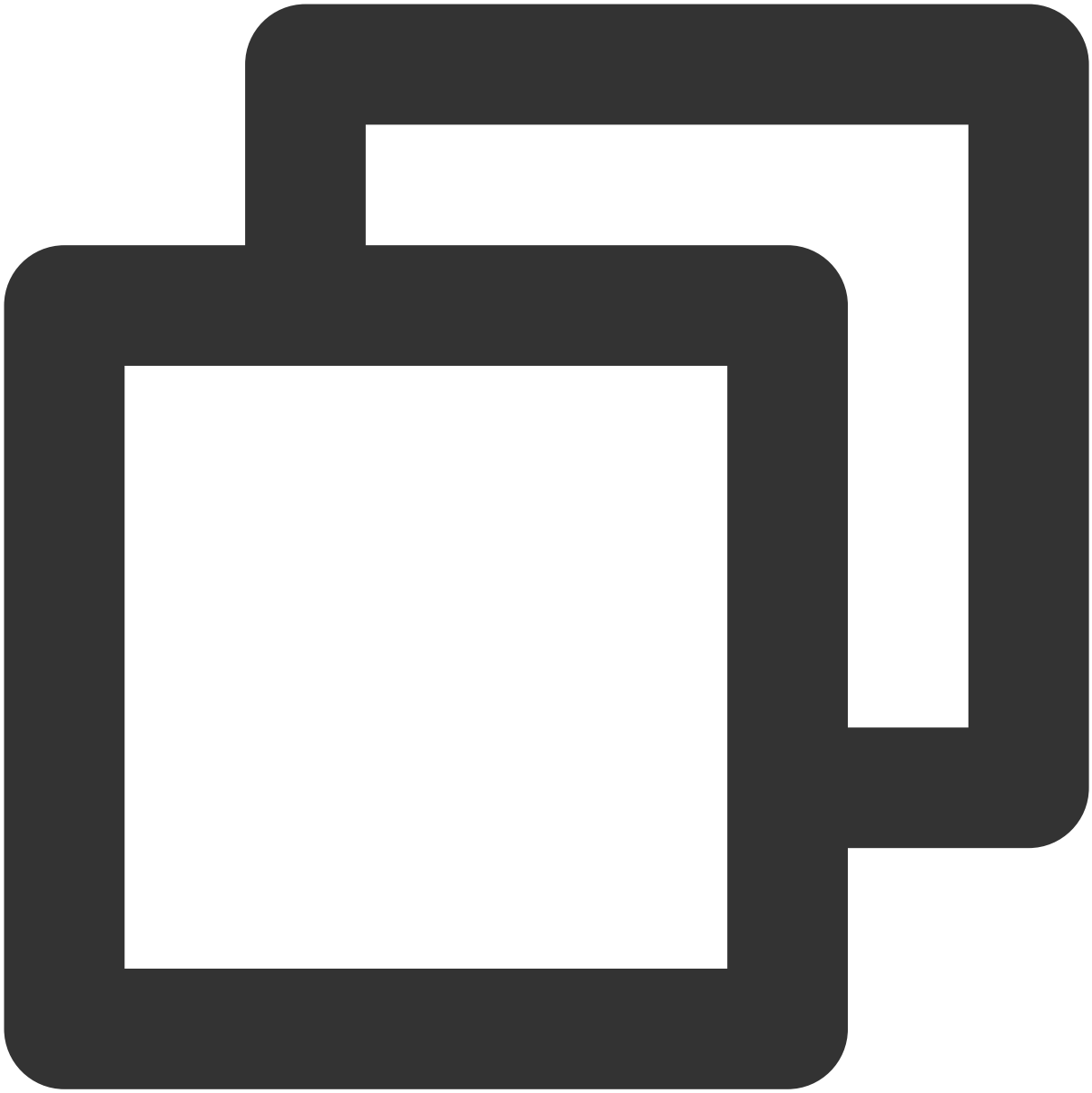
API	说明
图片标签	图片标签功能通过借助持久化处理接口，实现对对象存储（Cloud Object Storage，COS）存量数据的图片标签识别

图片标签

功能说明

图片标签功能通过借助对象存储的持久化处理接口，实现对 COS 存量数据的图片标签识别，返回图片中置信度较高的主题标签。

示例代码



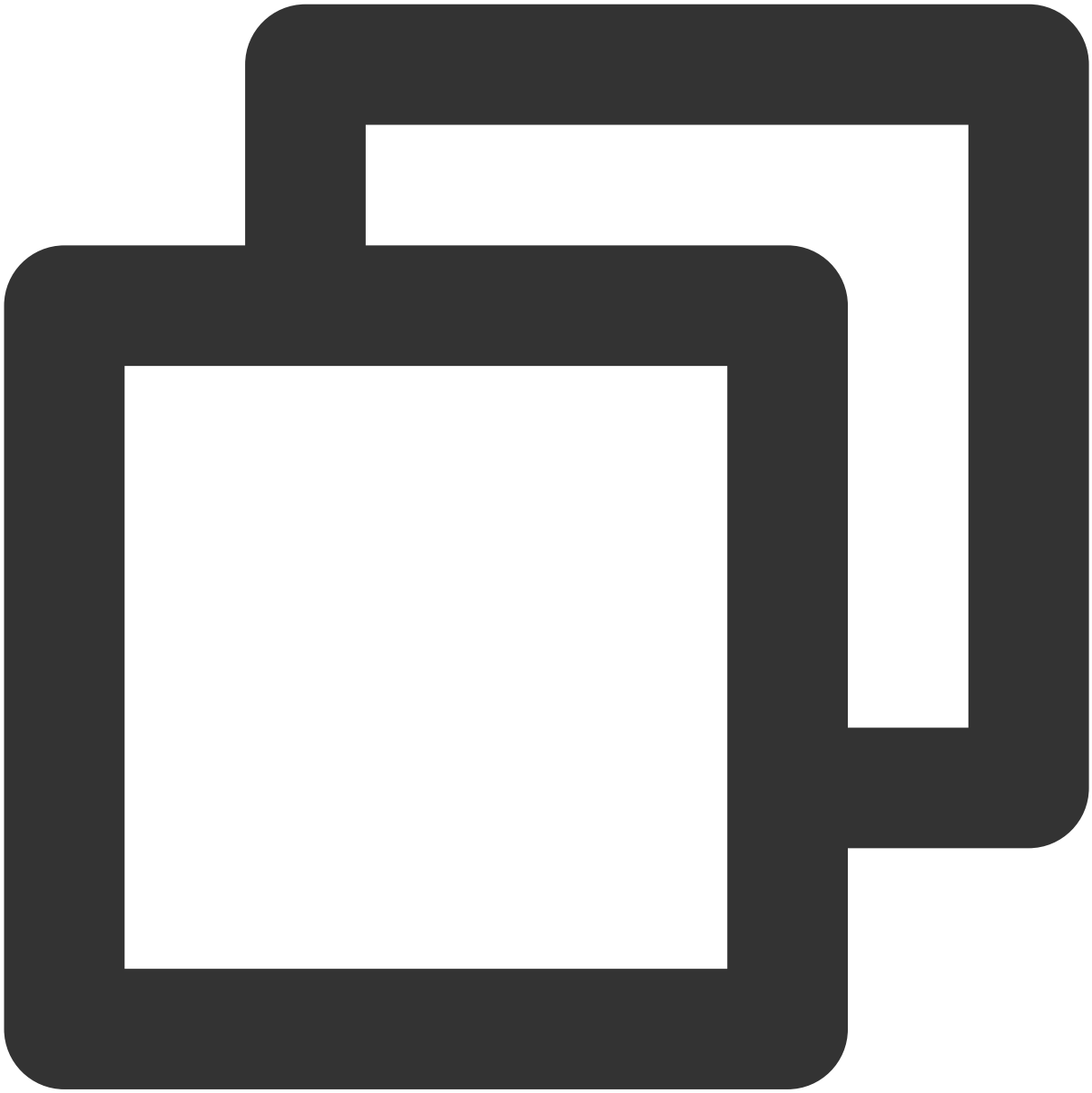
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->DetectLabel(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成,
        'Key' => 'exampleobject',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [Labels] => Array
            (
```

```
                [0] => Array
                (
                    [Confidence] => 83
                    [Name] => 玩具
                )

                [1] => Array
                (
                    [Confidence] => 77
                    [Name] => 毛绒玩具
                )

                [2] => Array
                (
                    [Confidence] => 15
                    [Name] => 艺术
                )
            )
        )
)
```

返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
Labels	Array	标签信息	无

版权所有：腾讯云计算（北京）有限责任公司

图片二维码

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于图片二维码相关的 API 概览以及 SDK 示例代码。

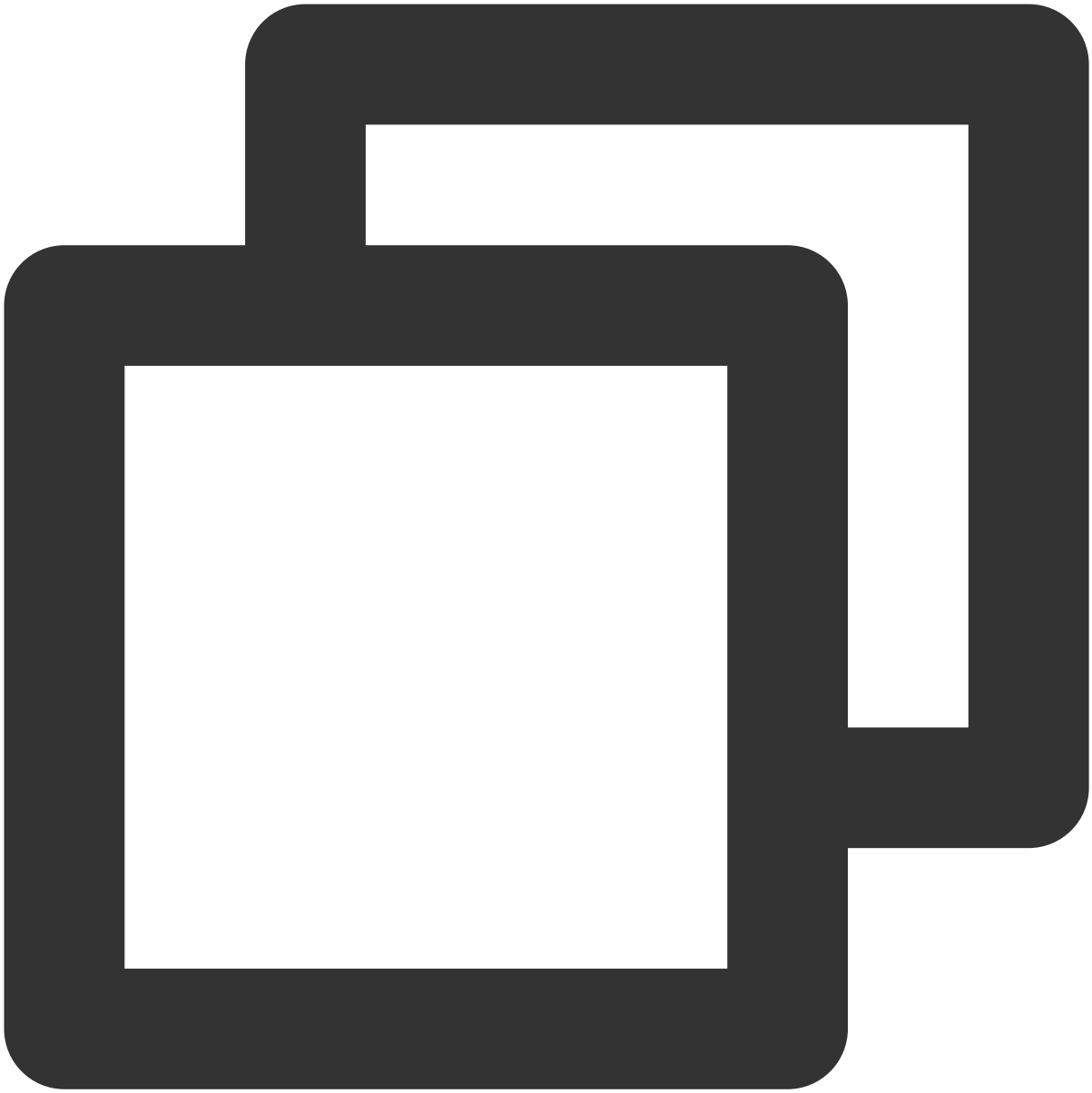
API	说明
二维码识别	二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克
二维码生成	二维码生成功能可根据用户指定的文本信息（URL 或文本），生成对应的二维码或条形码

二维码识别

功能说明

二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克。

请求示例1：上传时识别



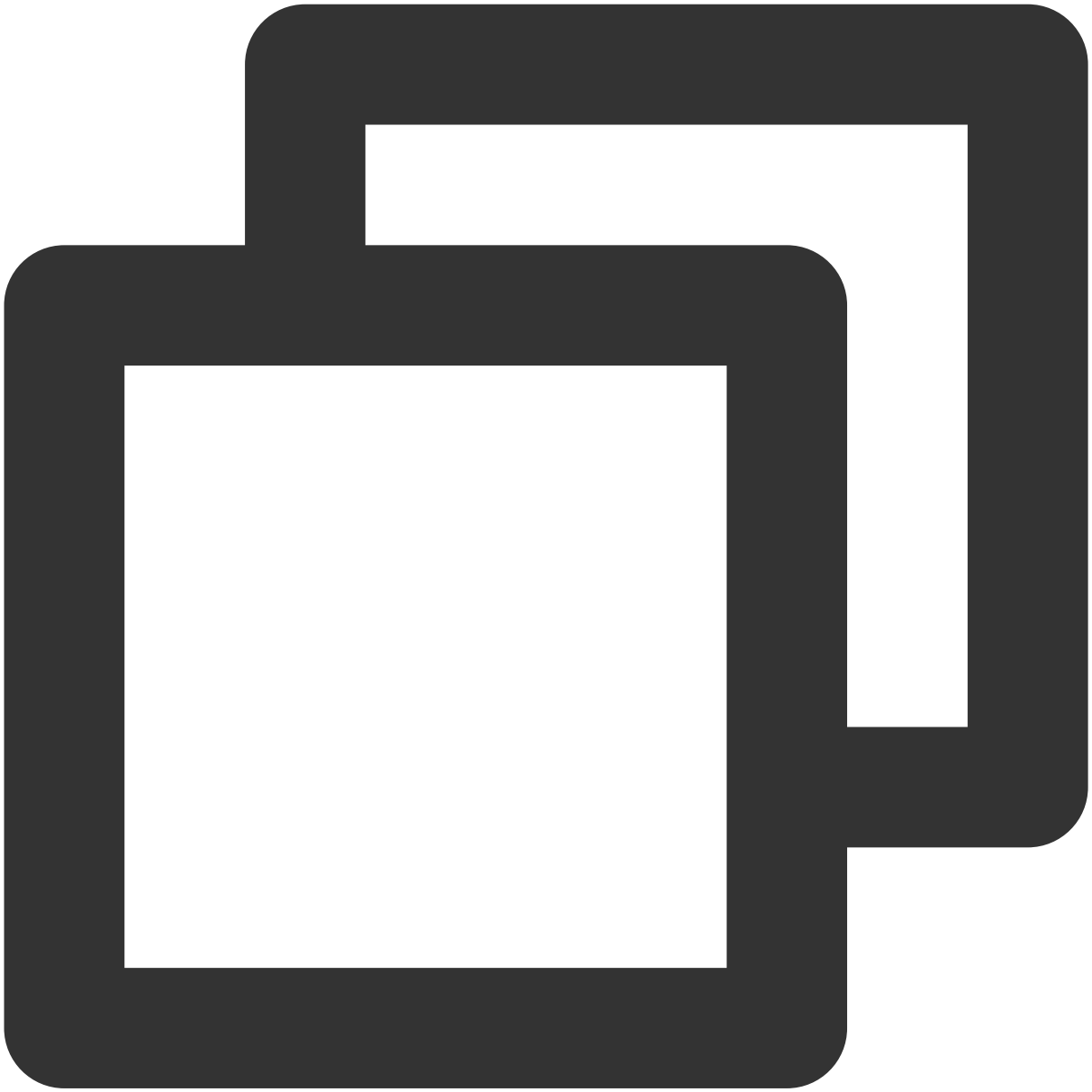
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $imageQrcodeTemplate = new Qcloud\\Cos\\ImageParamTemplate\\ImageQrcodeTemp
        $imageQrcodeTemplate->setMode(0);//二维码覆盖功能。0表示不开启二维码覆盖，1表示开启
        $picOperationsTemplate = new \\Qcloud\\Cos\\ImageParamTemplate\\PicOperatio
        $picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，
        $picOperationsTemplate->addRule($imageQrcodeTemplate, "resultobject");//设置
        $result = $cosClient->putObject(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'Body' => fopen('path/to/localFile', 'rb'),
        'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
Body	File/String	上传的内容	是
PicOperations	Json/String	图片持久化处理信息	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [Body] =>
            [ETag] => "698d51a19d8a121ce581499d7b701668"
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 238186
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
```

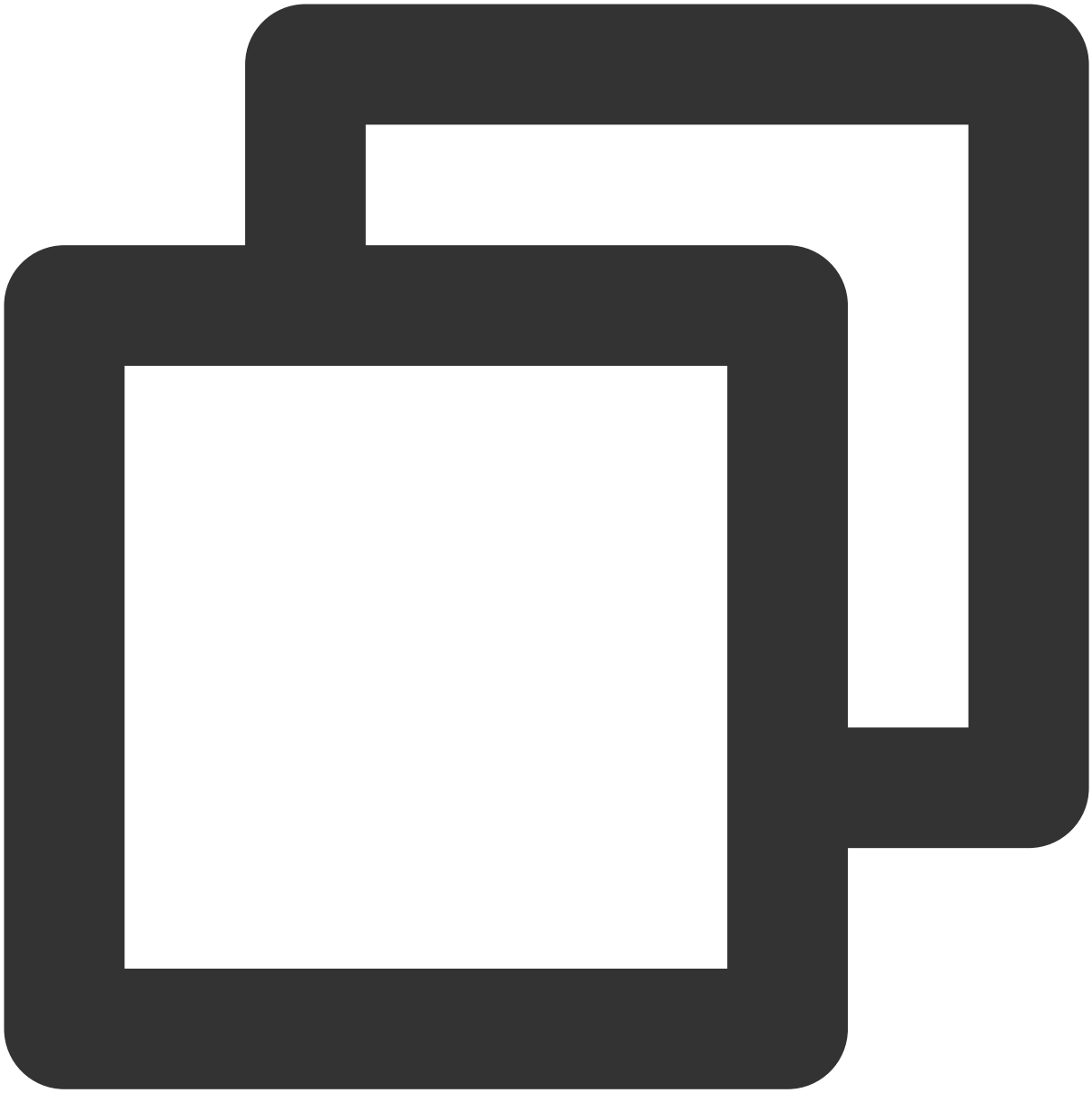
```
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [Data] => Array
            (
                [OriginalInfo] => Array
                (
                    [Key] => exampleobject
                    [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.
                    [ETag] => "7037fb6fb4cca43b958a28789605e73d98088720"
                    [ImageInfo] => Array
                    (
                            [Format] => JPEG
                            [Width] => 600
                            [Height] => 500
                            [Quality] => 90
                            [Ave] => 0x46442e
                            [Orientation] => 0
                     )

                )
                [ProcessResults] => Array
                (
                    [Object] => Array
                    (
                        [0] => Array(
                            [Key] => resultobject
                            [Location] => examplebucket-1250000000.cos.ap-beijing.m
                            [Format] => JPEG
                            [Width] => 300
                            [Height] => 200
                            [Size] => 30000
                            [Quality] => 90
                            [CodeStatus] => 1
                            [QRcodeInfo] => Array
                            (
                                    [0] => Array
                                    (
                                            [CodeUrl] => xxxxxxxxxxxx
                                            [CodeLocation] => Array
                                            (
                                                    [Point] => Array
                                                    (
                                                            [0] => 100,100
                                                            [1] => 100,200
                                                            [2] => 200,200
                                                            [3] => 200,100
                                                    )
                                             )
```

```
                                    )
                                )
                                [ETag] => "87c153bc2909aa0ba111ca126b675c510d36b817"
                            )
                        )
                    )
                )
            )
        )
    )
```

返回结果说明

参数名称	类型	描述	父节点
Body	File/String	返回体	无
ETag	String	文件的 MD5 值	无
RequestId	String	请求 ID 标识	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
Data	Array	图片处理结果信息(二维码信息)	无

请求示例2：下载时识别



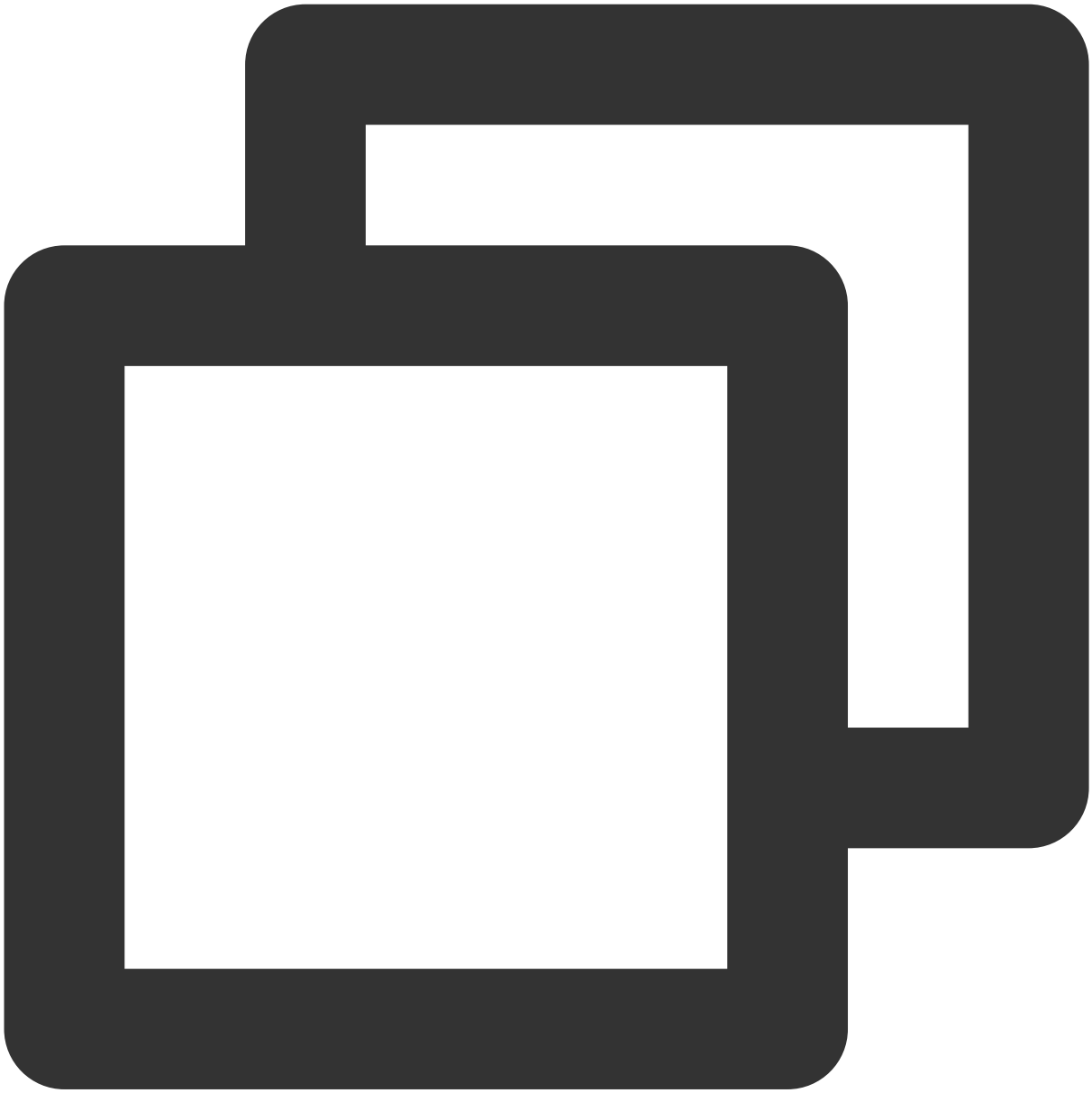
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->Qrcode(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Key' => 'exampleobject',
        'Cover' => 1,//二维码覆盖功能
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	此处的 Key 为对象键，对象键是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 `doc/pic.jpg`	是
Cover	Int	二维码覆盖功能，将对识别出的二维码覆盖上马赛克。0表示不开启二维码覆盖，1表示开启二维码覆盖，默认值0	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [ContentLength] => 238186
            [Key] => exampleobject
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exam
            [CodeStatus] => 1
```

```
            [QRcodeInfo] => Array
            (
                    [0] => Array
                    (
                            [CodeUrl] => xxxxxxxxxxxx
                            [CodeLocation] => Array
                            (
                                    [Point] => Array
                                    (
                                            [0] => 100,100
                                            [1] => 100,200
                                            [2] => 200,200
                                            [3] => 200,100
                                    )
                             )
                    )
            )
            [ResultImage] =>

    )
)
```

返回结果说明

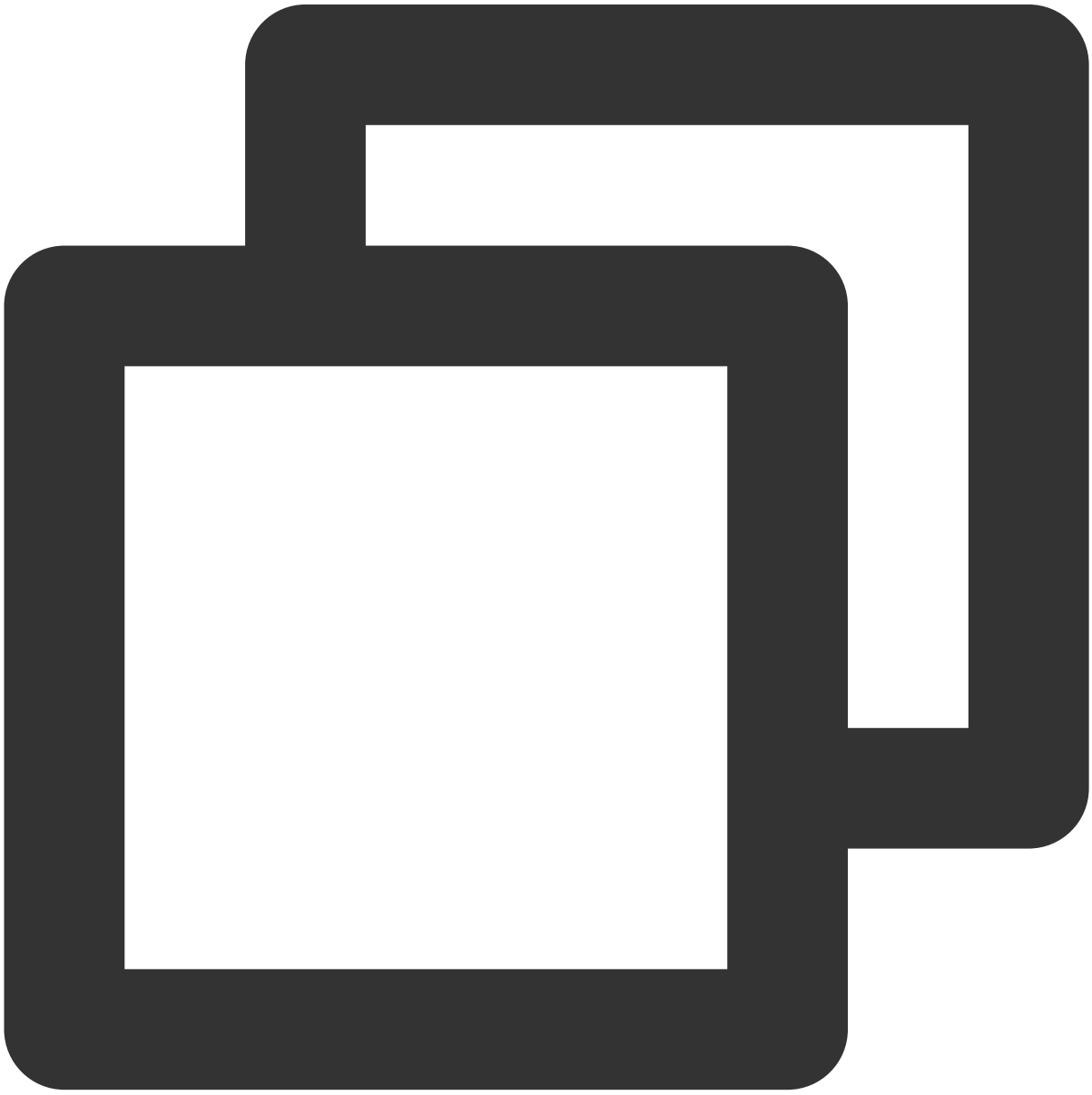
参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
ContentLength	Int	响应体长度	无
Key	String	对象键	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
CodeStatus	Int	二维码识别结果。0表示未识别到二维码，1表示识别到二维码	无
QRcodeInfo	Array	二维码识别结果，可能有多个	无
ResultImage	String	处理后的图片 base64数据，请求参数 cover 为1时返回	无

二维码生成

功能说明

二维码生成功能可根据用户指定的文本信息（URL 或文本），生成对应的二维码或条形码。

示例代码



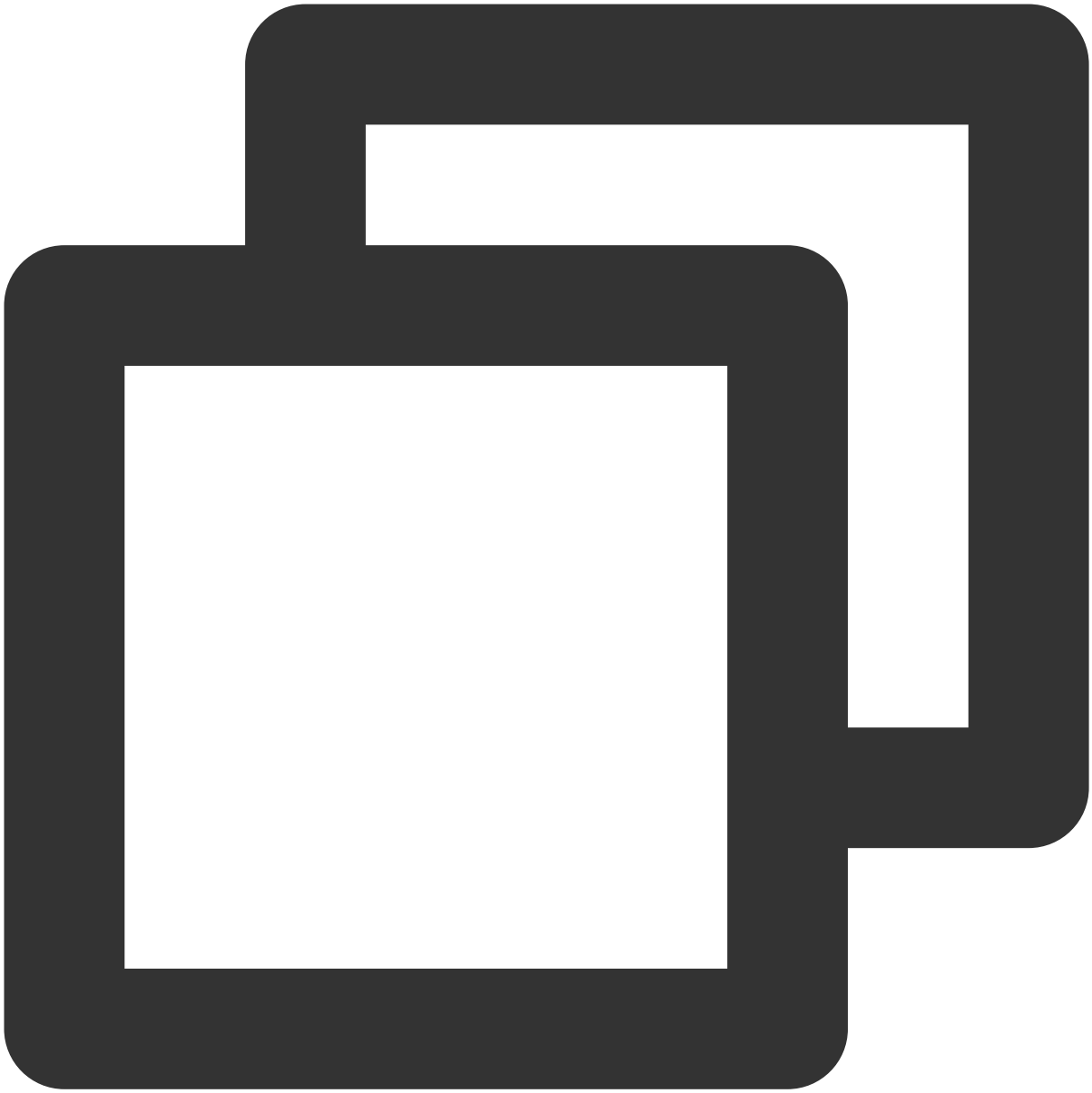
```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
```

```
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));

try {
        $result = $cosClient->QrcodeGenerate(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'QrcodeContent' => '<https://www.example.com>',//可识别的二维码文本信息
        'QrcodeMode' => 0 //生成的二维码类型，可选值：0或1。0为二维码，1为条形码，默认值为0
        'QrcodeWidth' => '200',//指定生成的二维码或条形码的宽度，高度会进行等比压缩
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
QrcodeContent	String	可识别的二维码文本信息	是
QrcodeMode	Int	生成的二维码类型，可选值：0或1。0为二维码，1为条形码，默认值为0	是
QrcodeWidth	String	指定生成的二维码或条形码的宽度，高度会进行等比压缩	是

返回结果示例



```
Guzzle\\Service\\Resource\\Model Object
(
    [structure:protected] =>
    [data:protected] => Array
    (
            [RequestId] => NWQwOGRkNDdfMjJiMjU4NjRfNzVjXzEwNmVjY2M=
            [Bucket] => examplebucket-1250000000
            [Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/
            [ResultImage] =>
    )
)
```

返回结果说明

参数名称	类型	描述	父节点
RequestId	String	请求 ID 标识	无
Bucket	String	存储桶名称，格式：BucketName-APPID	无
Location	String	请求资源地址	无
ResultImage	String	处理后的图片 base64数据	无

版权所有：腾讯云计算（北京）有限责任公司

任务接口
动图任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交动图任务的 API 概览和 SDK 示例代码。

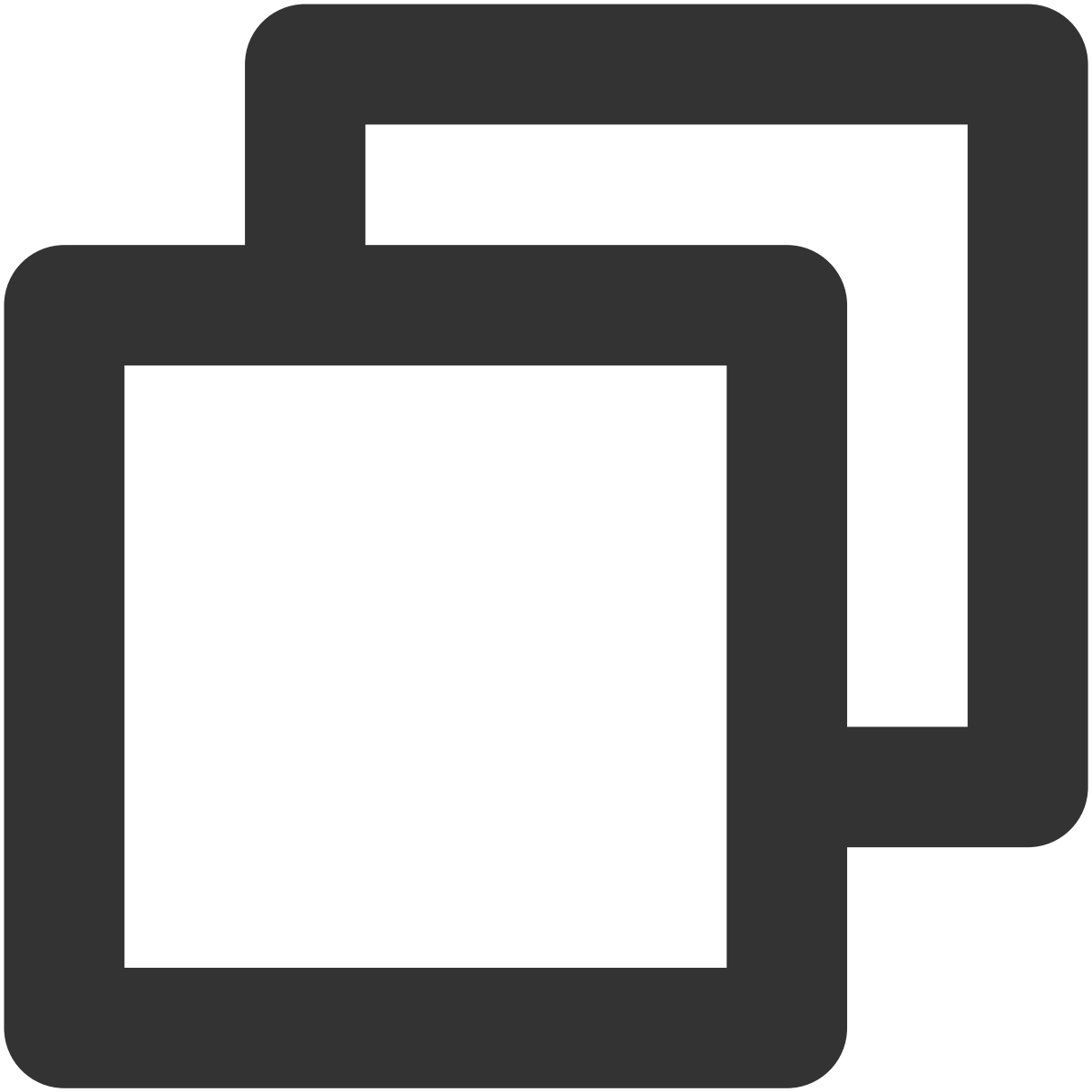
API	操作描述
提交动图任务	提交一个动图任务

提交动图任务

功能说明

用于提交一个动图任务。

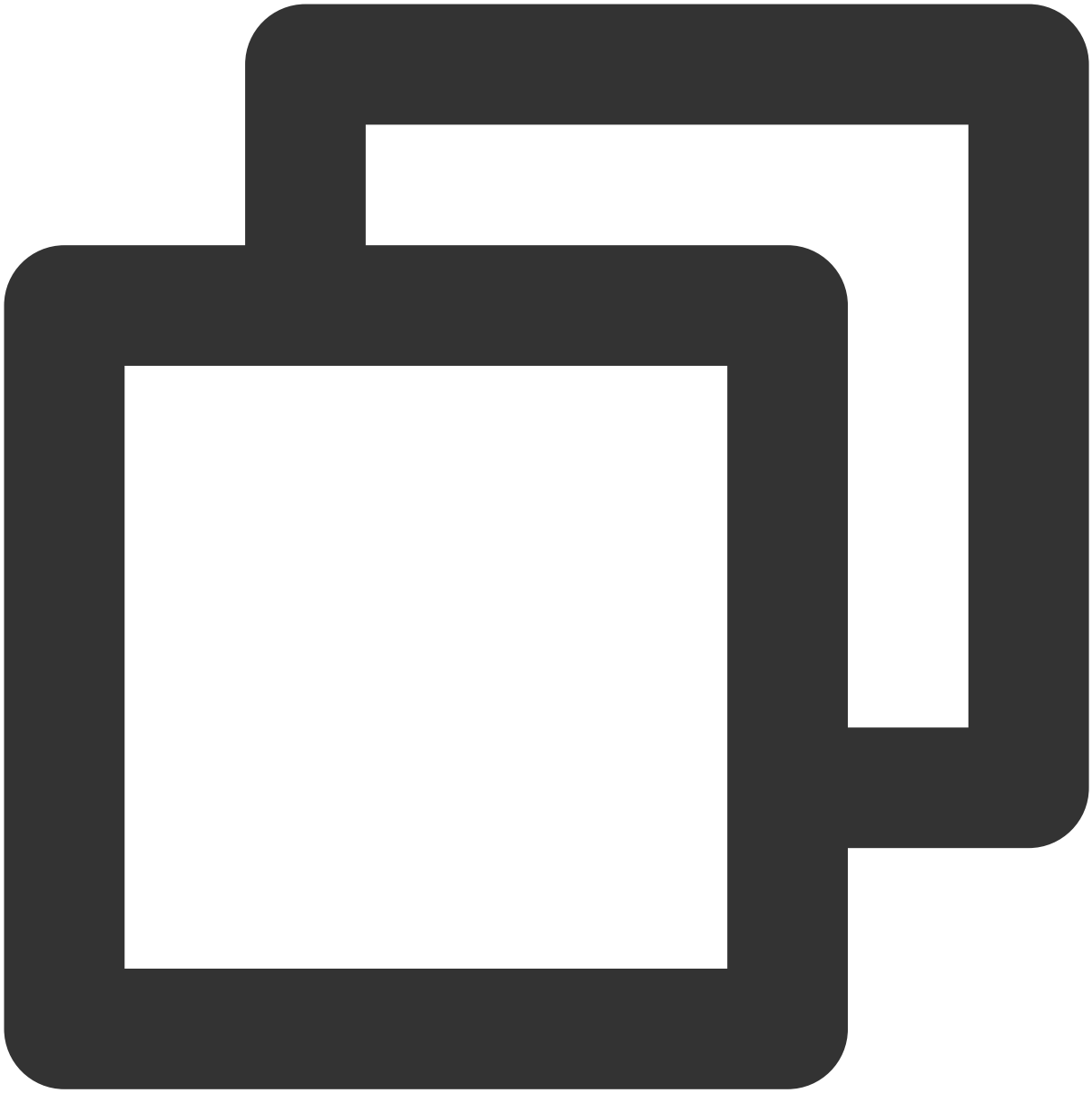
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaAnimationJobs(array $args = arra
```

请求示例

示例一： 使用模板

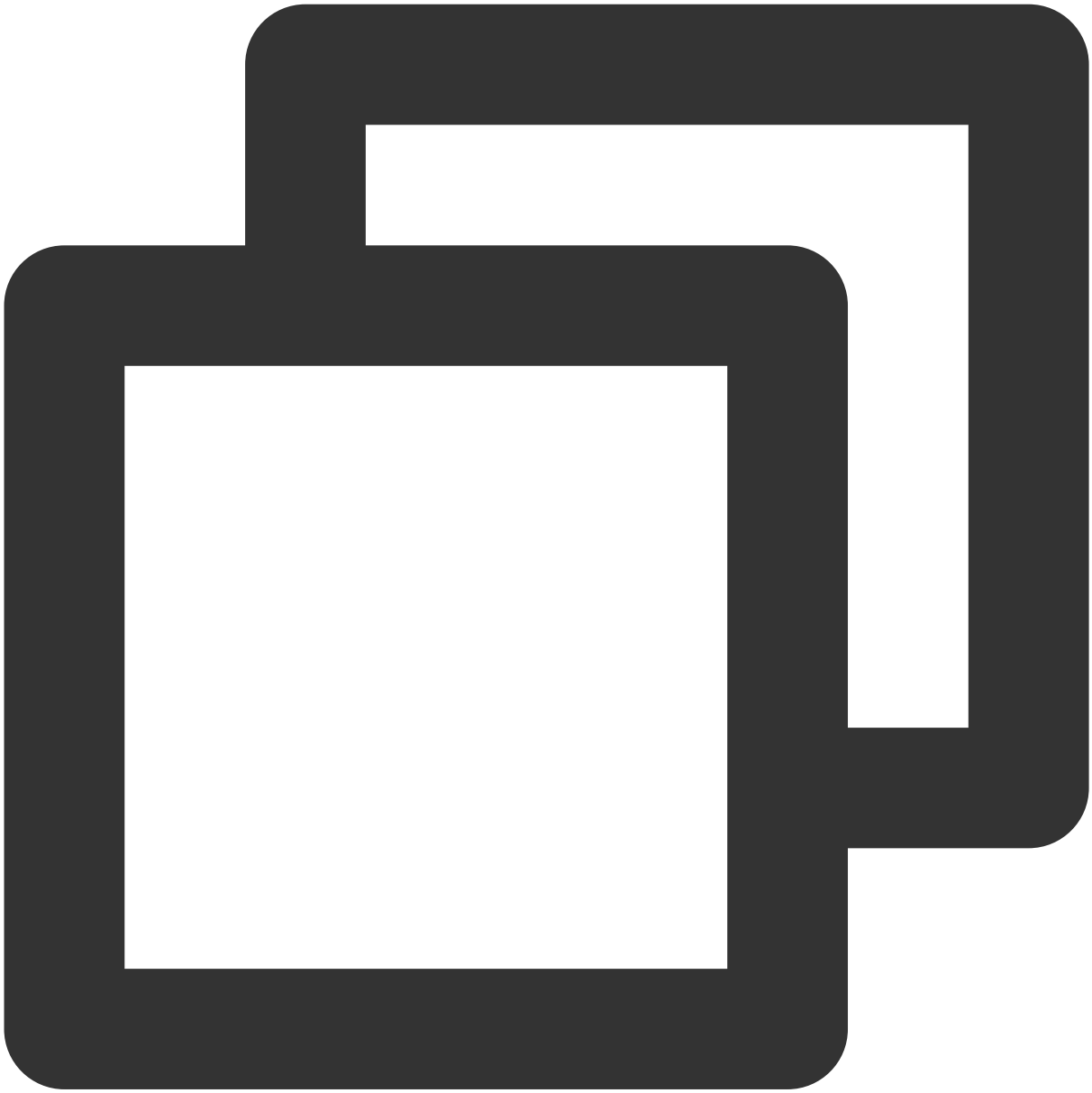


```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

版权所有：腾讯云计算（北京）有限责任公司

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaAnimationJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Animation',
        'QueueId' => 'p81e648af2aee49688xxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 't1de276cbdab16xxxxxxxxxxxxxxxxxxxx',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'Animation.gif',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaAnimationJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Animation',
        'QueueId' => 'p81e648af2aee49688xxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'Animation' => array(
                'Container' => array(
                    'Format' => '',
                ),
                'Video' => array(
                    'Codec' => '',
                    'Width' => '',
                    'Height' => '',
                    'Fps' => '',
                    'AnimateOnlyKeepKeyFrame' => '',
                    'AnimateTimeIntervalOfFrame' => '',
                    'AnimateFramesPerSecond' => '',
                    'Quality' => '',
                ),
                'TimeInterval' => array(
                    'Start' => '',
                    'Duration' => '',
                ),
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'Animation.gif',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：Animation	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Animation	Request.Operation	指定该任务的参数	Container	否
TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是

版权所有：腾讯云计算（北京）有限责任公司

UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过 1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

注意：

优先使用 TemplateId，无 TemplateId 时使用 Animation。

Container 类型 Animation 的具体数据描述如下：

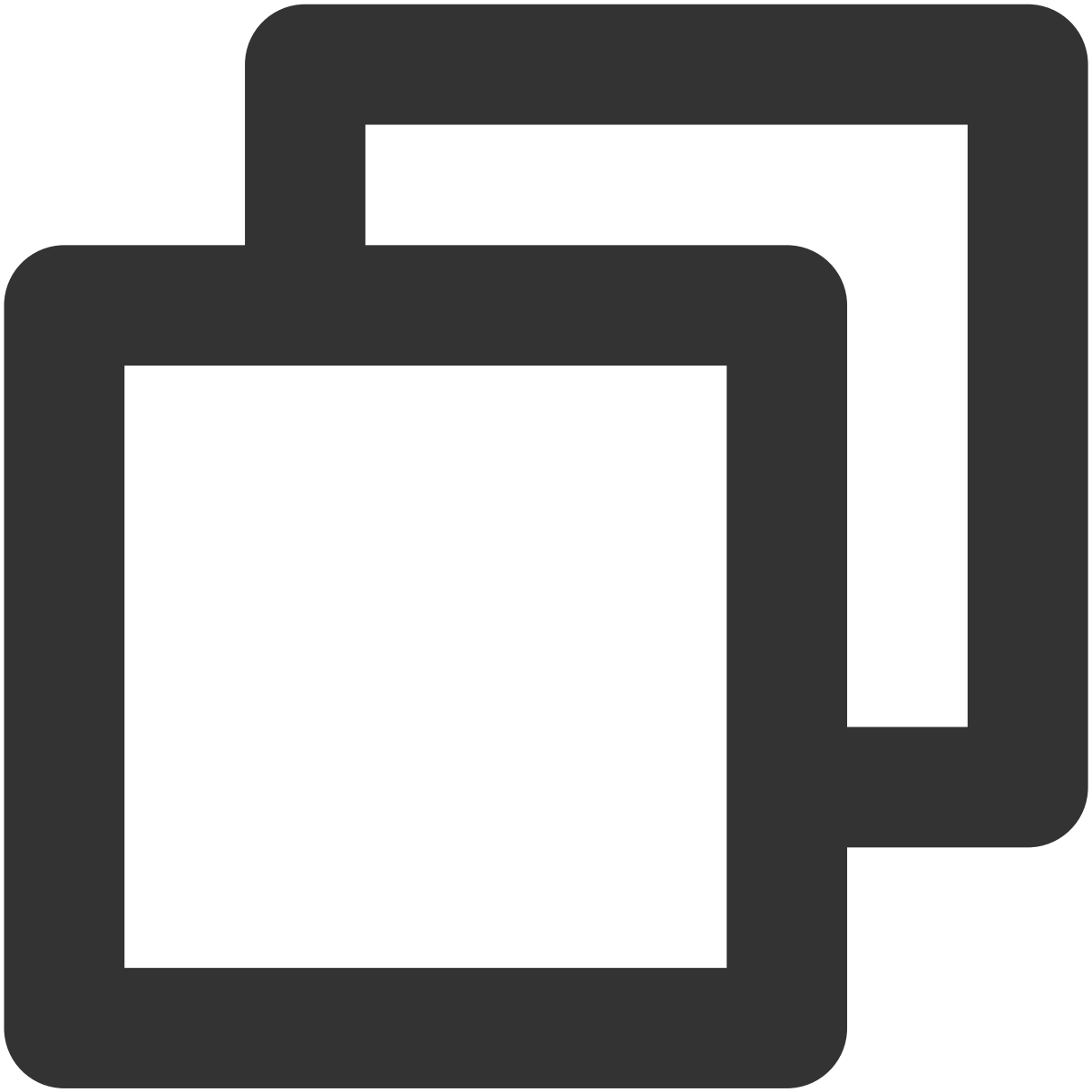
节点名称（关键字）	父节点	描述	类型	是否必选
Container	Request.Operation.Animation	同动图模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Animation	同动图模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
TimeInterval	Request.Operation.Animation	同动图模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例

版权所有：腾讯云计算（北京）有限责任公司



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

```
                (
                )

        )

    [RequestId] => NjI2MTQHOAIHDOIDUOIA7879ADJOIMDc2YTk=
    [ContentType] => application/xml
    [ContentLength] => 797
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-21T19:29:36+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => j5318zx9c890zx8c0z98c0fa6355174de
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => Animation.gif
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t1de276cbdz78xcz09xc80zx901af71
                            [TemplateName] => Animation-gif-1
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2aeez87c6z78xc7z98xc79ca86
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => Animation
                )
```

```
            )

)
```

版权所有：腾讯云计算（北京）有限责任公司

截图任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交截图任务的 API 概览和 SDK 示例代码。

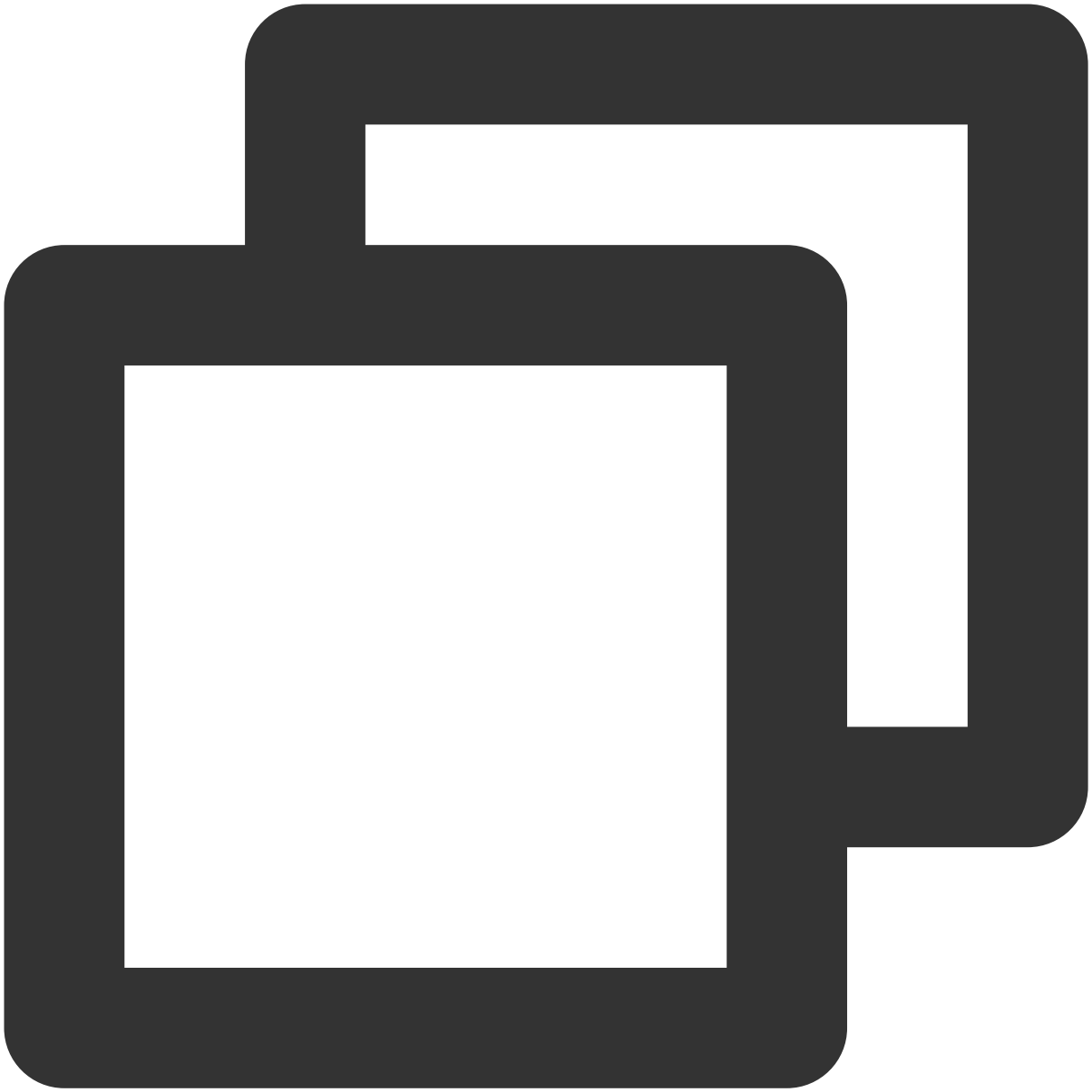
API	操作描述
提交截图任务	提交一个截图任务

提交截图任务

功能说明

用于提交一个截图任务。

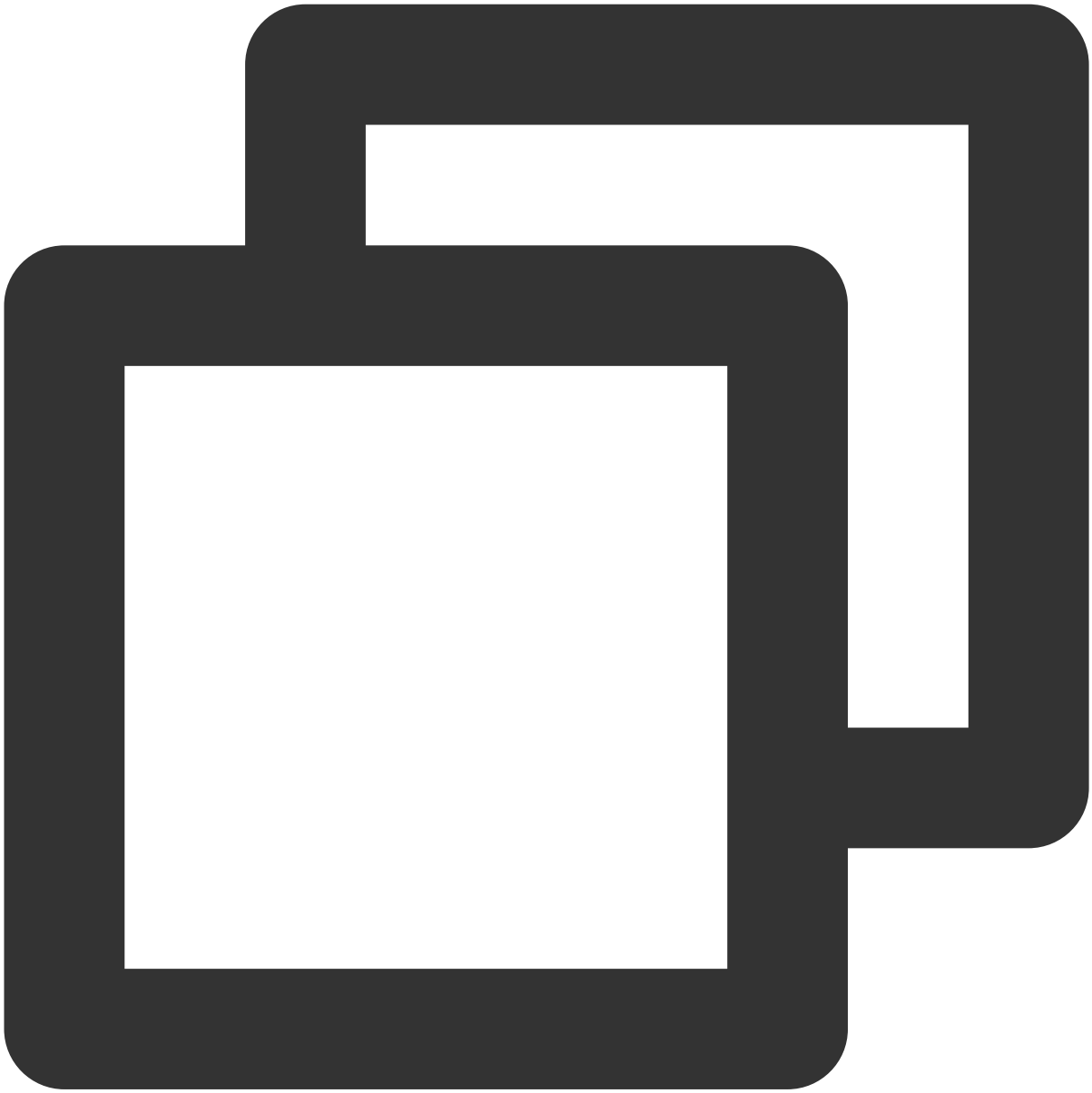
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaSnapshotJobs(array $args = array
```

请求示例

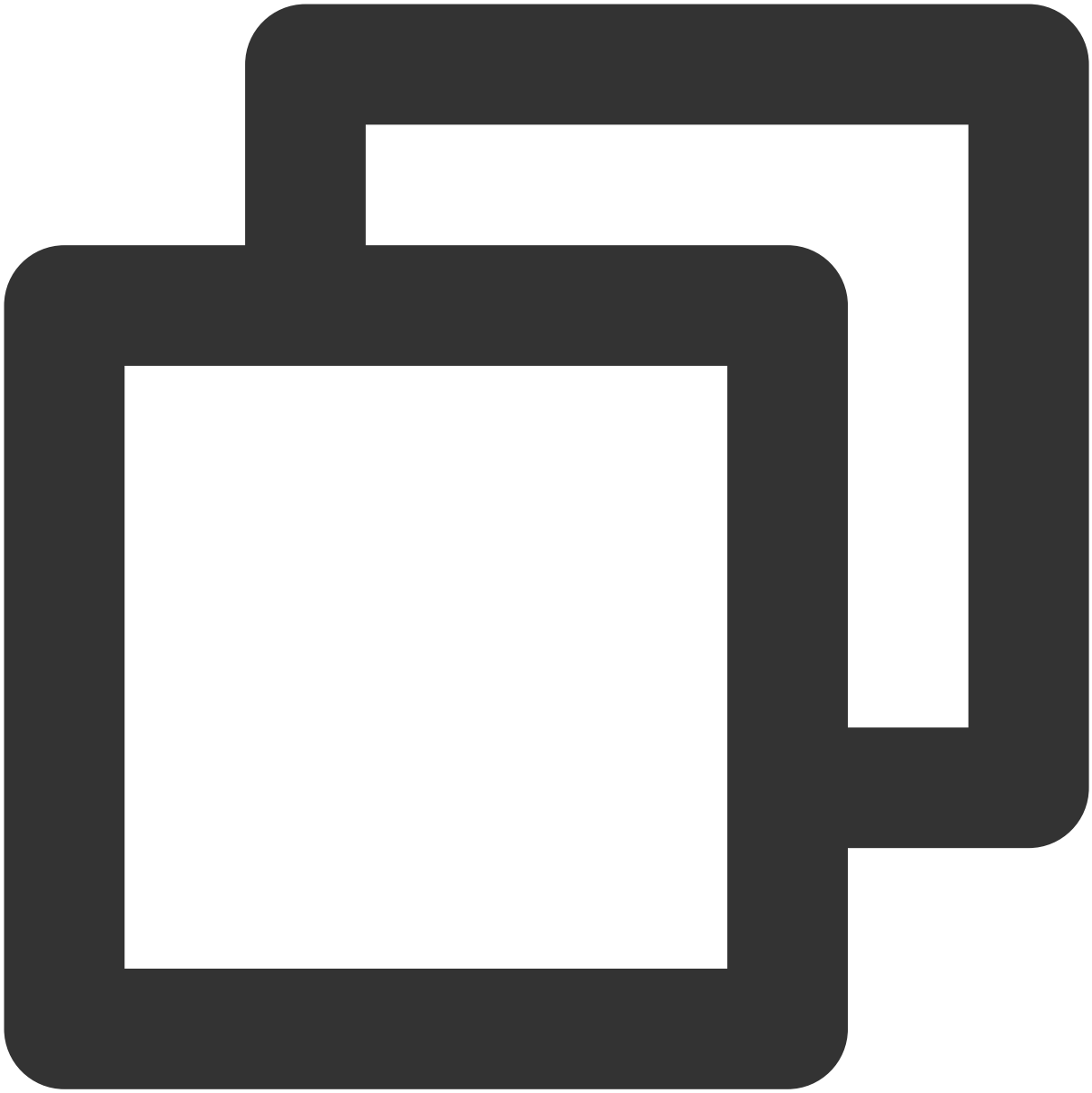
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaSnapshotJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Snapshot',
        'QueueId' => 'asdadadfafsdkjhfjghdfjg',
        'CallBack' => 'https://example.com/callback',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 'asdfafiahfiushdfisdhfuis',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'snapshot-${Number}.jpg',
//                'SpriteObject' => 'sprite-${Number}.jpg',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
//        'CallBackMqConfig' => array(
//            'MqRegion' => '',
//            'MqMode' => '',
//            'MqName' => '',
//        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaSnapshotJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Snapshot',
        'QueueId' => 'asdadadfafsdkjhfjghdfjg',
        'CallBack' => 'https://example.com/callback',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'snapshot-${Number}.jpg',
            ),
            'Snapshot' => array(
                'Mode' => 'Average',
                'Start' => 3,
                'TimeInterval' => '',
                'Count' => 3,
                'Width' => '1280',
                'Height' => '',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：Snapshot	String	是

Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Snapshot	Request.Operation	指定该任务的参数，同创建截图模板 CreateMediaTemplate 接口中的 Request.Snapshot	Container	否
TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过 1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

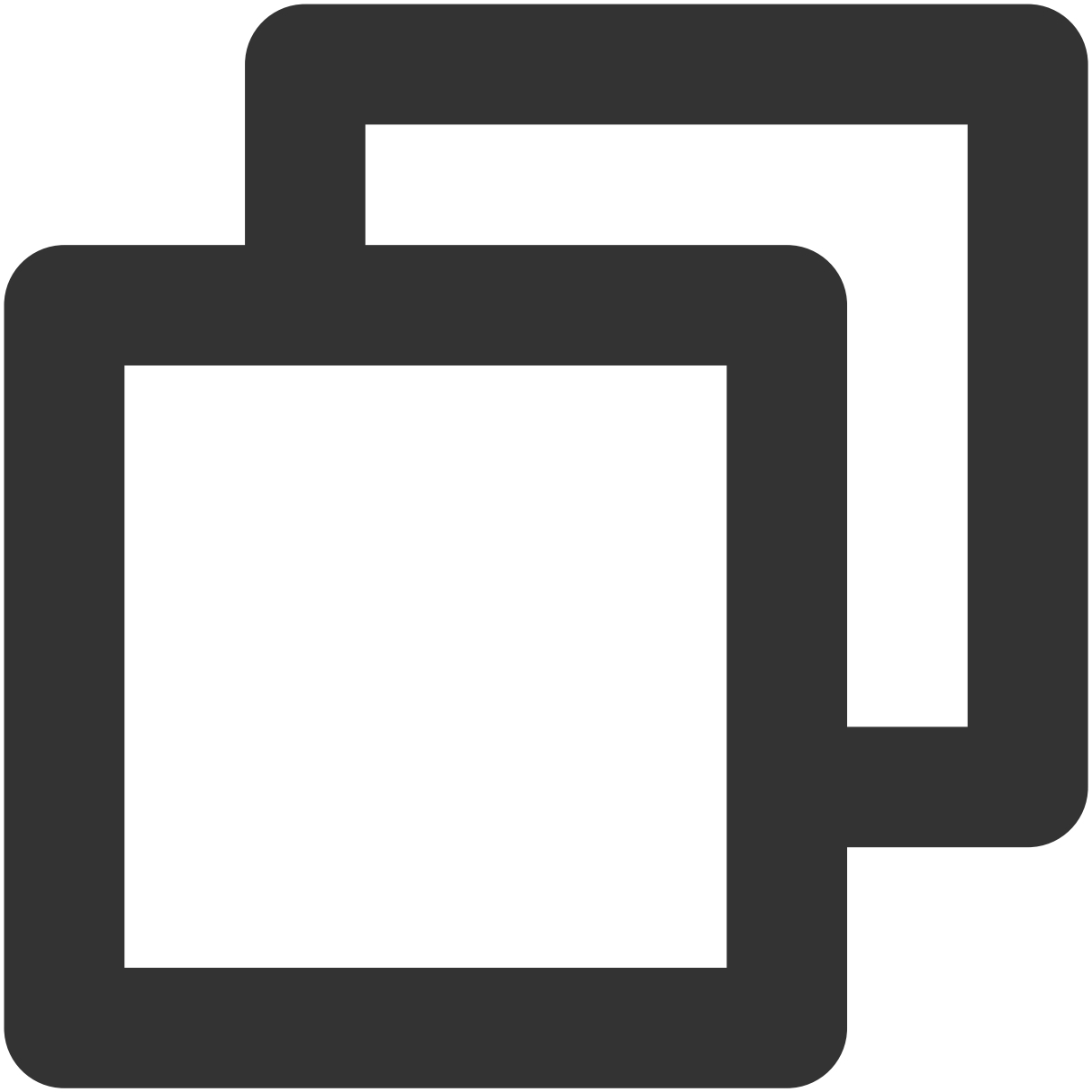
注意：

优先使用 TemplateId，无 TemplateId 时使用 Snapshot。

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	结果文件的名字。 **必须包含 ${Number} 参数。** 如 Object 为 snapshot-${Number}.jpg	String	否
SpriteObject	Request.Operation.Output	雪碧图的名字。 **必须包含 ${Number} 参数。** 如 Object 为 sprite-${Number}.jpg ** 仅支持 jpg 格式	String	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

```
                (
                )

        )

    [RequestId] => NjI2MTNkYAHDUIAHDRfNjQ0OF8yMGI4MjQ=
    [ContentType] => application/xml
    [ContentLength] => 829
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-21T19:19:25+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => je6dcd3d6a80d98a09ds8ac4214b9b
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => snapshot-${Number}.jpg
                                    [Region] => ap-beijing
                                    [SpriteObject] =>
                                )

                            [TemplateId] => t0778zc908zc908z0c8z0adcffd54a
                            [TemplateName] => snapshot_1280*720_3
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648afz7zc08c0z8xc09e086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => Snapshot
                )
```

```
            )

)
```

转码任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交转码任务的 API 概览和 SDK 示例代码。

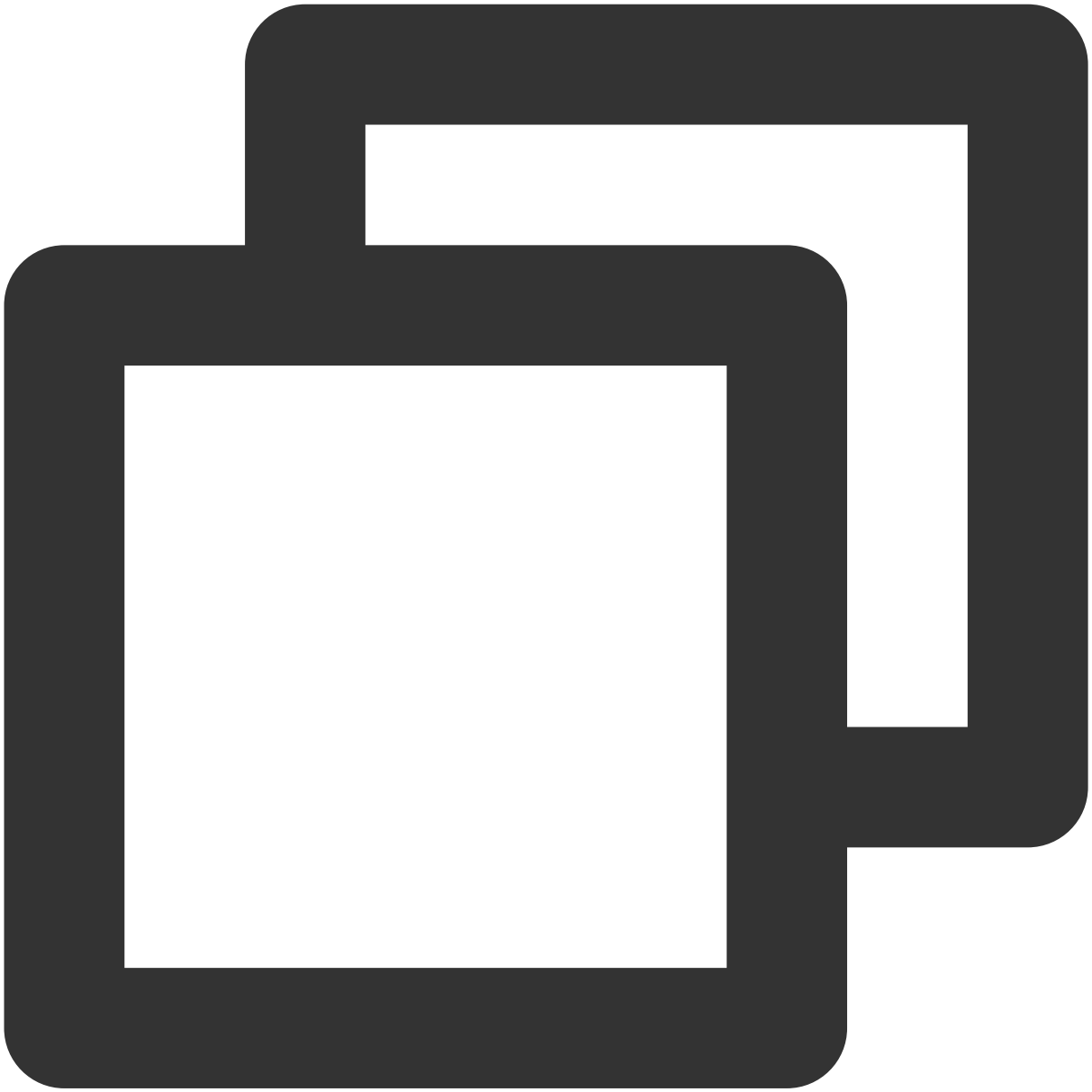
API	操作描述
提交转码任务	提交转码任务

提交转码任务

功能说明

提交转码任务。

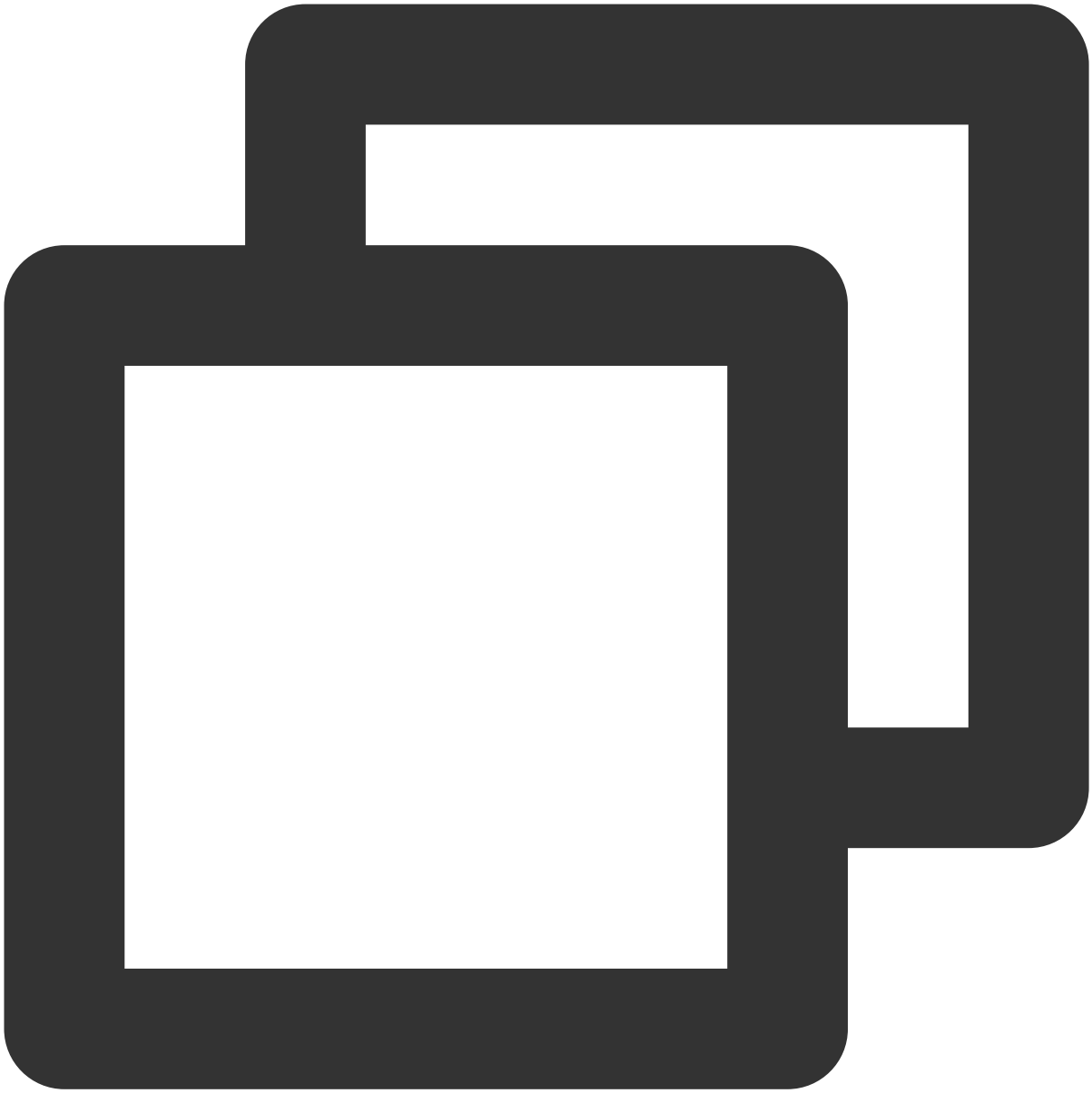
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaTranscodeJobs(array $args = arra
```

请求示例

示例一： 使用模板

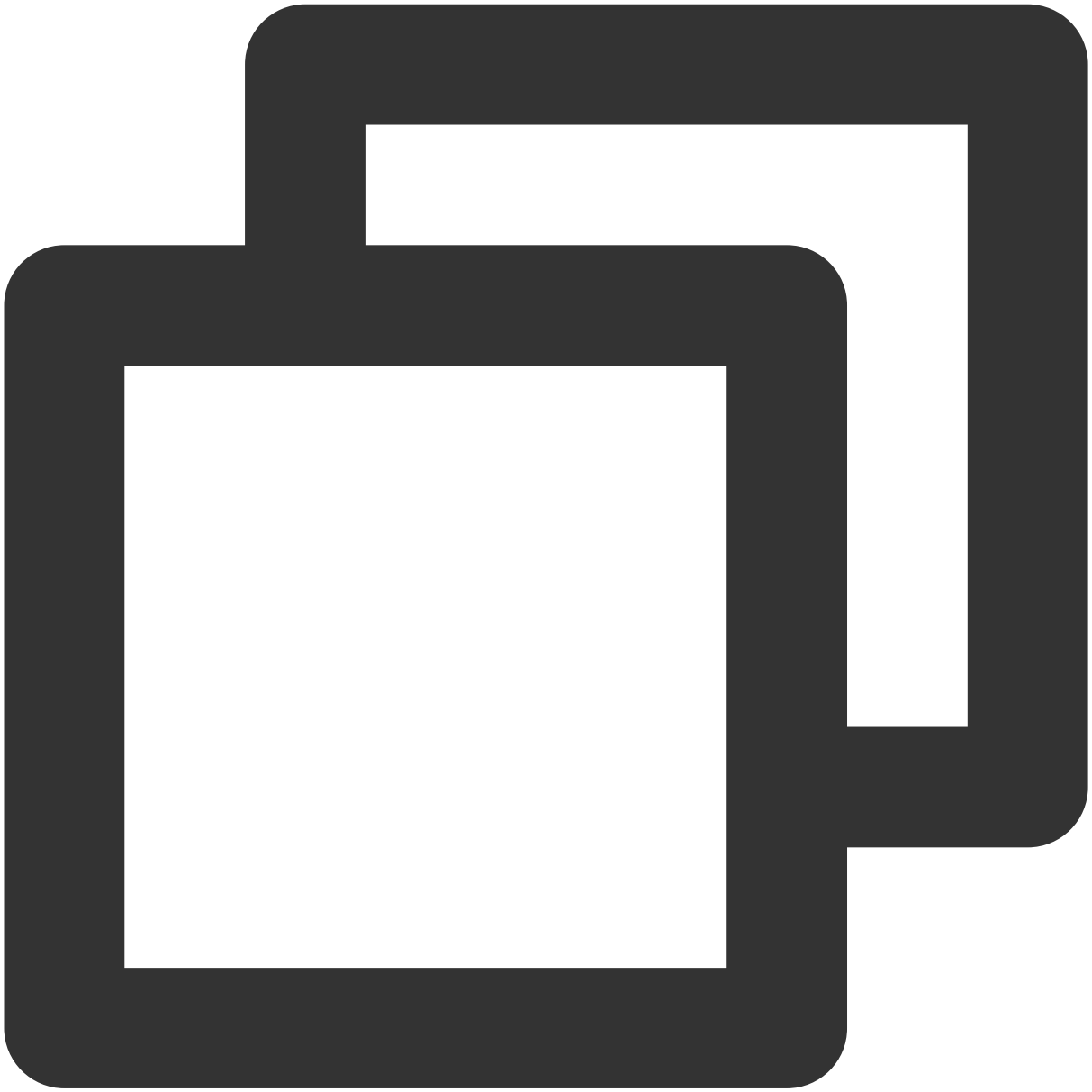


```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaTranscodeJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Transcode',
        'QueueId' => 'paaf4fce5521a40888a3034a5de80f6ca',
        'Input' => array(
            'Object' => 'example.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 't04e1ab86554984f1aa17c062fbf6c007c',
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'video02.mp4',
            ),
            'Watermark' => array(
                array(
                    'Type' => 'Text',
                    'LocMode' => 'Absolute',
                    'Dx' => '64',
                    'Dy' => '64',
                    'Pos' => 'TopRight',
                    'Text' => array(
                        'Text' => '第一个水印',
                        'FontSize' => '30',
                        'FontType' => 'simfang.ttf',
                        'FontColor' => '#99ff00',
                        'Transparency' => '100', // 不透明度
                     ),
                ),
                array(
                    'Type' => 'Text',
                    'LocMode' => 'Absolute',
                    'Dx' => '64',
                    'Dy' => '64',
                    'Pos' => 'TopLeft',
                    'Text' => array(
                        'Text' => '第二个水印',
                        'FontSize' => '30',
                        'FontType' => 'simfang.ttf',
                        'FontColor' => '#99ff00',
                        'Transparency' => '100', // 不透明度
```

```
                        ),
                    ),
                ),
            ),
        ));
        // 请求成功
        print_r($result);
    } catch (\\Exception $e) {
        // 请求失败
        echo($e);
    }
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaTranscodeJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Transcode',
        'QueueId' => 'asdadadfafsdkjhfjghdfjg',
        'CallBack' => 'https://example.com/callback',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'video01.mkv',
            ),
            'Transcode' => array(
                'Container' => array(
                    'Format' => 'mp4'
                ),
                'Video' => array(
                    'Codec' => 'H.264',
                    'Profile' => 'high',
                    'Bitrate' => '1000',
                    'Preset' => 'medium',
                    'Width' => '1280',
                    'Fps' => '30',
                ),
                'Audio' => array(
                    'Codec' => 'aac',
                    'Samplerate' => '44100',
                    'Bitrate' => '128',
                    'Channels' => '4',
                ),
                'TransConfig' => array(
                    'AdjDarMethod' => 'scale',
                    'IsCheckReso' => 'false',
                    'ResoAdjMethod' => '1',
                ),
                'TimeInterval' => array(
                    'Start' => '0',
                    'Duration' => '60',
                ),
```

```
            ),
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：Transcode	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
TemplateId	Request.Operation	指定的模板 ID	String	否
Transcode	Request.Operation	指定转码模板参数	Container	否
WatermarkTemplateId	Request.Operation	指定的水印模板 ID，可以传多个水印模板 ID	String	否
Watermark	Request.Operation	指定水印模板参数，同创建水印模板 CreateMediaTemplate 接口中的 Request.Watermark	Container	否
RemoveWatermark	Request.Operation	指定去除水印参数, H265暂不支持次参数	Container	否
DigitalWatermark	Request.Operation	指定数字水印参数	Container	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

说明：

对于转码参数，优先使用优先使用 TemplateId，无 TemplateId 时使用 Transcode；对于水印参数，可以使用 WatermarkTemplateId 或 Watermark 设置，WatermarkTemplateId 优先级更高。

Container 类型 Transcode 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
TimeInterval	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否
Container	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
Audio	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate	Container	否

版权所有：腾讯云计算（北京）有限责任公司

		接口中的 Request.Audio		
TransConfig	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TransConfig	Container	否
AudioMix	Request.Operation.Transcode	混音参数，详情见AudioMix	Container 数组	否

Container 类型 RemoveWatermark 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Dx	Request.Operation.RemoveWatermark	距离左上角原点 x 偏移，范围为[1, 4096]	string	是
Dy	Request.Operation.RemoveWatermark	距离左上角原点 y 偏移，范围为[1, 4096]	string	是
Width	Request.Operation.RemoveWatermark	宽，范围为[1, 4096]	string	是
Height	Request.Operation.RemoveWatermark	高，范围为[1, 4096]	string	是

Container 类型 DigitalWatermark 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Message	Request.Operation.DigitalWatermark	数字水印嵌入的字符串信息，长度不超过64个字符，仅支持中文、英文、数字、_、-和*	string	是
Type	Request.Operation.DigitalWatermark	水印类型，当前仅可设置为 Text	String	是
Version	Request.Operation.DigitalWatermark	水印版本，当前仅可设置为 V1	String	是
IgnoreError	Request.Operation.DigitalWatermark	当添加水印失败是否忽略错误继续执行任务，限制为 true/false，默认为 false	string	是

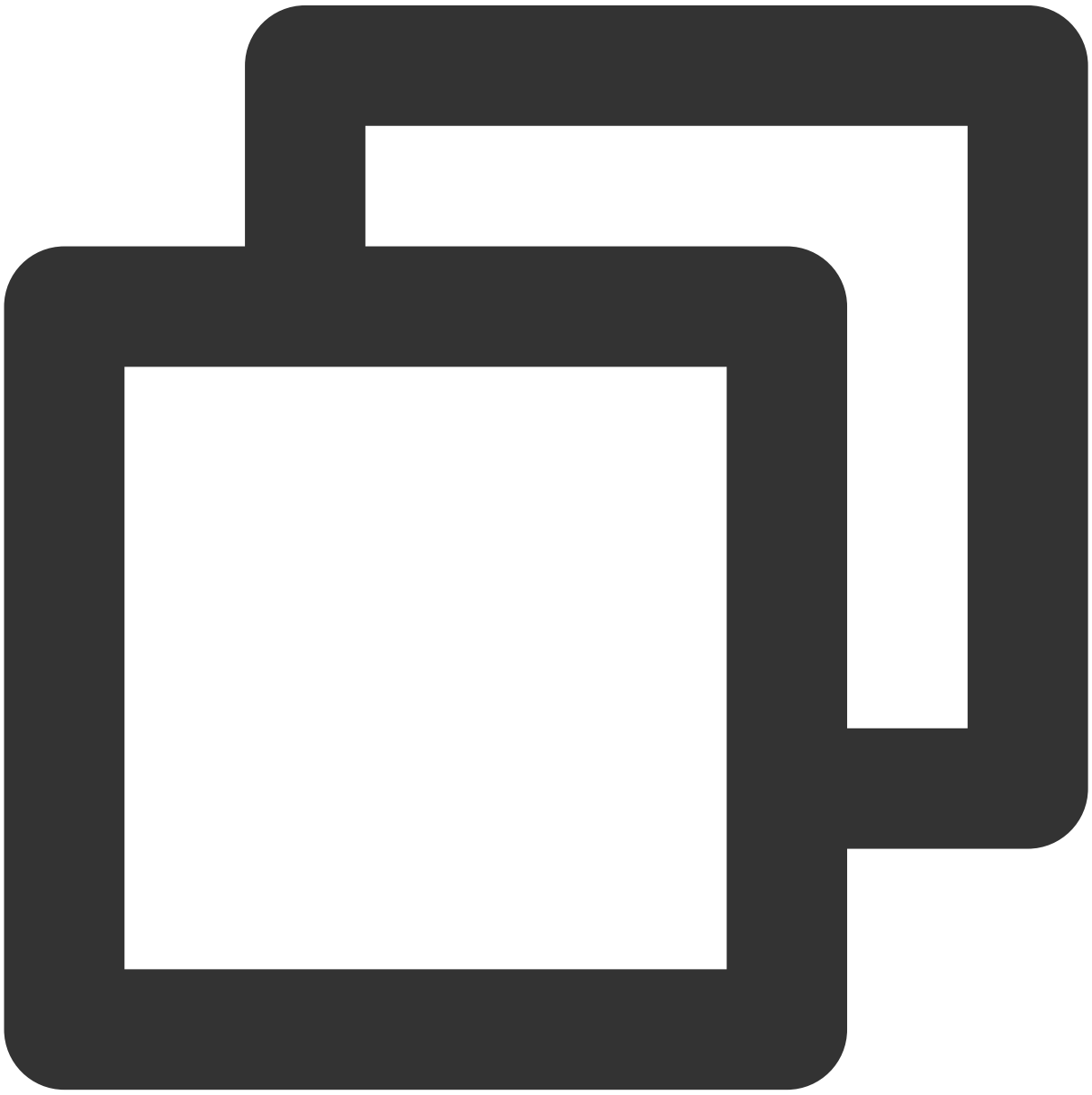
Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是

版权所有：腾讯云计算（北京）有限责任公司

Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
```

```
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
                (
                )

        )

    [RequestId] => NjI2MTAzasdaHUISDHUINzE3YV8xZWVmNjU=
    [ContentType] => application/xml
    [ContentLength] => 1554
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-21T15:10:15+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => j1815a4c81h2kj31231gh31kj2f9e223404e
                    [Message] => Array
                        (
                        )

                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => Transcode.flv
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t0b612860as90d8a09sd65dca38
                            [TemplateName] => FLV-SD
                            [UserData] => xxx
                            [JobLevel] => 0
```

```
                [Watermark] => Array
                    (
                        [0] => Array
                            (
                                [Dx] => 64
                                [Dy] => 64
                                [EndTime] => Array
                                    (
                                    )

                                [LocMode] => Absolute
                                [Pos] => TopRight
                                [StartTime] => Array
                                    (
                                    )

                                [Text] => Array
                                    (
                                        [FontColor] => #99ff00
                                        [FontSize] => 30
                                        [FontType] => simfang.ttf
                                        [Text] => 第一个水印
                                        [Transparency] => 100
                                    )

                                [Type] => Text
                            )

                        [1] => Array
                            (
                                [Dx] => 64
                                [Dy] => 64
                                [EndTime] => Array
                                    (
                                    )

                                [LocMode] => Absolute
                                [Pos] => TopLeft
                                [StartTime] => Array
                                    (
                                    )

                                [Text] => Array
                                    (
                                        [FontColor] => #99ff00
                                        [FontSize] => 30
                                        [FontType] => simfang.ttf
```

```
                                                    [Text] => 第二个水印
                                                    [Transparency] => 100
                                                )

                                            [Type] => Text
                                        )

                                    )

                                )

                            [Progress] => 0
                            [QueueId] => p81e648a7s9d88a09sd89a0757be086
                            [StartTime] => -
                            [State] => Submitted
                            [Tag] => Transcode
                        )

                )

)
```

拼接任务接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于提交拼接任务的 API 概览和 SDK 示例代码。

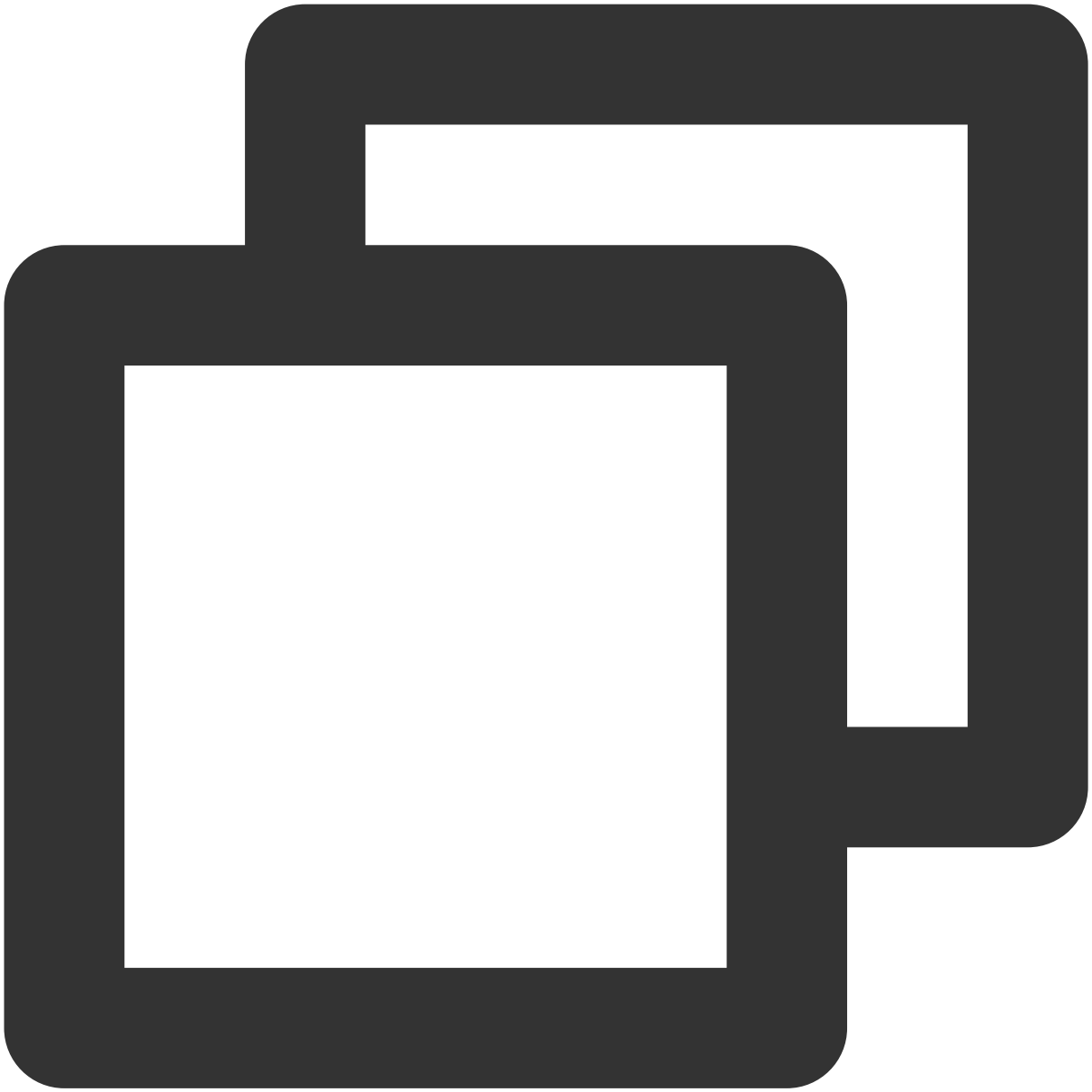
API	操作描述
提交拼接任务	提交一个拼接任务

提交拼接任务

功能说明

用于提交一个拼接任务。

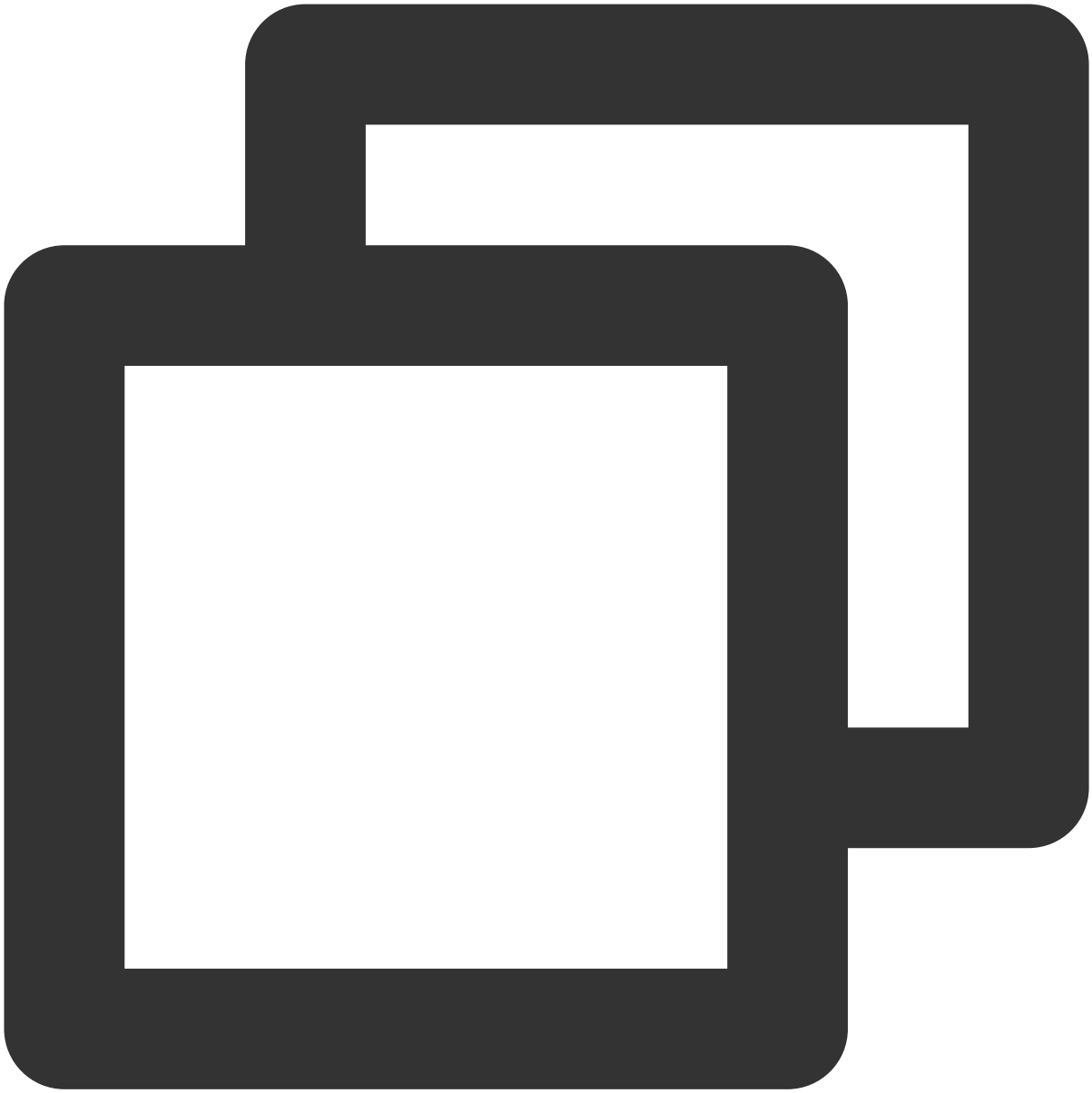
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaConcatJobs(array $args = array()
```

请求示例

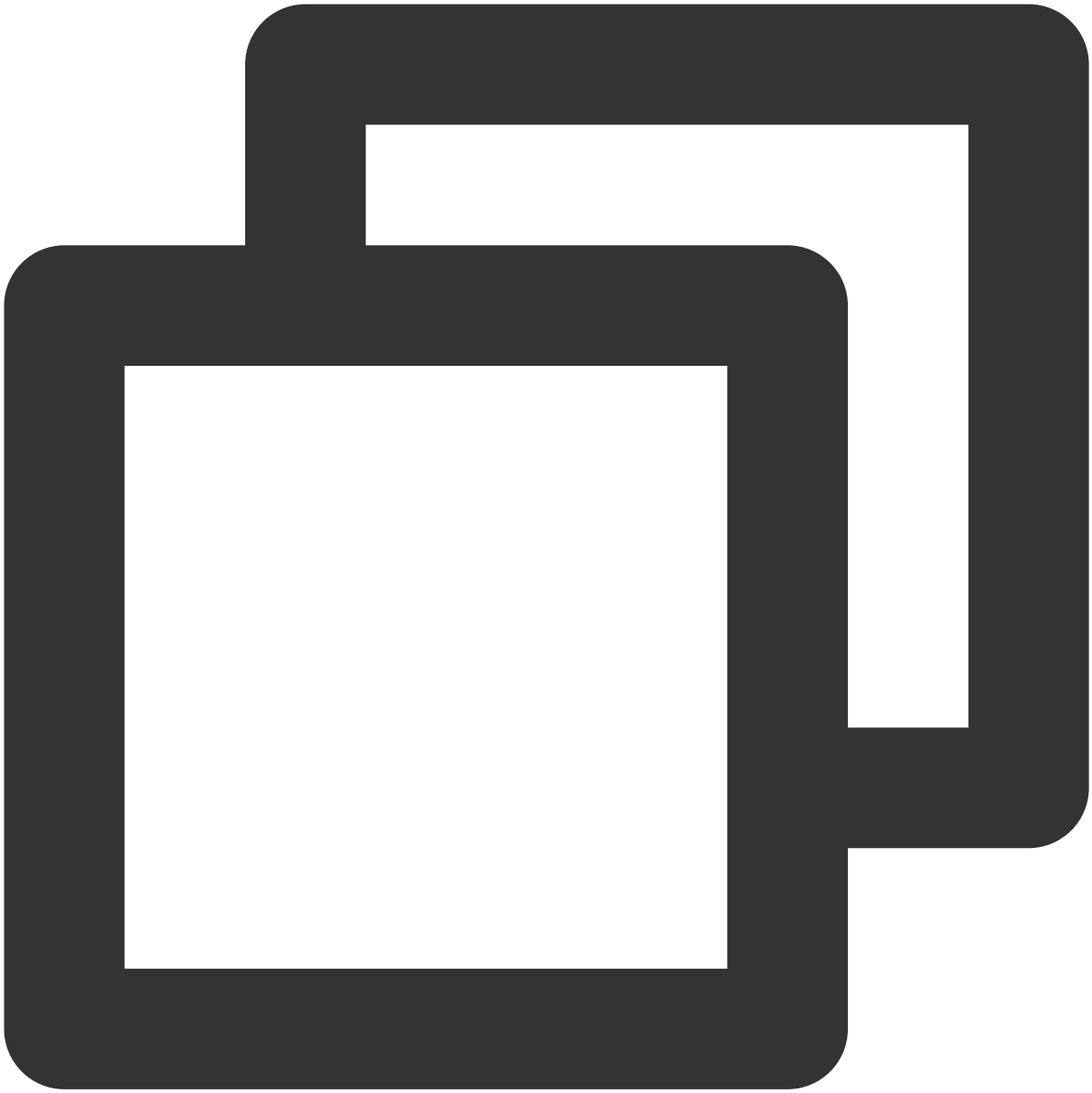
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaConcatJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Concat',
        'QueueId' => 'asdadadfafsdkjhfjghdfjg',
        'CallBack' => 'https://example.com/callback',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 'asdfafiahfiushdfisdhfuis',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'concat-video02.mp4',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaConcatJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Concat',
        'QueueId' => 'asdadadfafsdkjhfjghdfjg',
        'CallBack' => 'https://example.com/callback',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'concat-video03.mp4',
            ),
            'ConcatTemplate' => array(
                'ConcatFragments' => array(
                    array(
                        'Url' => 'https://example.com/video01.mp4',
                        'Mode' => 'Start',
//                        'StartTime' => '0',
//                        'EndTime' => '7',
                    ),
                    array(
                        'Url' => 'https://example.com/video02.mp4',
                        'Mode' => 'Start',
//                        'StartTime' => '0',
//                        'EndTime' => '10',
                    ),
                    // ... repeated
                ),
                'Index' => 1,
                'Container' => array(
                    'Format' => 'mp4'
                ),
                'Audio' => array(
                    'Codec' => 'mp3',
                    'Samplerate' => '',
                    'Bitrate' => '',
                    'Channels' => '',
                ),
                'Video' => array(
```

```
                    'Codec' => 'H.264',
                    'Bitrate' => '1000',
                    'Width' => '1280',
                    'Height' => '',
                    'Fps' => '30',
                ),
            ),
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：Concat	String	是
Input	Request	待操作的媒体信息	Container	否
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选

Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
ConcatTemplate	Request.Operation	指定拼接参数	Container	否
TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0、1、2。级别越大任务优先级越高，默认为0	String	否

注意：

优先使用 TemplateId，无 TemplateId 时使用 ConcatTemplate。

Container 类型 ConcatTemplate 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	默认值
ConcatFragment	Request.Operation.ConcatTemplate	拼接节点	Container 数组	否	无
Audio	Request.Operation.ConcatTemplate	音频参数，同创建拼接模板 CreateMediaTemplate 接口中的 Request.ConcatTemplate.Audio	Container	否	无
Video	Request.Operation.ConcatTemplate	视频参数，同创建拼接模板 CreateMediaTemplate 接口中的 Request.ConcatTemplate.Video	Container	否	无
Container	Request.Operation.ConcatTemplate	封装格式，同创建拼接模板 CreateMediaTemplate 接口中的 Request.ConcatTemplate.Container	Container	是	无
AudioMix	Request.Operation.ConcatTemplate	混音参数，详情见 AudioMix	Container 数组	否	无

版权所有：腾讯云计算（北京）有限责任公司

Index	Request.Operation.ConcatTemplate	Input 节点位于 ConcatFragment 序列索引	String	否	0
DirectConcat	Request.Operation.ConcatTemplate	简单拼接方式（不转码直接拼接），其他的视频和音频参数失效	String	否	false

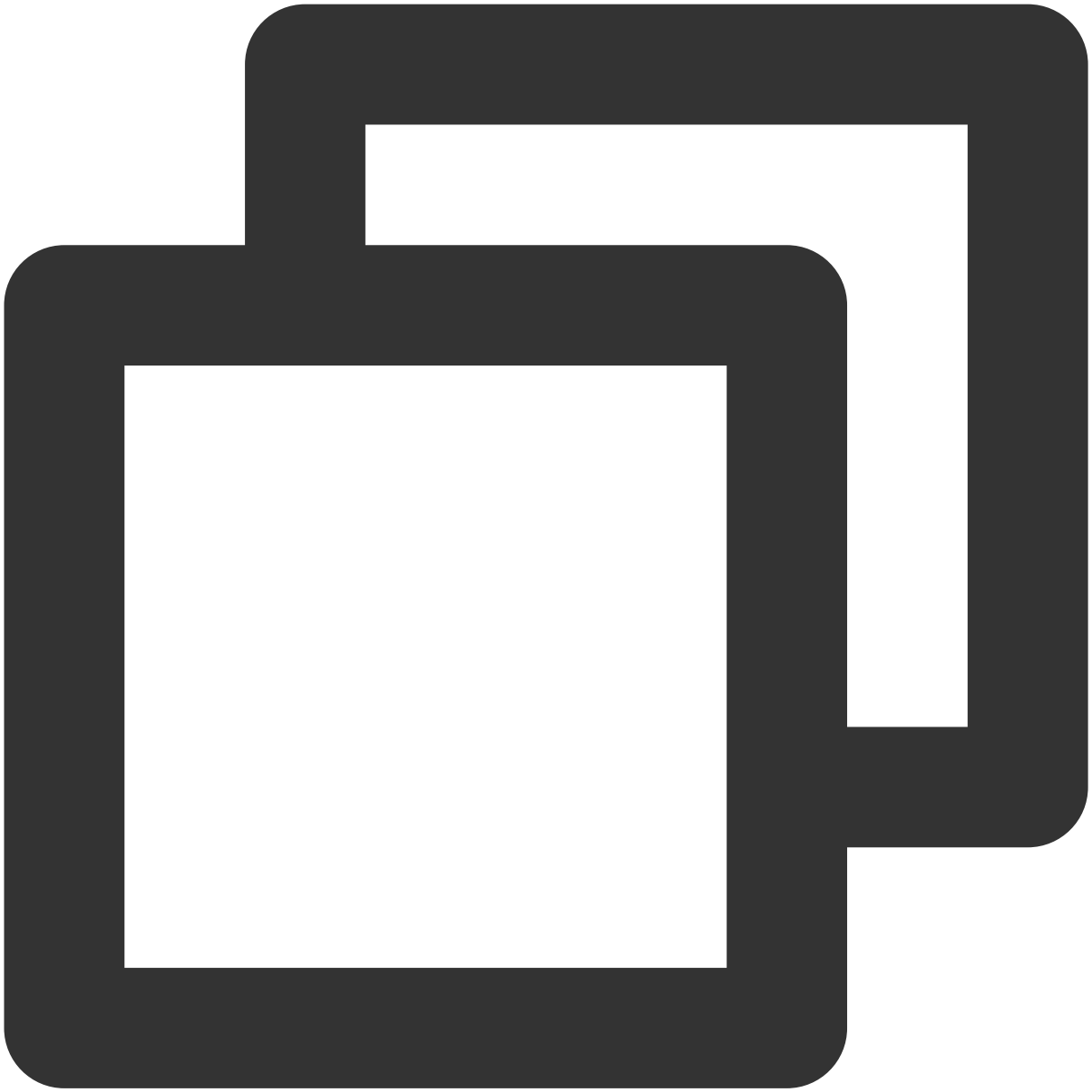
Container 类型 ConcatFragment 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Url	Request.Operation.ConcatTemplate.ConcatFragment	拼接对象地址	String	是	无	同 bucket 对象文件
FragmentIndex	Request.Operation.ConcatTemplate.ConcatFragment	拼接对象的索引位置	String	否	0	大于等于0的整数
StartTime	Request.Operation.ConcatTemplate.ConcatFragment	开始时间	String	否	视频开始	[0 视频时长]单位为秒
EndTime	Request.Operation.ConcatTemplate.ConcatFragment	结束时间	String	否	视频结束	[0 视频时长]单位为秒

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2MjFhOTHASIUDHAUISDHl8yMjMxNmE=
    [ContentType] => application/xml
    [ContentLength] => 792
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-22T11:01:42+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => j898f573asd780a8sd90a8sd09e30cf31f
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => concat-video01.mp4
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t17b9c2e7a0s8d0a8d0a9d43552f648e
                            [TemplateName] => concat-1
                            [UserData] => xxx
                            [TemplateName] => 0
                        )

                    [QueueId] => p81e648af2aeas78d0a8d09a8d09757be086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => Concat
                )
```

```
            )

)
```

版权所有：腾讯云计算（北京）有限责任公司

智能封面任务接口

最近更新时间：2024-03-14 09:31:03

简介

本文档提供关于提交智能封面任务的 API 概览和 SDK 示例代码。

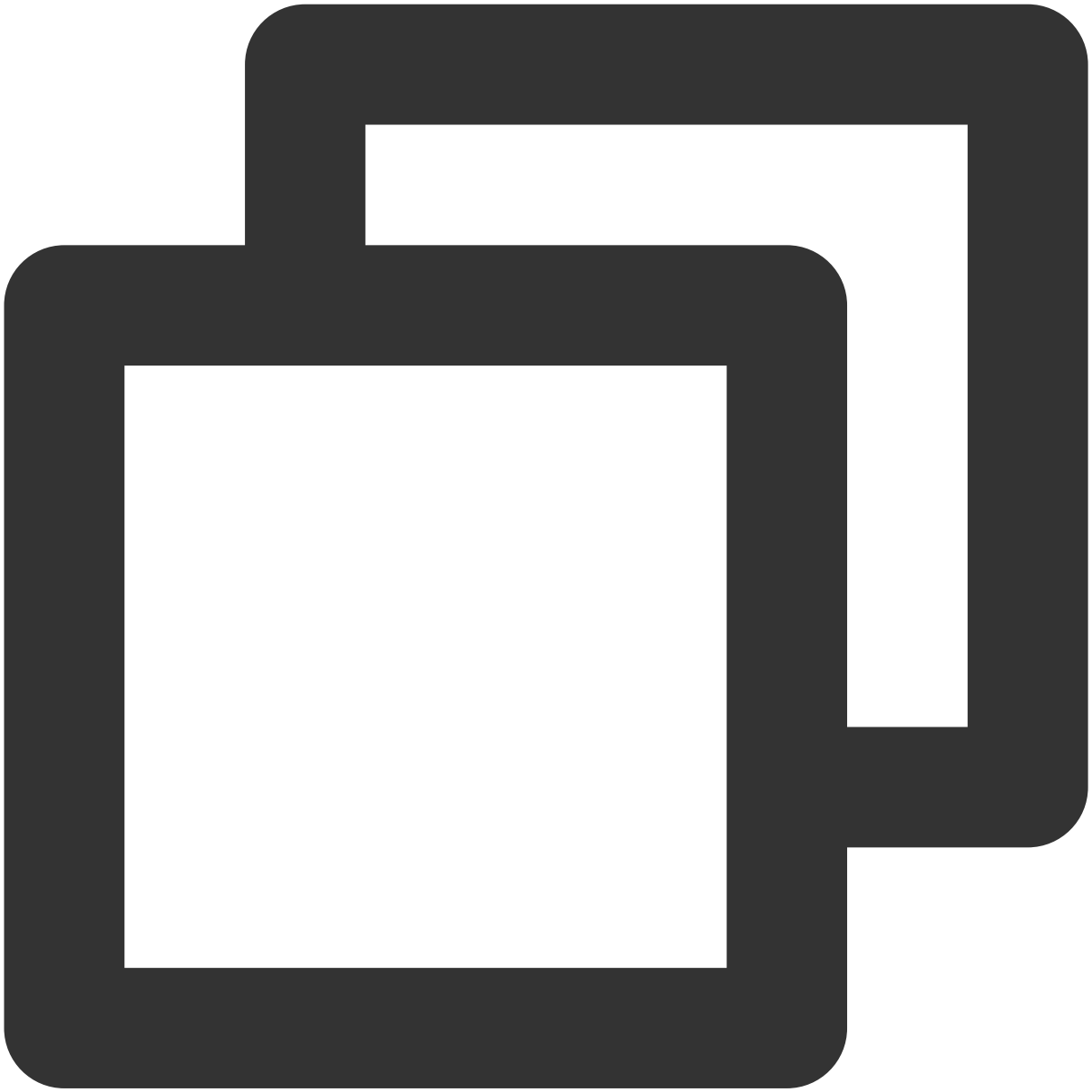
API	操作描述
提交智能封面任务	提交一个智能封面任务

提交智能封面任务

功能说明

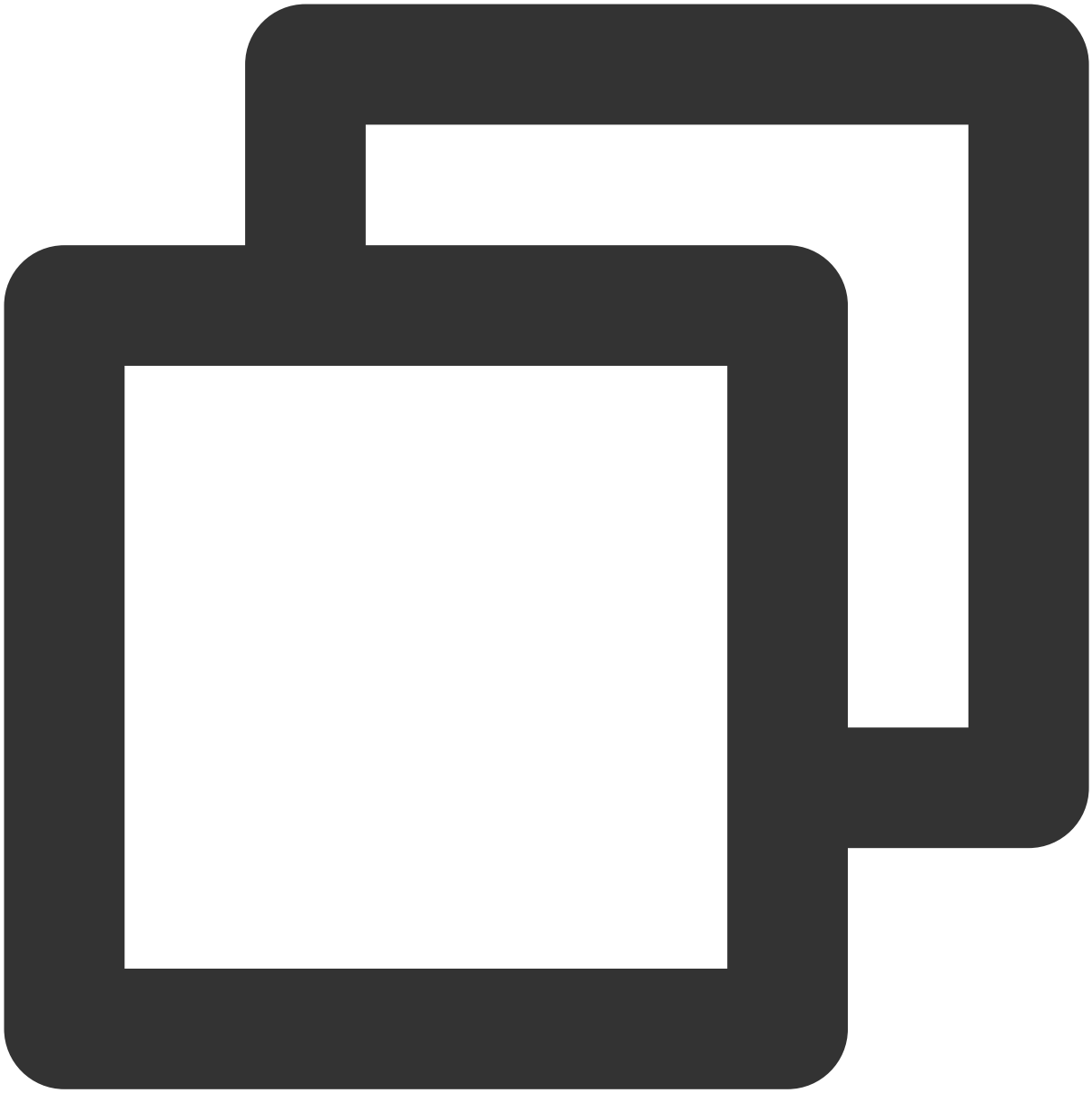
用于提交一个智能封面任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaSmartCoverJobs(array $args = arr
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 提交智能封面任务 https://cloud.tencent.com/document/product/436/54017
    $result = $cosClient->createMediaSmartCoverJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'SmartCover',
        'QueueId' => 'p81e648afxxxxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
//            'TemplateId' => '', // 使用模板
            'SmartCover' => array(
                'Format' => '',
                'Width' => '',
                'Height' => '',
                'Count' => '',
                'DeleteDuplicates' => '',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'SmartCover-${Number}.jpg',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
//        'CallBackFormat' => '',
//        'CallBackType' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选

Tag	Request	创建任务的 Tag：SmartCover	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
TemplateId	Request.Operation	智能封面模板 ID	String	否
SmartCover	Request.Operation	封面配置	Container	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0、1、2。级别越大任务优先级越高，默认为0	String	否

版权所有：腾讯云计算（北京）有限责任公司

说明

如果同时设置了 TemplateId 和 SmartCover，则优先使用 TemplateId。

Container 类型 SmartCover 的具体数据类型描述如下：

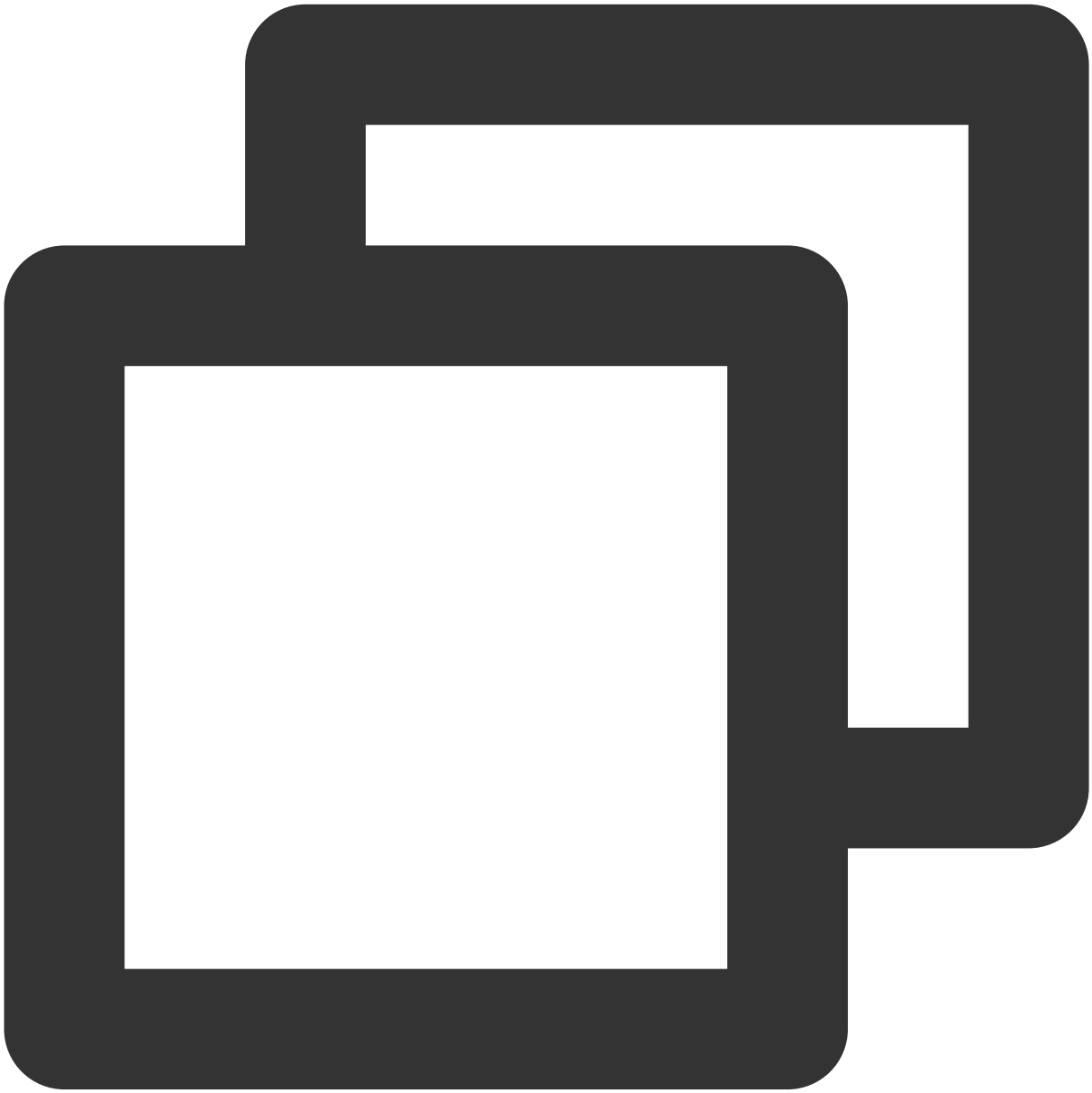
节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Format	Request.Operation.SmartCover	封面图片类型	String	是	无	png、jpg、webp
Width	Request.Operation.SmartCover	封面图片宽度	String	是	无	值范围：[128，4096] 单位：px
Height	Request.Operation.SmartCover	封面图片高度	String	是	无	值范围：[128，4096] 单位：px
Count	Request.Operation.SmartCover	封面数量	String	否	3	值范围：[1，10]
DeleteDuplicates	Request.Operation.SmartCover	封面是否去重	String	否	false	true/false

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名。 **必须包含 ${Number} 参数。** 如 Object 为 my-new-cover-${Number}.jpg，对应实际3张输出结果时，分别为 my-new-cover-0.jpg my-new-cover-1.jpg my-new-cover-2.jpg	String	是

版权所有：腾讯云计算（北京）有限责任公司

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
```

版权所有：腾讯云计算（北京）有限责任公司

```
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
                (
                )

        )

    [RequestId] => NjI2MASOIDUAOIDUJIAOJDIOADJyMjNjZDE=
    [ContentType] => application/xml
    [ContentLength] => 832
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-22T11:12:33+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => j0d9bc17z890zx8c9s877z97187147
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => SmartCover-${Number}.jpg
                                    [Region] => ap-beijing
                                )

                            [SmartCover] => Array
                                (
                                    [Count] =>
                                    [DeleteDuplicates] => false
                                    [Format] =>
                                    [Height] =>
                                    [Width] =>
                                )
                            [UserData] => xxx
                            [JobLevel] => 0
```

```
                        )

                    [QueueId] => p81e648afzx8c09z8xc09zx8c097be086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => SmartCover
                )

        )

)
```

多任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交多任务的 API 概览和 SDK 示例代码。

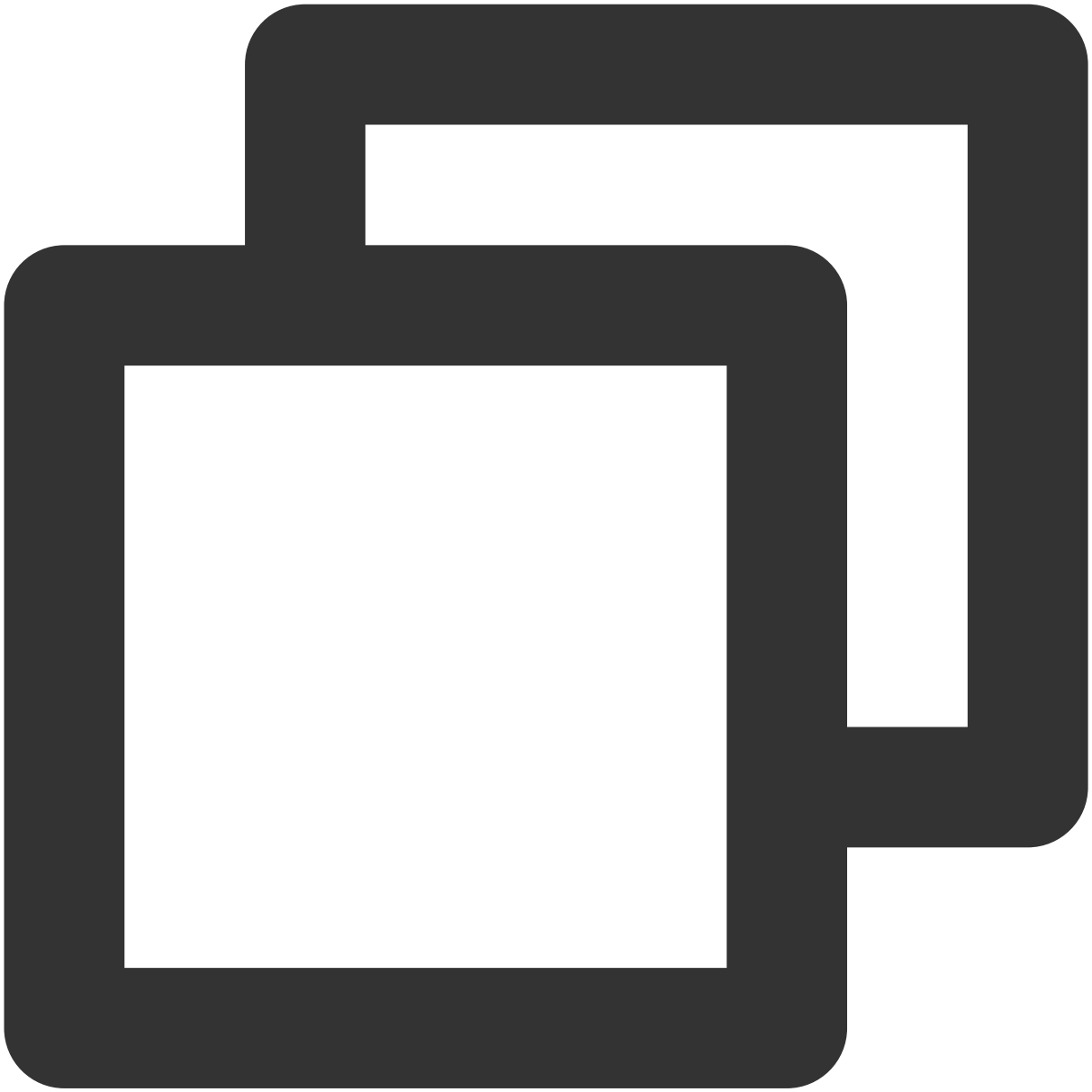
API	操作描述
提交多任务	用于提交多个任务。

提交多任务

功能说明

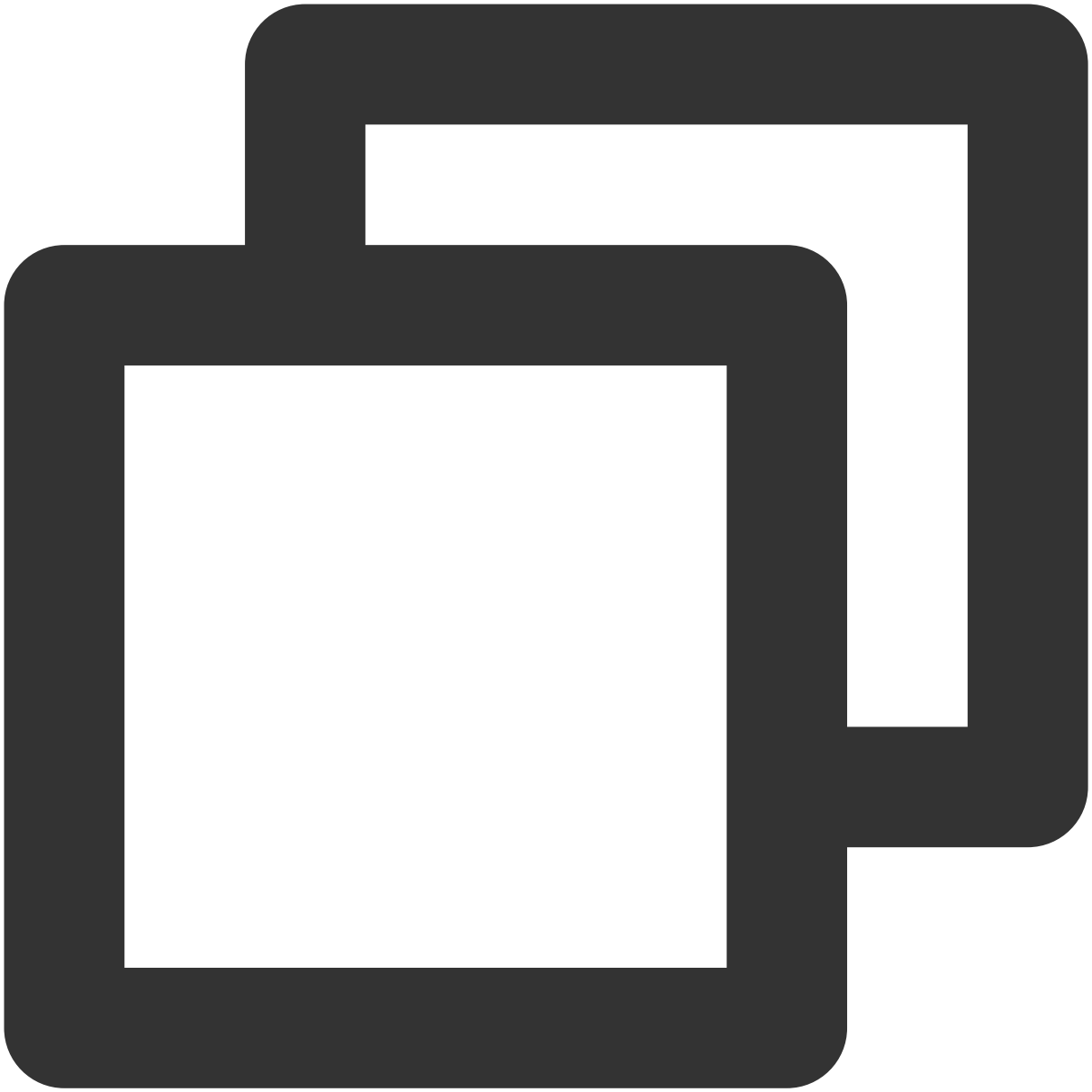
用于提交多个任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaJobs(array $args = array());
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 多任务接口 https://cloud.tencent.com/document/product/436/58335
    $result = $cosClient->CreateMediaJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，
        'Tag' => 'Transcode', // 可选，单一任务时，优先以Operation.Tag为准，当Operation无
        'QueueId' => 'paaf4fce5521a40888a303xxxxxxxxxxxxxx',
        'CallBack' => '',
        'Input' => array(
            'Object' => 'example.mp4'
        ),
        'Operation' => array(
            array(
                'Tag' => 'Transcode',
                'TemplateId' => 't04e1ab86554984f1aa17cxxxxxxxxxxxxx',
                'Output' => array(
                    'Region' => $region,
                    'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName
                    'Object' => 'video01.mp4',
                ),
                'WatermarkTemplateId' => array(
                    't112d18d9b2a9b430e91dxxxxxxxxxxxxx',
                ),
            ),
            array(
                'Tag' => 'Transcode',
                'TemplateId' => 't04e1ab86554984f1aa17xxxxxxxxxxxxx',
                'Output' => array(
                    'Region' => $region,
                    'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName
                    'Object' => 'video02.mp4',
                ),
                'WatermarkTemplateId' => array(
                    't1bf713bb5c6a5496e859axxxxxxxxxxxxx',
                ),
            ),
        ),
    ));
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则，个数不超过6个	Container	是
QueueId	Request	任务所在的队列 ID	String	是

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

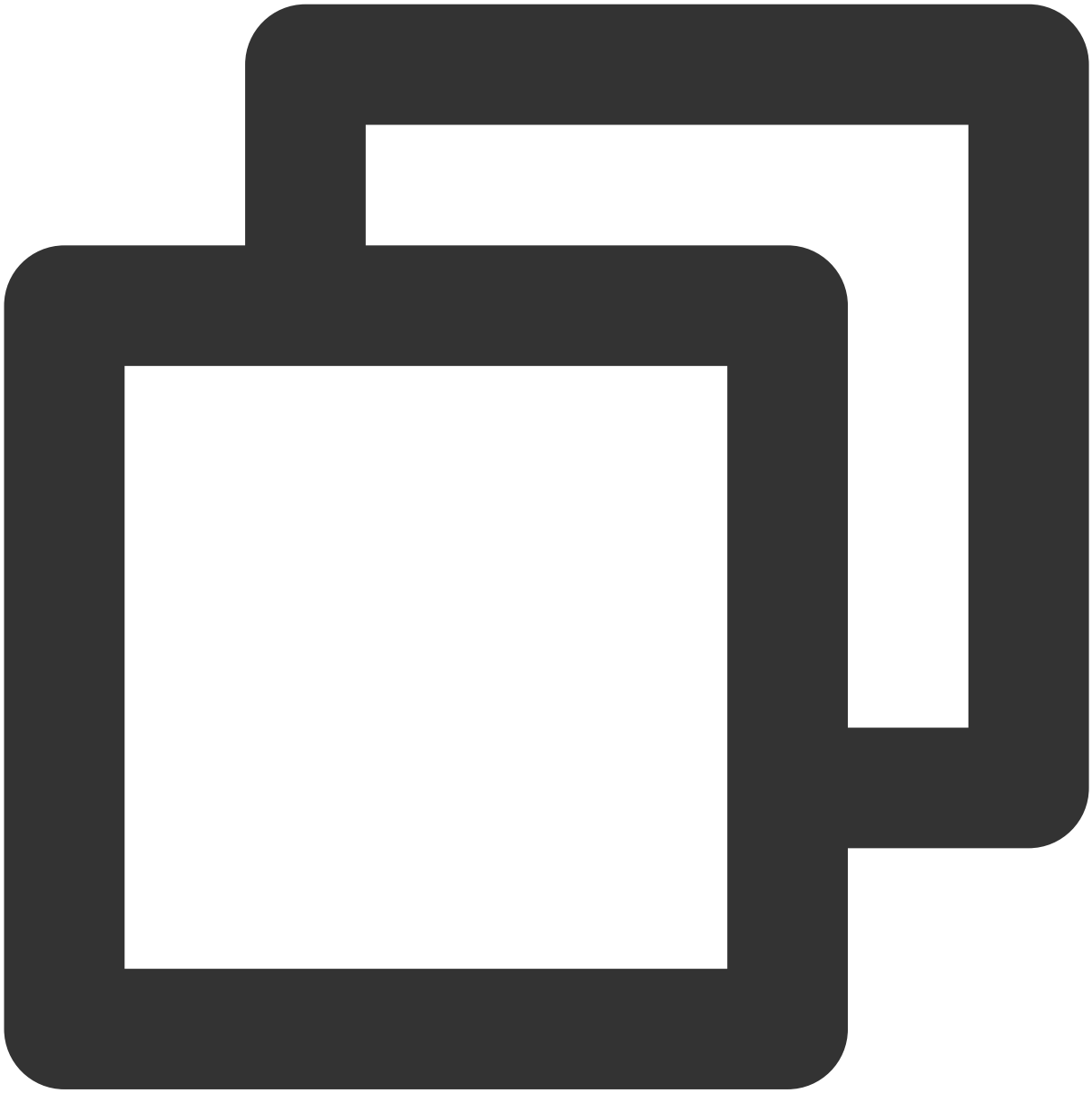
节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request.Operation	任务类型: Animation、Transcode、SmartCover、Snapshot	Container	是
TemplateId	Request.Operation	指定的模板 ID	String	是
Output	Request.Operation	结果输出地址	Container	是

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例

版权所有：腾讯云计算（北京）有限责任公司



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2MTAJOIDSjoiajsdoSDOASDHJoimY2YTQ=
    [ContentType] => application/xml
    [ContentLength] => 1731
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [0] => Array
                        (
                            [Code] => Success
                            [CreationTime] => 2022-04-21T18:03:31+0800
                            [EndTime] => -
                            [Input] => Array
                                (
                                    [BucketId] => examplebucket-1250000000
                                    [Object] => video01.mp4
                                    [Region] => ap-beijing
                                )

                            [JobId] => j4c211cas9a87sd90d80a988627cb30
                            [Message] =>
                            [Operation] => Array
                                (
                                    [Output] => Array
                                        (
                                            [Bucket] => examplebucket-1250000000
                                            [Object] => Transcode-01.flv
                                            [Region] => ap-beijing
                                        )

                                    [TemplateId] => t0b6128z8c098z09xcsd89s0a5dca38
                                    [TemplateName] => FLV-SD
                                    [WatermarkTemplateId] => t185z90x8c9z0da5sd67a5
                                )

                            [Progress] => 0
                            [QueueId] => pzx8c09za90sd890a8d09aa0757be086
                            [StartTime] => -
                            [State] => Submitted
                            [Tag] => Transcode
```

```
                        )

                    [1] => Array
                        (
                            [Code] => Success
                            [CreationTime] => 2022-04-21T18:03:31+0800
                            [EndTime] => -
                            [Input] => Array
                                (
                                    [BucketId] => examplebucket-1250000000
                                    [Object] => video01.mp4
                                    [Region] => ap-beijing
                                )

                            [JobId] => j4cz9x7c8z9xcsda98sd09ad80627cb30
                            [Message] =>
                            [Operation] => Array
                                (
                                    [Output] => Array
                                        (
                                            [Bucket] => examplebucket-1250000000
                                            [Object] => Transcode-02.flv
                                            [Region] => ap-beijing
                                        )

                                    [TemplateId] => t0bz98xc789zxs9d80a8sd90asd37f8
                                    [TemplateName] => FLV-HD
                                    [WatermarkTemplateId] => t18zc8z90csd876a78sdas
                                )

                            [Progress] => 0
                            [QueueId] => p81zxc908zaf2aee496a09s8d90as8d90
                            [StartTime] => -
                            [State] => Submitted
                            [Tag] => Transcode
                        )

                )

        )

)
```

视频增强任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交视频增强任务的 API 概览和 SDK 示例代码。

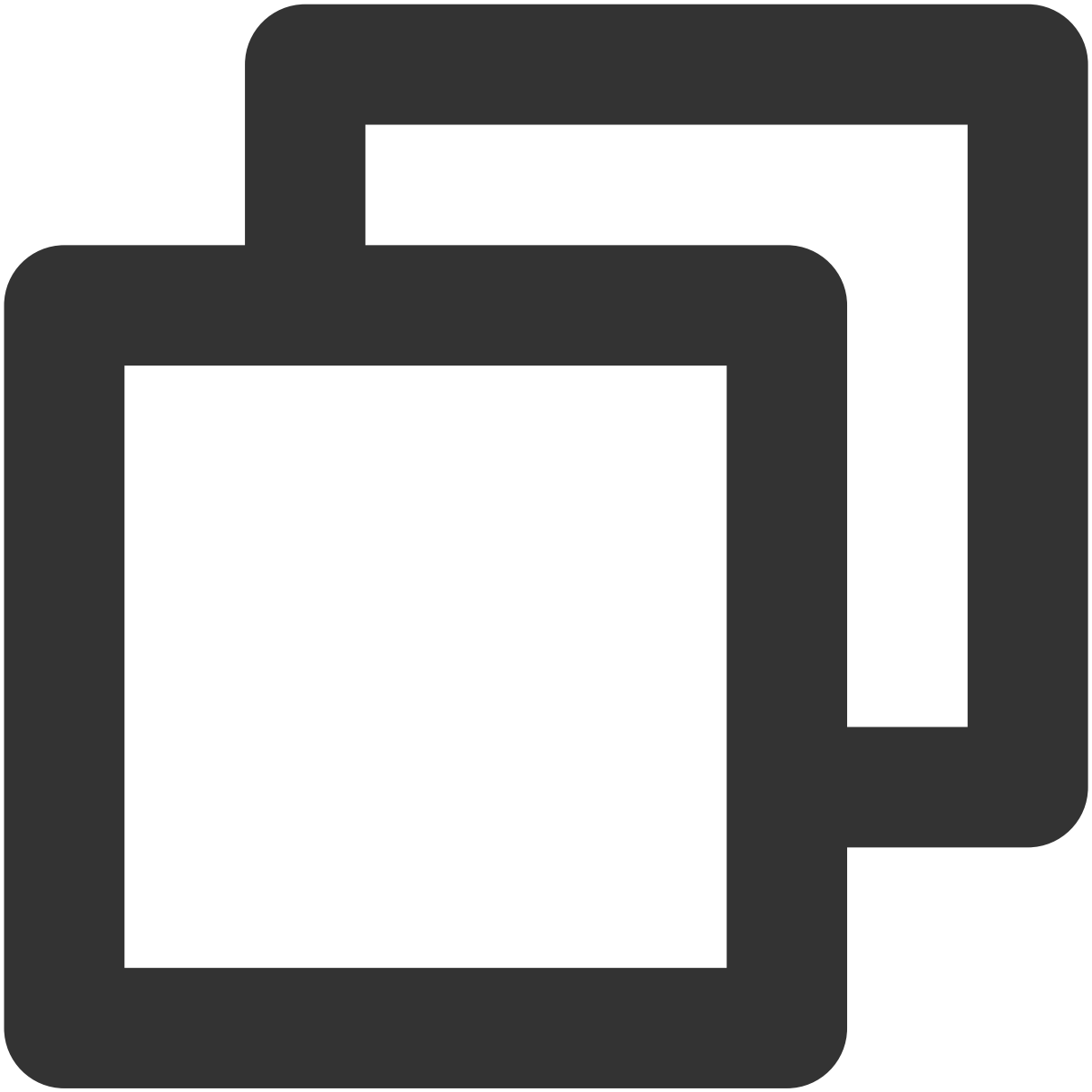
API	操作描述
提交视频增强任务	提交一个视频增强任务

提交视频增强任务

功能说明

用于提交一个视频增强任务。

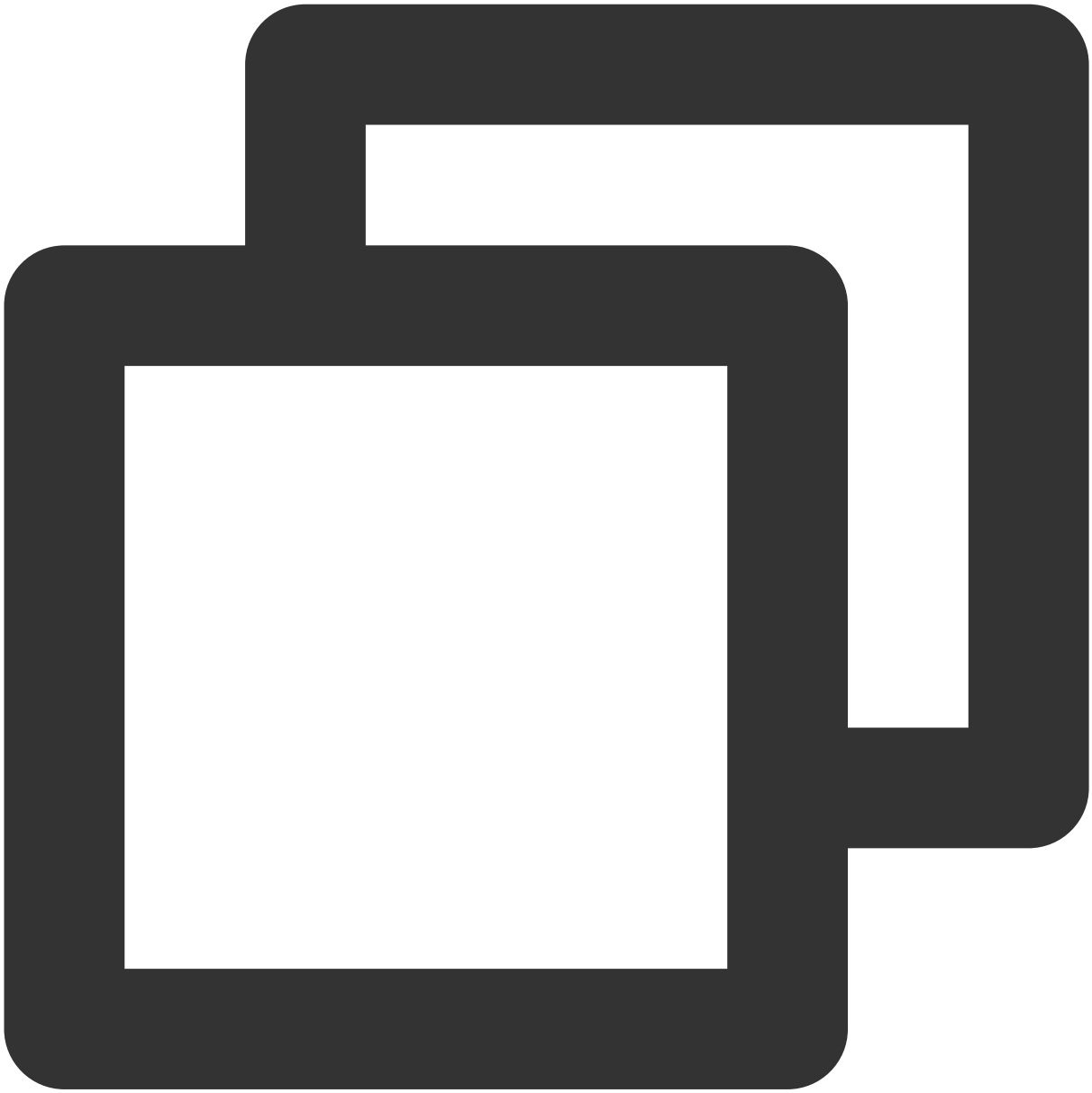
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaVideoProcessJobs(array $args = a
```

请求示例

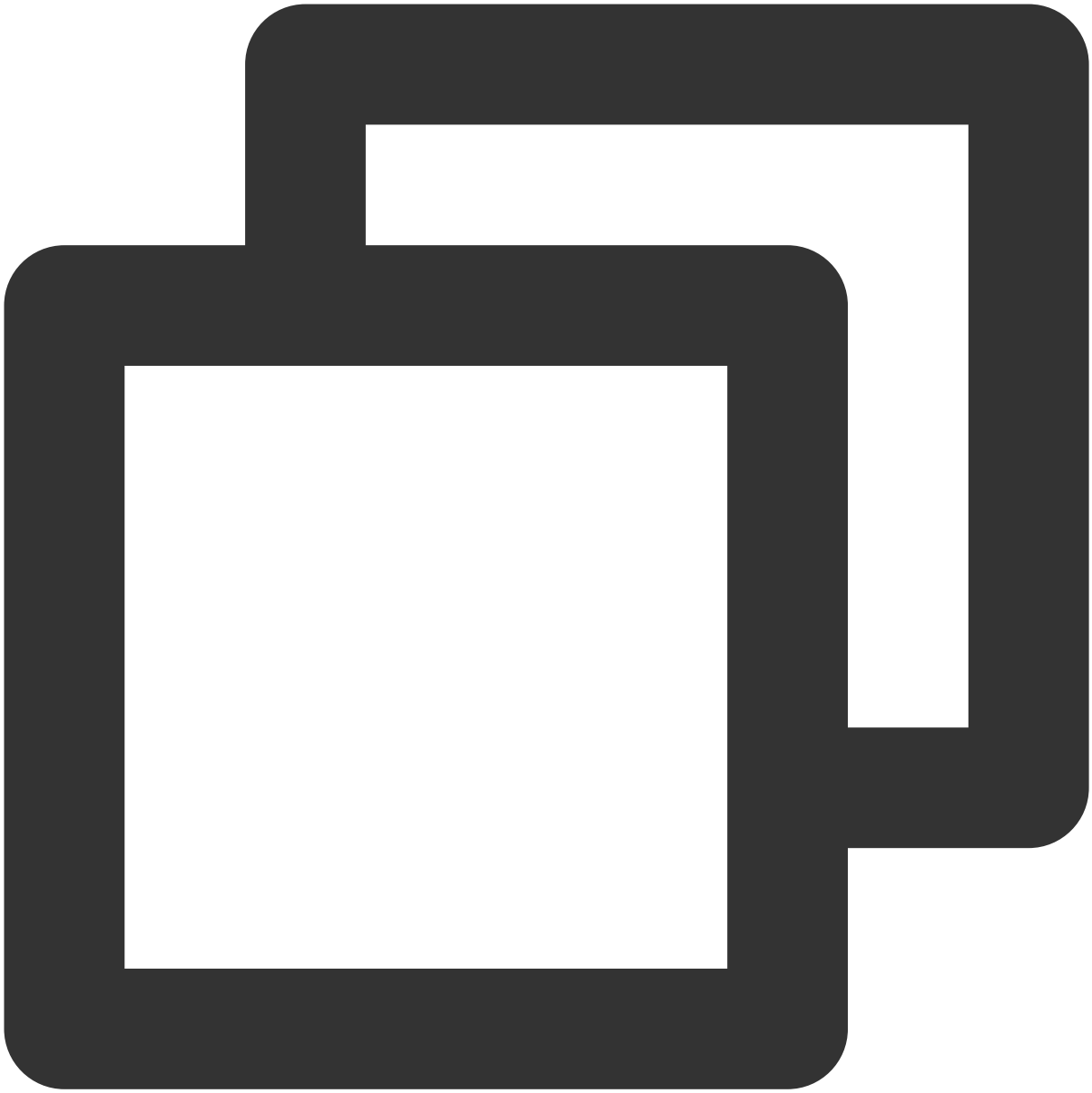
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVideoProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VideoProcess',
        'QueueId' => 'p81e648afxxxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 't13466f1ea41a14c0xxxxxxxxxxxxx', // 视频增强模板 ID
            'TranscodeTemplateId' => 't0b6a845f5e42847bd81xxxxxxxxxxxx', // 转码模板
            'WatermarkTemplateId' => 't185e2e24551b24259a0xxxxxxxxxxxx', // 水印模板
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VideoProcess.flv',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVideoProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VideoProcess',
        'QueueId' => 'p81e648afxxxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'VideoProcess' => array(
                'ColorEnhance' => array(
                    'Enable' => '',
                    'Contrast' => '',
                    'Correction' => '',
                    'Saturation' => '',
                ),
                'MsSharpen' => array(
                    'Enable' => '',
                    'SharpenLevel' => '',
                ),
            ),
            'Transcode' => array(
                'Tag' => '',
                'Name' => '',
                'Container' => array(
                    'Format' => '',
                ),
                'Video' => array(
                    'Codec' => '',
                    'Width' => '',
                    'Height' => '',
                    'Fps' => '',
                    'Remove' => '',
                    'Profile' => '',
                    'Bitrate' => '',
                    'Crf' => '',
                    'Gop' => '',
                    'Preset' => '',
                    'Bufsize' => '',
                    'Maxrate' => '',
                    'HlsTsTime' => '',
                    'Pixfmt' => '',
                    'LongShortMode' => '',
                ),
```

```
                'TimeInterval' => array(
                    'Start' => '',
                    'Duration' => '',
                ),
                'Audio' => array(
                    'Codec' => '',
                    'Samplerate' => '',
                    'Bitrate' => '',
                    'Channels' => '',
                    'Remove' => '',
                    'KeepTwoTracks' => '',
                    'SwitchTrack' => '',
                    'SampleFormat' => '',
                ),
                'TransConfig' => array(
                    'AdjDarMethod' => '',
                    'IsCheckReso' => '',
                    'ResoAdjMethod' => '',
                    'IsCheckVideoBitrate' => '',
                    'VideoBitrateAdjMethod' => '',
                    'IsCheckAudioBitrate' => '',
                    'AudioBitrateAdjMethod' => '',
                    'DeleteMetadata' => '',
                    'IsHdr2Sdr' => '',
                    'HlsEncrypt' => array(
                        'IsHlsEncrypt' => '',
                        'UriKey' => '',
                    ),
                ),
            ),
            'Watermark' => array(
                'Type' => '',
                'Pos' => '',
                'LocMode' => '',
                'Dx' => '',
                'Dy' => '',
                'StartTime' => '',
                'EndTime' => '',
                'Image' => array(
                    'Url' => '',
                    'Mode' => '',
                    'Width' => '',
                    'Height' => '',
                    'Transparency' => '',
                    'Background' => '',
                ),
                'Text' => array(
```

```
                    'FontSize' => '',
                    'FontType' => '',
                    'FontColor' => '',
                    'Transparency' => '',
                    'Text' => '',
                ),
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VideoProcess.flv',
            ),
//              'UserData' => 'xxx', // 透传用户信息
//              'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：VideoProcess	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为	String	否

		no 时，表示队列的回调地址不产生回调		
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
VideoProcess	Request.Operation	指定视频增强模板参数，不能与 TemplateId 同时为空	Container	否
TemplateId	Request.Operation	指定的模板 ID，不能与 VideoProcess 同时为空	String	否
Transcode	Request.Operation	指定转码模板参数，不能与 TranscodeTemplateId 同时为空	Container	否
TranscodeTemplateId	Request.Operation	指定的转码模板 ID，优先使用模板 ID，不能与 Transcode 同时为空	String 数组	否
Watermark	Request.Operation	指定水印模板参数，同创建水印模板 CreateMediaTemplate 接口的 Request.Watermark，最多传3个	Container 数组	否
WatermarkTemplateId	Request.Operation	指定的水印模板 ID，可以传多个水印模板 ID，最多传3个，优先使用模板 ID	String	否
DigitalWatermark	Request.Operation	指定数字水印参数	Container	否
Output	Request.Operation	结果输出地址	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0、1、2。级别越大任务优先级越高，默认为0	String	否

说明：

提交视频增强任务必须传入转码参数。对于视频增强参数，优先使用 TemplateId，无 TemplateId 时使用 VideoProcess；对于转码参数，优先使用 TranscodeTemplateId，无 TranscodeTemplateId 时使用 Transcode；对于水印参数，可以使用 WatermarkTemplateId 或 Watermark 设置，WatermarkTemplateId 优先级更高。

Container 类型 VideoProcess 的具体数据描述如下：

节点名称（关键字）	父节点	描述
ColorEnhance	Request.Operation.VideoProcess	同创建视频增强模板 CreateMediaTemplate 接口中的 Request.ColorEnhance
MsSharpen	Request.Operation.VideoProcess	同创建视频增强模板 CreateMediaTemplate 接口中的 Request.MsSharpen

Container 类型 Transcode 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
TimeInterval	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否
Container	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
Audio	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Audio	Container	否
TransConfig	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TransConfig	Container	否
AudioMix	Request.Operation.Transcode	混音参数，详情见 AudioMix	Container 数组	否

Container 类型 DigitalWatermark 的具体数据描述如下：

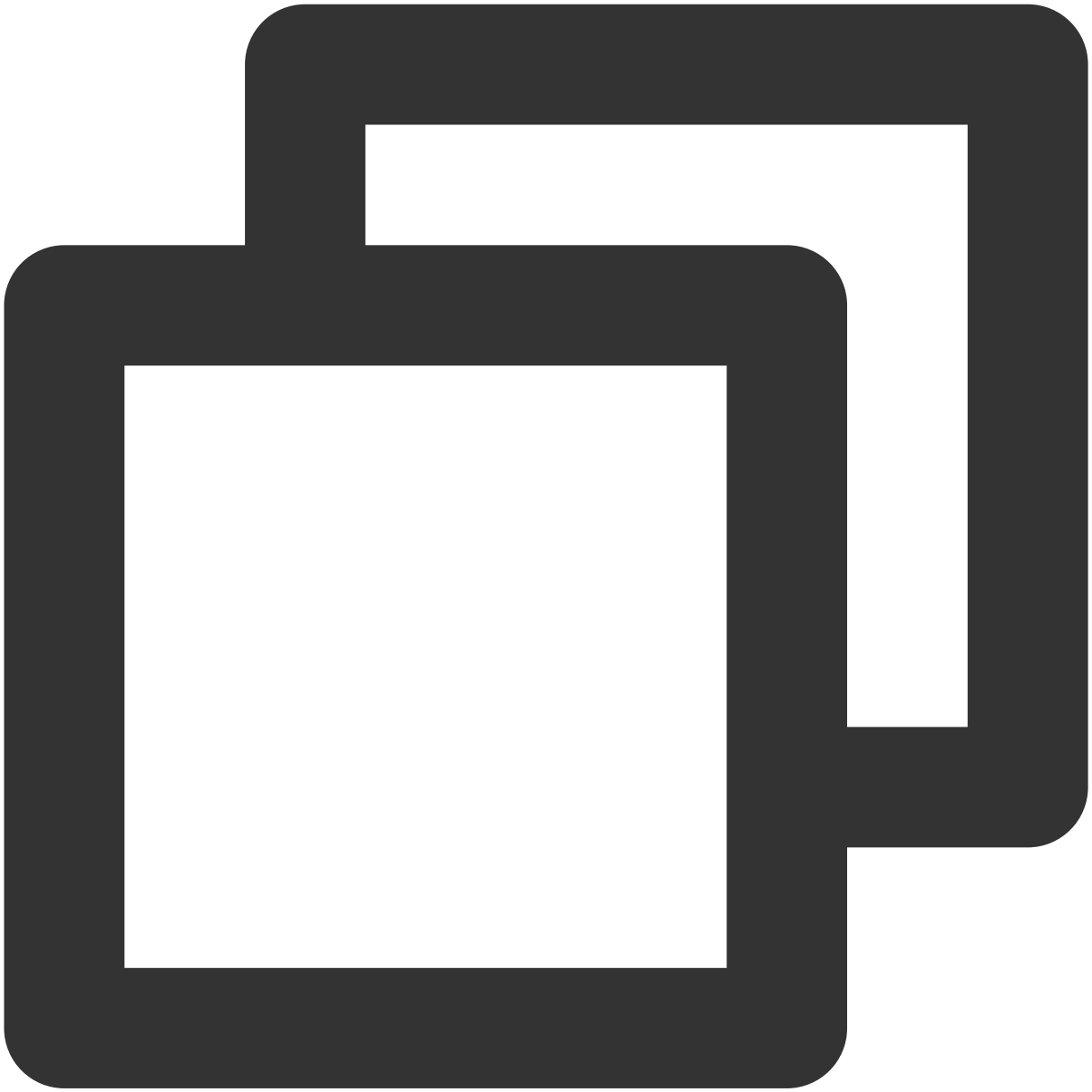
节点名称（关键字）	父节点	描述	类型	必选
Message	Request.Operation.DigitalWatermark	数字水印嵌入的字符串信息，长度不超过64个字符，仅支持中文、英文、数字、_、-和*	string	是
Type	Request.Operation.DigitalWatermark	水印类型，当前仅可设置为 Text	String	是

Version	Request.Operation.DigitalWatermark	水印版本，当前仅可设置为 V1	String	是
IgnoreError	Request.Operation.DigitalWatermark	当添加水印失败是否忽略错误继续执行任务，限制为 true/false，默认为 false	string	是

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2MjNjYHAOIHDOIADJHOIA18yMjFlMWE=
    [ContentType] => application/xml
    [ContentLength] => 965
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-22T11:19:15+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => jfd27519z8zx90c8z90c7809z9f35eb
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => VideoProcess.flv
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t13466f1eazx08c09zx8c09zx8e9cd
                            [TemplateName] => video-process-1
                            [TranscodeTemplateId] => t0b612860z087xc09zx8c09xz8c095
                            [WatermarkTemplateId] => t185e2e24zxc780z8c0z98c323c642
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2aee496885zx098c09z8xc09zxc086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => VideoProcess
```

```
                    )

                )

)
```

精彩集锦任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交精彩集锦任务的 API 概览和 SDK 示例代码。

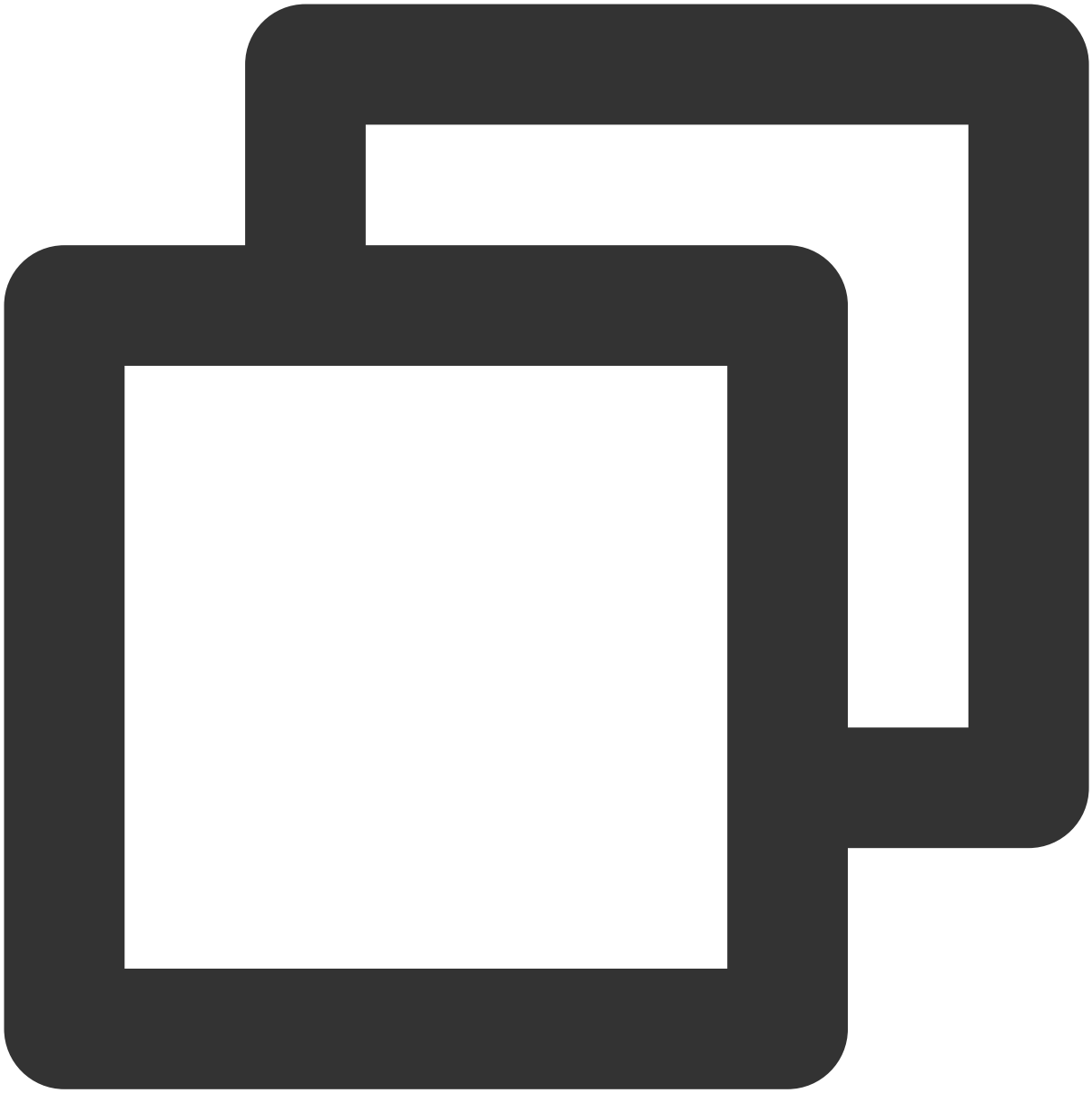
API	操作描述
提交精彩集锦任务	提交一个精彩集锦任务

提交精彩集锦任务

功能说明

用于提交一个精彩集锦任务。

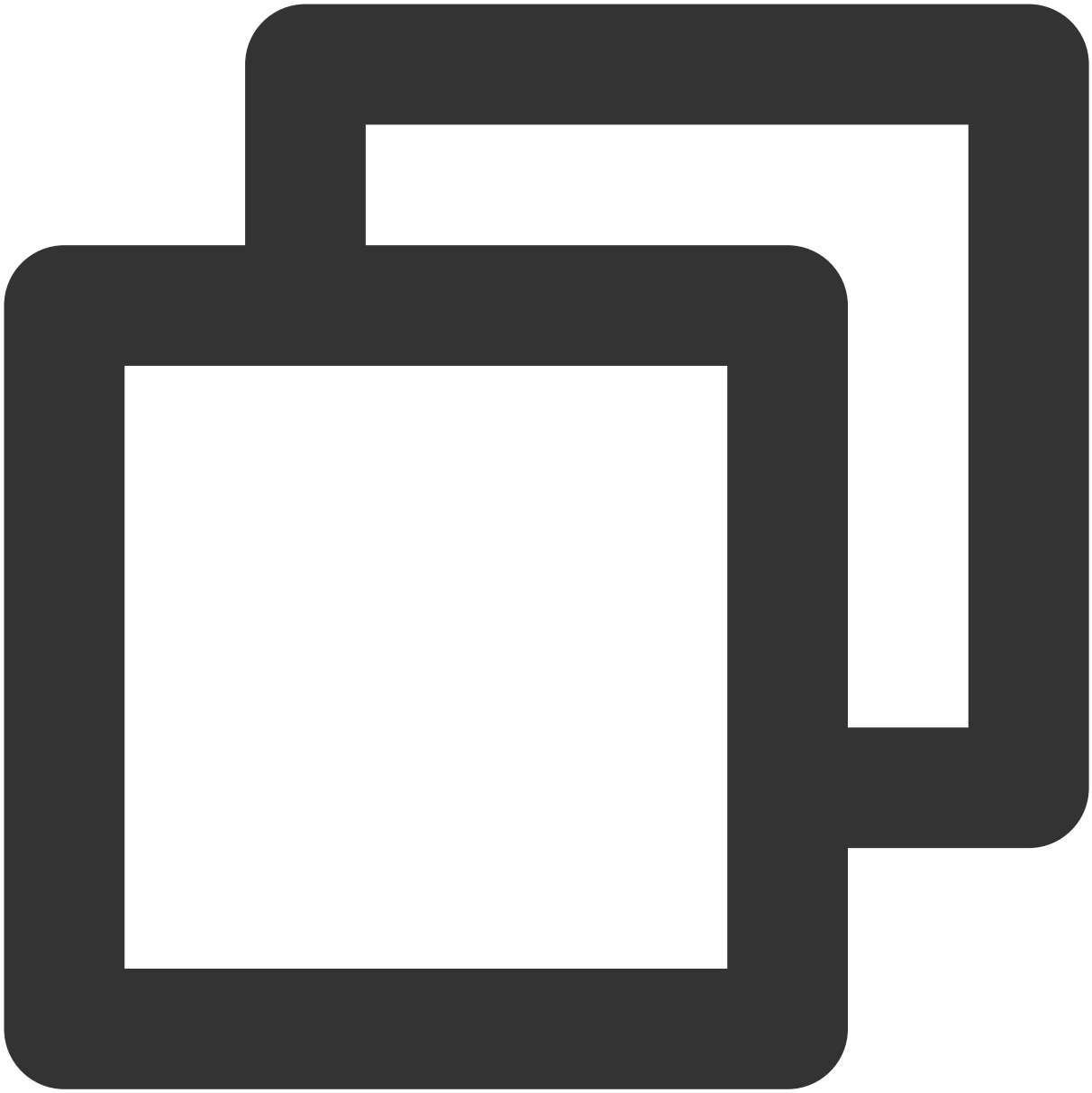
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaVideoMontageJobs(array $args = a
```

请求示例

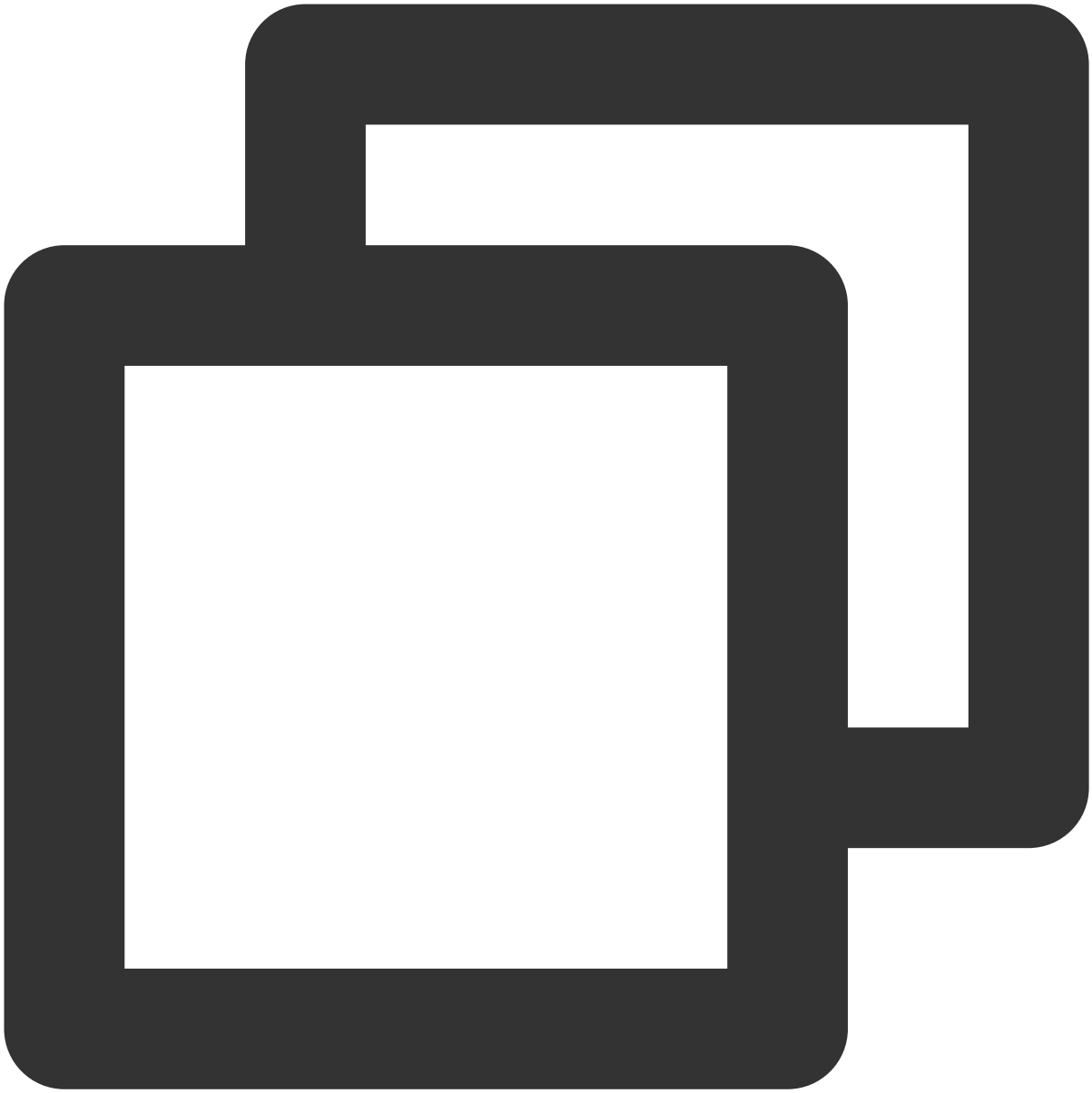
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVideoMontageJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VideoMontage',
        'QueueId' => 'p81e648af2aeexxxxxxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' => 't1fcc3770199e04737axxxxxxxxxxxxxx',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VideoMontage.mp4',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVideoMontageJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VideoMontage',
        'QueueId' => 'p81e648af2aeexxxxxxxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'VideoMontage' => array(
                'Duration' => '',
                'Container' => array(
                    'Format' => '',
                ),
                'Video' => array(
                    'Codec' => '',
                    'Width' => '',
                    'Height' => '',
                    'Fps' => '',
                    'Bitrate' => '',
                    'Crf' => '',
                ),
                'Audio' => array(
                    'Codec' => '',
                    'Samplerate' => '',
                    'Bitrate' => '',
                    'Channels' => '',
                    'Remove' => '',
                ),
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VideoMontage.mp4',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
```

```
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：VideoMontage	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
VideoMontage	Request.Operation	指定精彩集锦模板参数	Container	否

版权所有：腾讯云计算（北京）有限责任公司

TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过 1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

注意：

优先使用 TemplateId，无 TemplateId 时使用 VideoMontage。

Container 类型 VideoMontage 的具体数据描述如下：

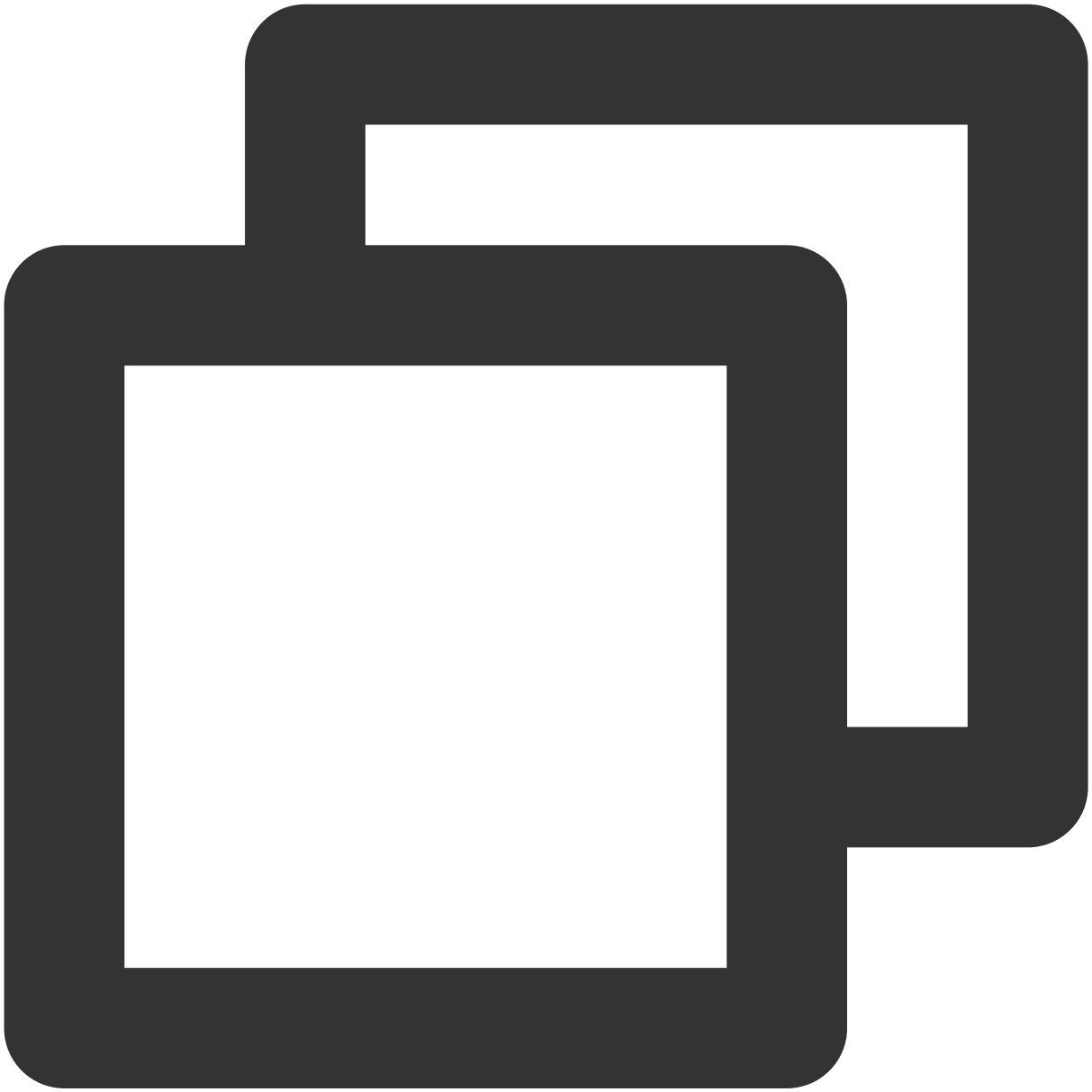
节点名称（关键字）	父节点	描述	类型	必选
TimeInterval	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否
Container	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
Audio	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Audio	Container	否
TransConfig	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TransConfig	Container	否
AudioMix	Request.Operation.Transcode	混音参数，详情见 AudioMix	Container 数组	否

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例

版权所有：腾讯云计算（北京）有限责任公司



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2MjIxNDVfHAUOSDHAOIHDOIyMmZkOTQ=
    [ContentType] => application/xml
    [ContentLength] => 803
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-22T11:30:13+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-beijing
                        )

                    [JobId] => j85803a96zx7c98z7xc987z9c6eb232
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => VideoMontage.mp4
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t1fcc3770199e0zx7cz897c9zxc9b
                            [TemplateName] => video-montage-1
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2aezx7c9z7xc98zx7c98e086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => VideoMontage
                )
```

```
            )

)
```

版权所有：腾讯云计算（北京）有限责任公司

人声分离任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于提交人声分离任务的 API 概览和 SDK 示例代码。

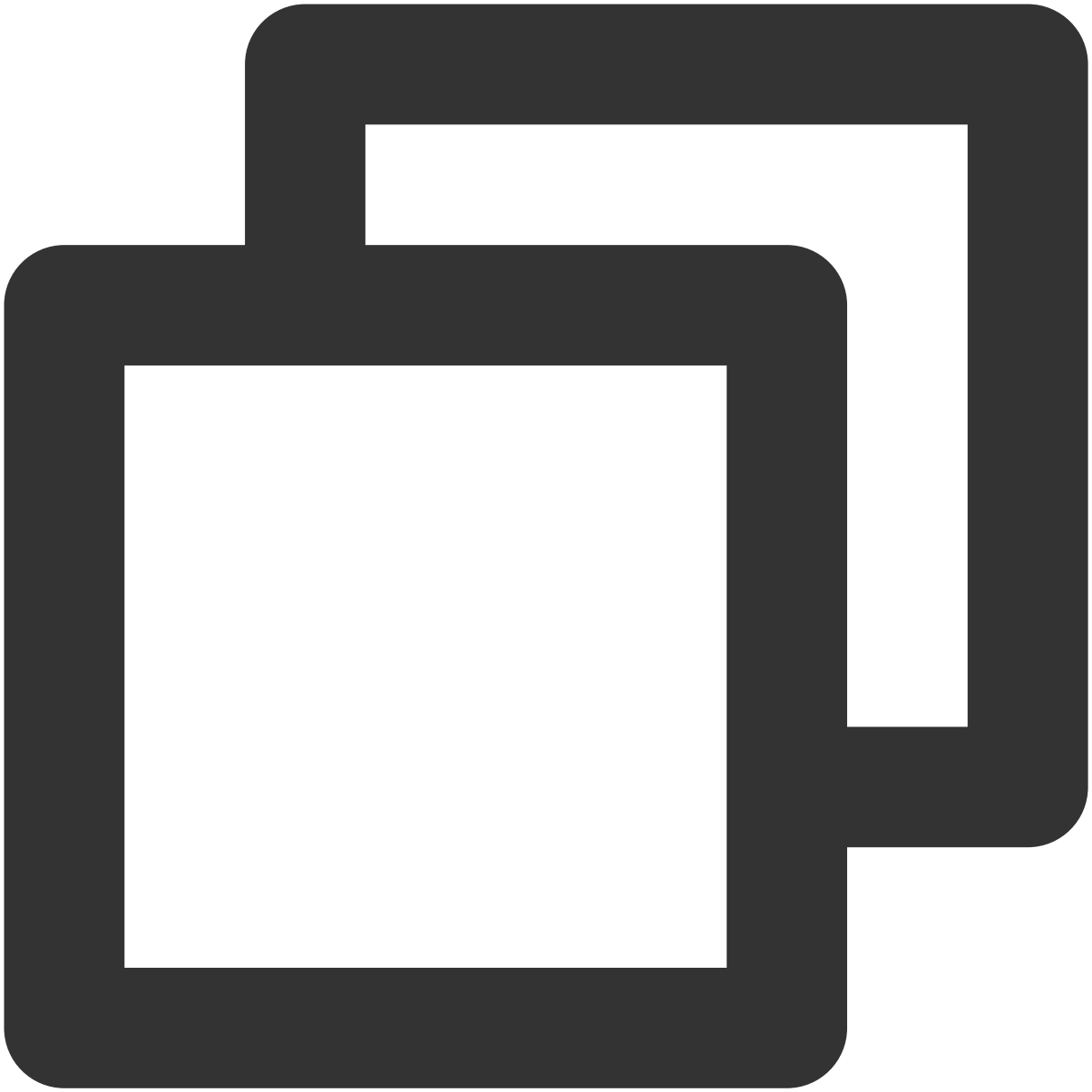
API	操作描述
提交人声分离任务	提交一个人声分离任务

提交人声分离任务

功能说明

用于提交一个人声分离任务。

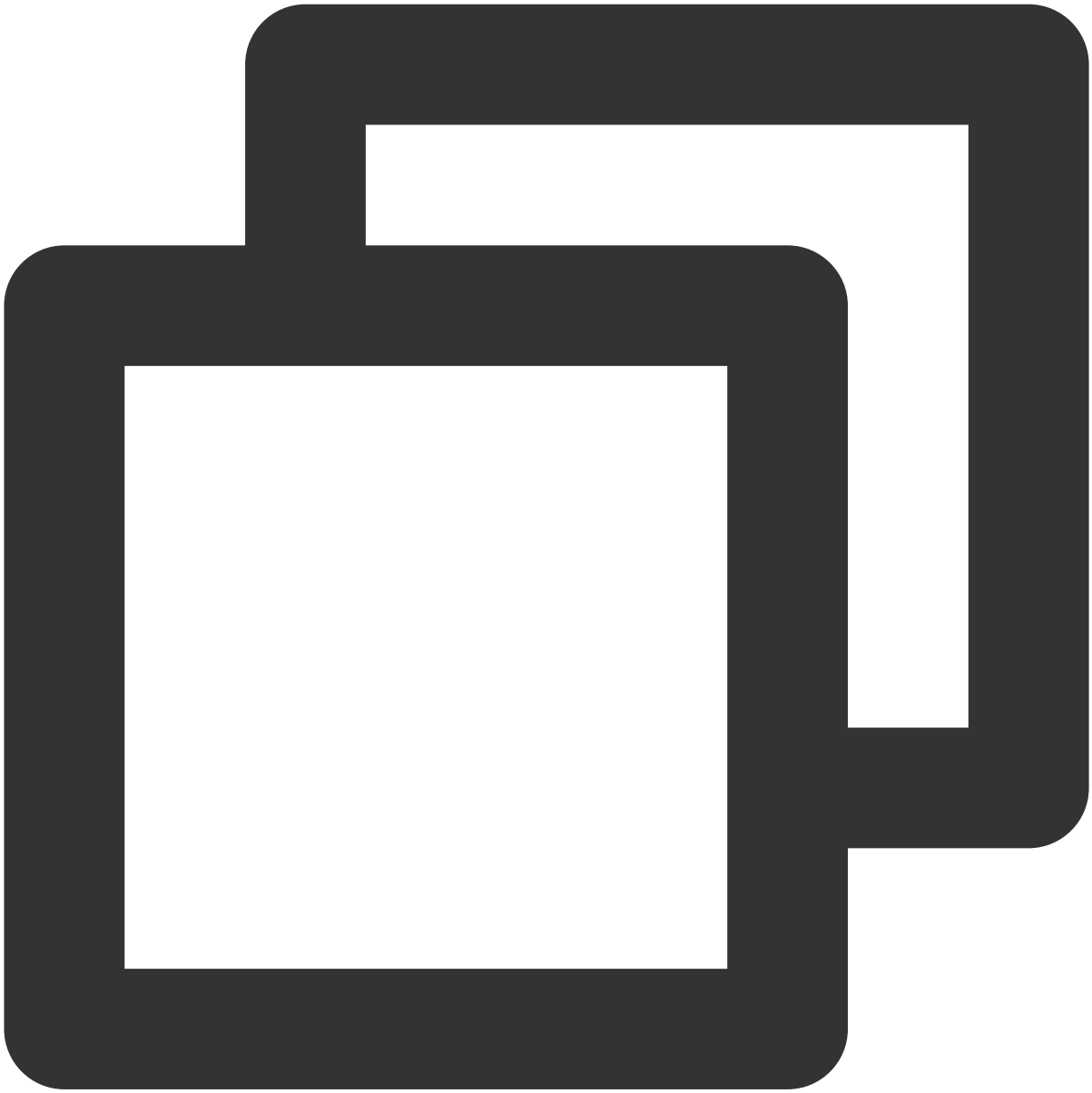
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaVoiceSeparateJobs(array $args =
```

请求示例

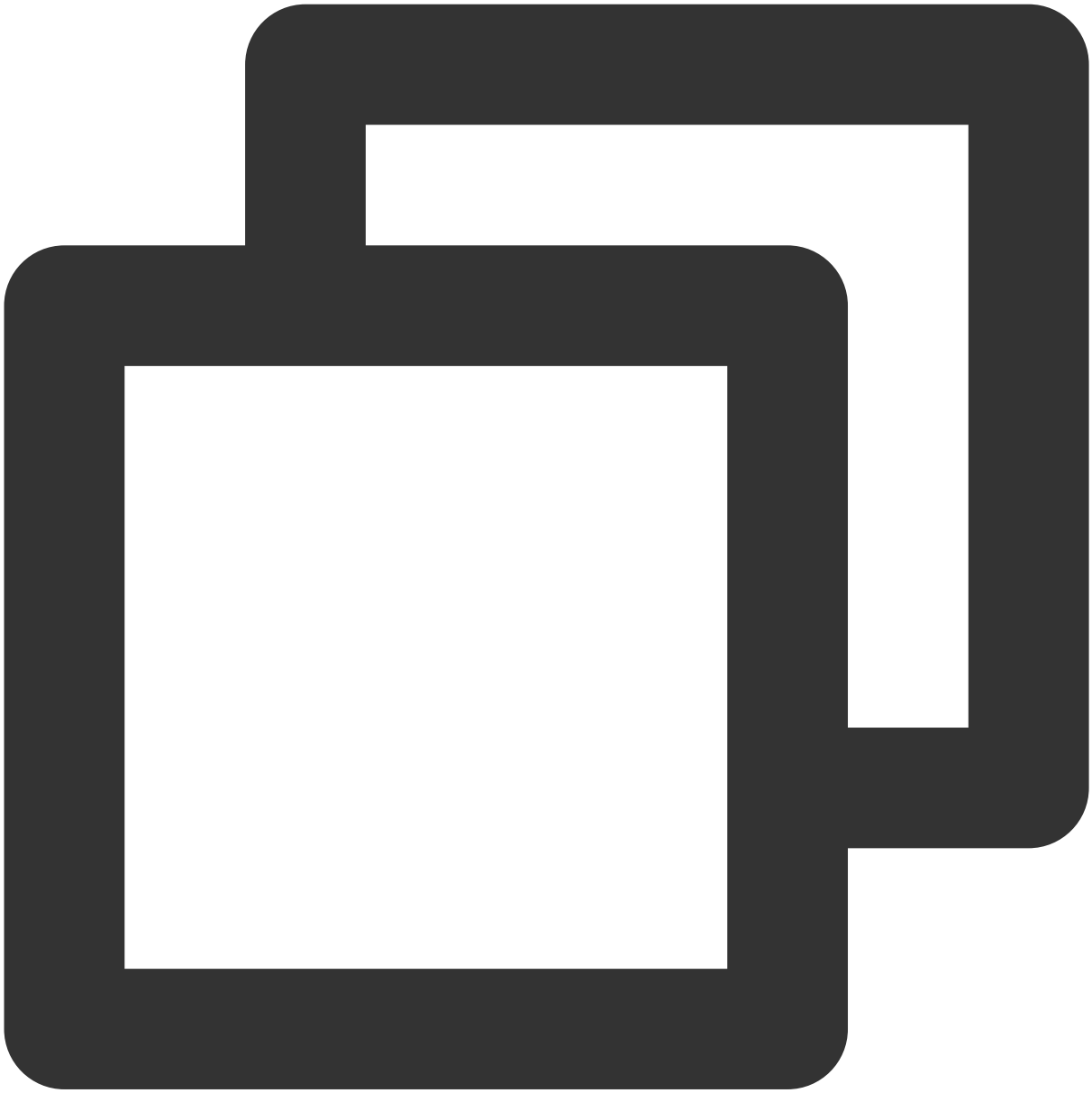
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVoiceSeparateJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VoiceSeparate',
        'QueueId' => '',
        'CallBack' => '',
        'Input' => array(
            'Object' => 'test.mp3'
        ),
        'Operation' => array(
            'TemplateId' => '',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VoiceSeparate01.mp3',
                'AuObject' => 'VoiceSeparate02.mp3',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaVoiceSeparateJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VoiceSeparate',
        'QueueId' => '',
        'CallBack' => '',
        'Input' => array(
            'Object' => 'test.mp3'
        ),
        'Operation' => array(
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'VoiceSeparate01.mp3',
                'AuObject' => 'VoiceSeparate02.mp3',
            ),
            'VoiceSeparate' => array(
                'AudioMode' => 'AudioAndBackground',
                'AudioConfig' => array(
                    'Codec' => 'mp3',
                    'Samplerate' => '11025',
                    'Bitrate' => '256',
                    'Channels' => '2',
                ),
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选

Tag	Request	创建任务的 Tag：VoiceSeparate	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
VoiceSeparate	Request.Operation	指定转码模板参数	Container	否
TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0、1、2。级别越大任务优先级越高，默认为0	String	否

注意：

优先使用 TemplateId，无 TemplateId 时使用 VoiceSeparate。

Container 类型 VoiceSeparate 的具体数据描述如下：

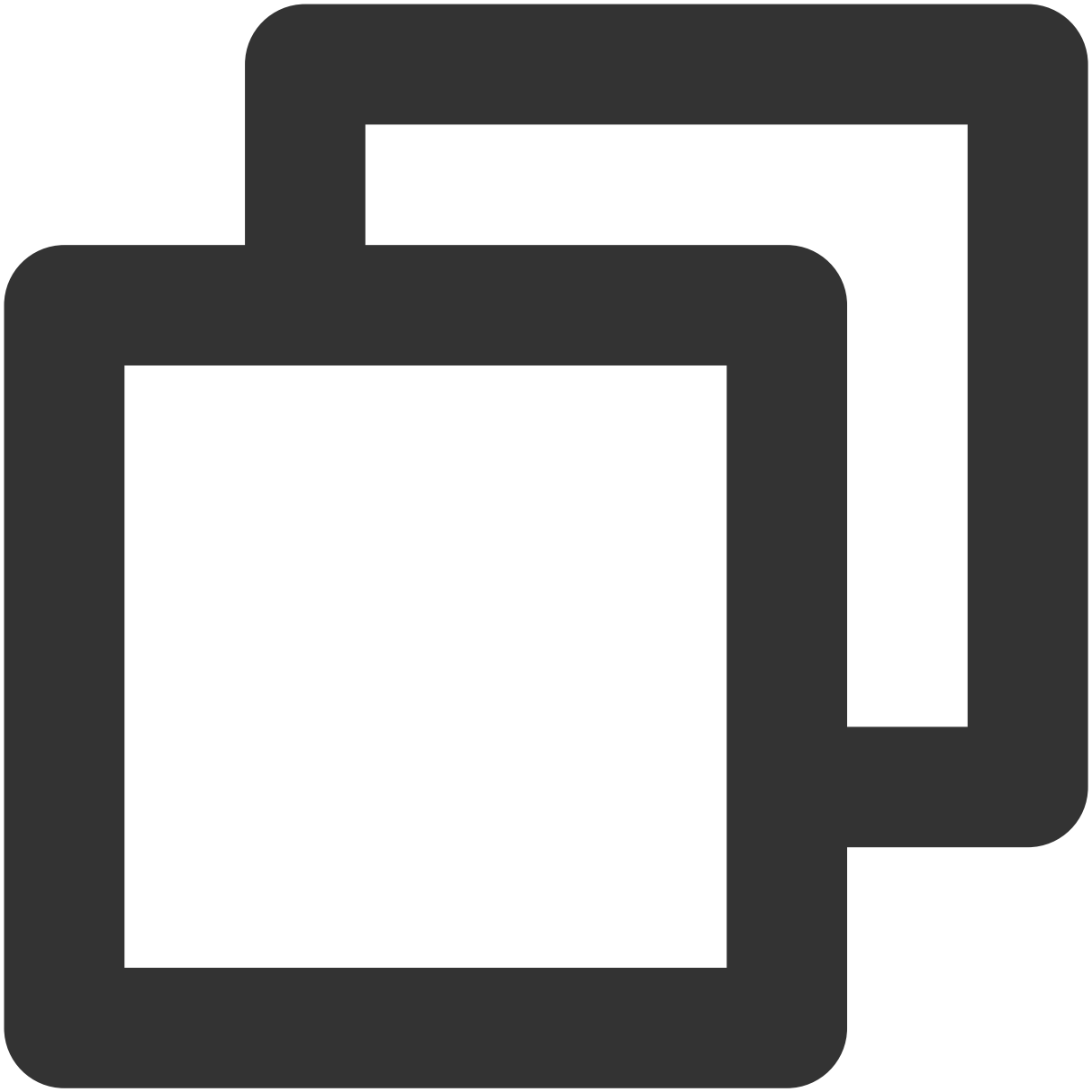
节点名称（关键字）	父节点	描述	类型	是否必选
AudioMode	Request.Operation.VoiceSeparate	同创建人声分离模板 CreateMediaTemplate 接口中的 Request.AudioMode	Container	否
AudioConfig	Request.Operation.VoiceSeparate	同创建人声分离模板 CreateMediaTemplate 接口中的 Request.AudioConfig	Container	否

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	背景音结果文件名，不能与 AuObject 同时为空	String	否
AuObject	Request.Operation.AuObject	人声结果文件名，不能与 Object 同时为空	String	否

返回结果示例

版权所有：腾讯云计算（北京）有限责任公司



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2MjIyOTHADOHDOADJHOjQ0OV8yMmU0OWM=
    [ContentType] => application/xml
    [ContentLength] => 857
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-beijing.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-22T11:35:47+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => test.mp3
                            [Region] => ap-beijing
                        )

                    [JobId] => j4c70446zxc780z98xc09zxc6eb232
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [AuObject] => VoiceSeparate02.mp3
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => VoiceSeparate01.mp3
                                    [Region] => ap-beijing
                                )

                            [TemplateId] => t1456ea89fczx8c0z8c09z8c0985
                            [TemplateName] => VoiceSeparate-1
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2azc709zx8c09z8xc0z7be086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => VoiceSeparate
                )
```

```
            )

)
```

版权所有：腾讯云计算（北京）有限责任公司

SDR to HDR 任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于SDR to HDR 任务接口的 API 概览和 SDK 示例代码。

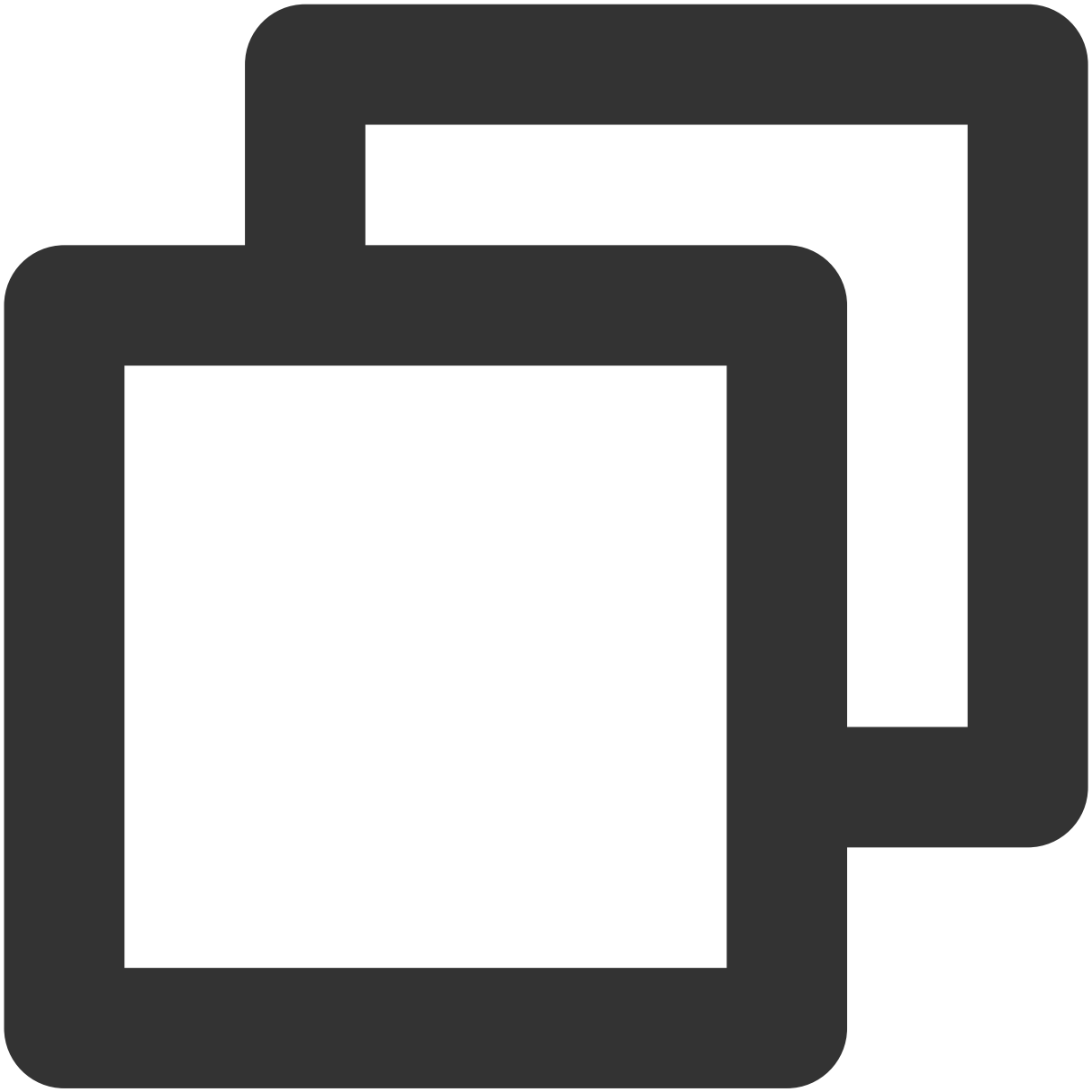
API	操作描述
提交 SDR to HDR 任务	提交 SDR to HDR 任务

提交 SDR to HDR 任务

功能说明

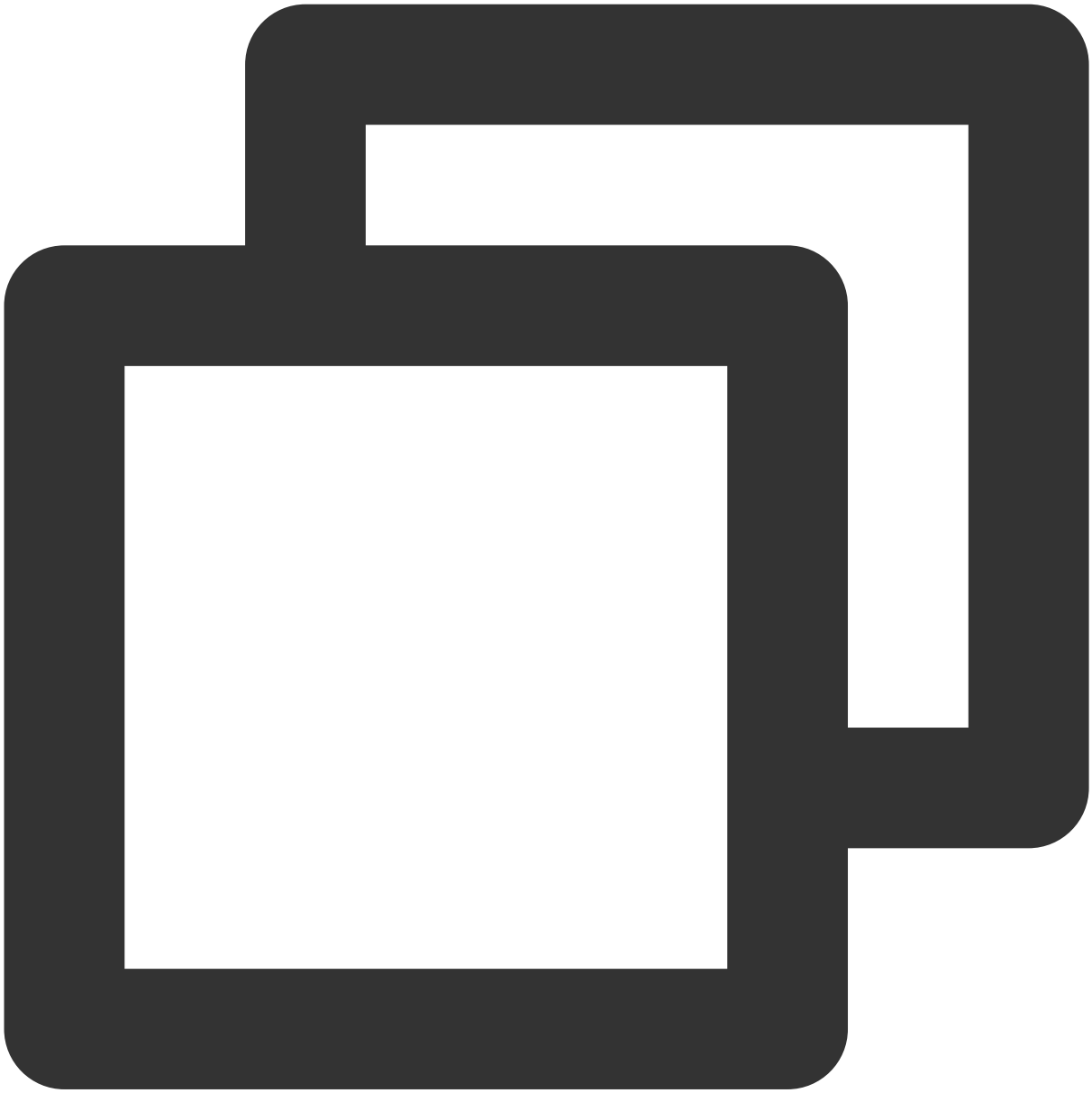
提交 SDR to HDR 任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaSDRtoHDRJobs(array $args = array
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 提交 SDR to HDR 任务 https://www.tencentcloud.com/document/product/436/49052
    $result = $cosClient->createMediaSDRtoHDRJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'SDRtoHDR',
        'QueueId' => 'p81e648af2aee496885a8d09a8s09d8a0sd6',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TranscodeTemplateId' => '',
            'WatermarkTemplateId' => '',
            'SDRtoHDR' => array(
                'HdrMode' => 'HLG',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'SDRtoHDR.flv',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：SDRtoHDR	String	是
Input	Request	待操作的媒体信息	Container	是

Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
SDRtoHDR	Request.Operation	指定 SDRtoHDR 参数	Container	是
Transcode	Request.Operation	指定转码模板参数，不能与 TranscodeTemplateId 同时为空	Container	否
TranscodeTemplateId	Request.Operation	指定的转码模板 ID，优先使用模板 ID，不能与 Transcode 同时为空	String	否
Watermark	Request.Operation	指定水印模板参数，同创建水印模板 CreateMediaTemplate 接口中的 Request.Watermark, 最多传3个	Container 数组	否
WatermarkTemplateId	Request.Operation	指定的水印模板 ID，可以传多个水印模板 ID，最多传3个，优先使用模板 ID	String 数组	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度	String	否

版权所有：腾讯云计算（北京）有限责任公司

		不超过1024		
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

说明：

提交 SDRtoHDR 任务必须传入转码参数。对于转码参数，优先使用 TranscodeTemplateId，无 TranscodeTemplateId 时使用 Transcode；对于水印参数，可以使用 WatermarkTemplateId 或Watermark 设置，WatermarkTemplateId 优先级更高。

Container 类型 SDRtoHDR 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
HdrMode	Request.Operation.SDRtoHDR	HDR 标准	String	是	1. HLG 2. HDR10

Container 类型 Transcode 的具体数据描述如下：

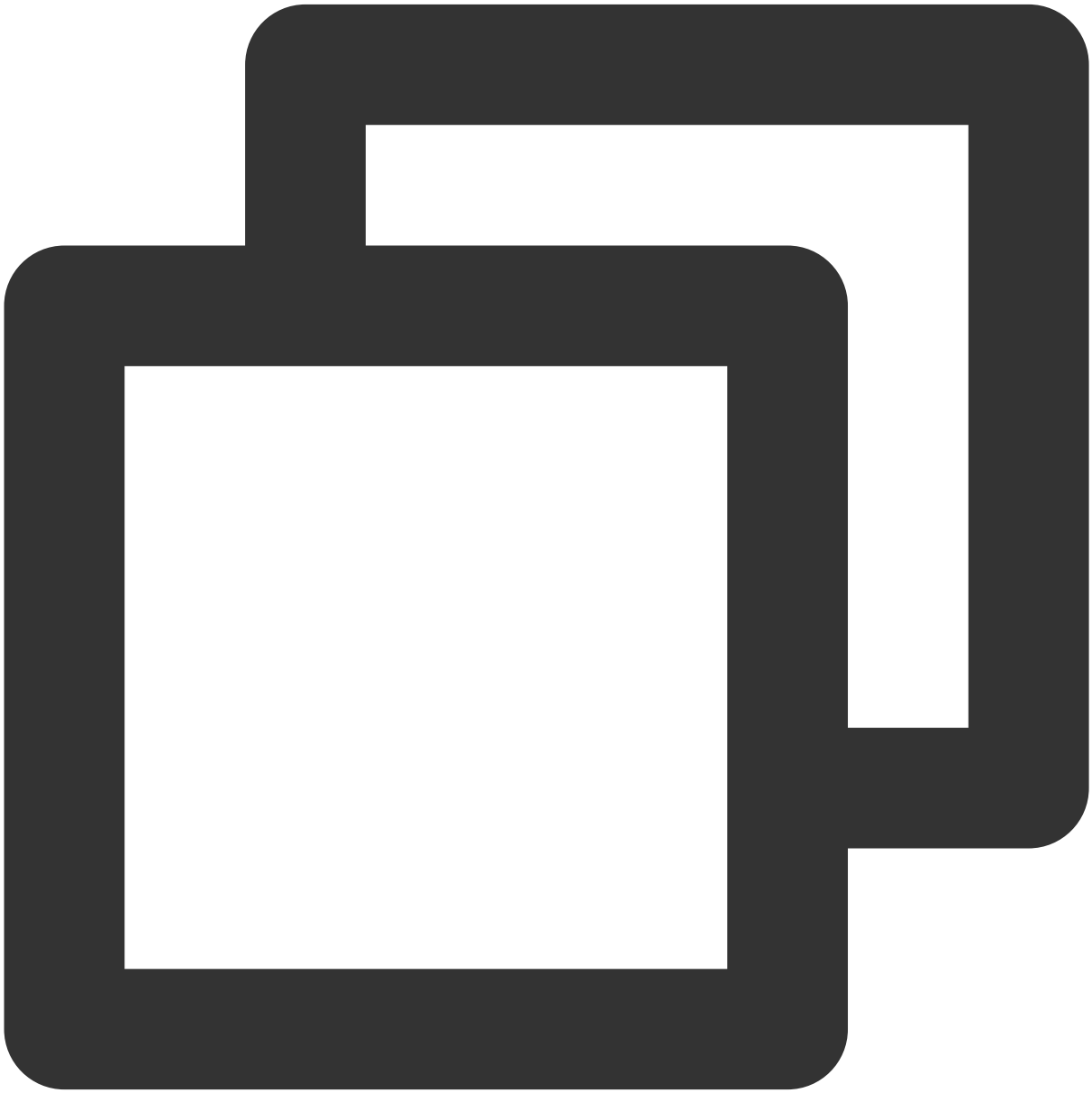
节点名称（关键字）	父节点	描述	类型	必选
TimeInterval	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否
Container	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
Audio	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Audio	Container	否
TransConfig	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TransConfig	Container	否
AudioMix	Request.Operation.Transcode	混音参数, 详情见AudioMix	Container 数组	否

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是

Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
```

```
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
                (
                )

        )

    [RequestId] => NjI2N2I1NWFfZmNjYTNHDOASJDOIA1Yw==
    [ContentType] => application/xml
    [ContentLength] => 902
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-26T17:03:22+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => jb9289626c53f11ec8a9c4f3d8d099dcb
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => SDRtoHDR.flv
                                    [Region] => ap-guangzhou
                                )

                            [SDRtoHDR] => Array
                                (
                                    [HdrMode] => HLG
                                )

                            [TranscodeTemplateId] => t0b612860a293f410785ba7s8d09a8
```

```
                            [WatermarkTemplateId] => t185e2e24551b242d09a80d8a0d804
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => t185e2e24551b242d09a80d8a0d80428a19c
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => SDRtoHDR
                )

        )

)
```

数字水印任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于数字水印任务接口的 API 概览和 SDK 示例代码。

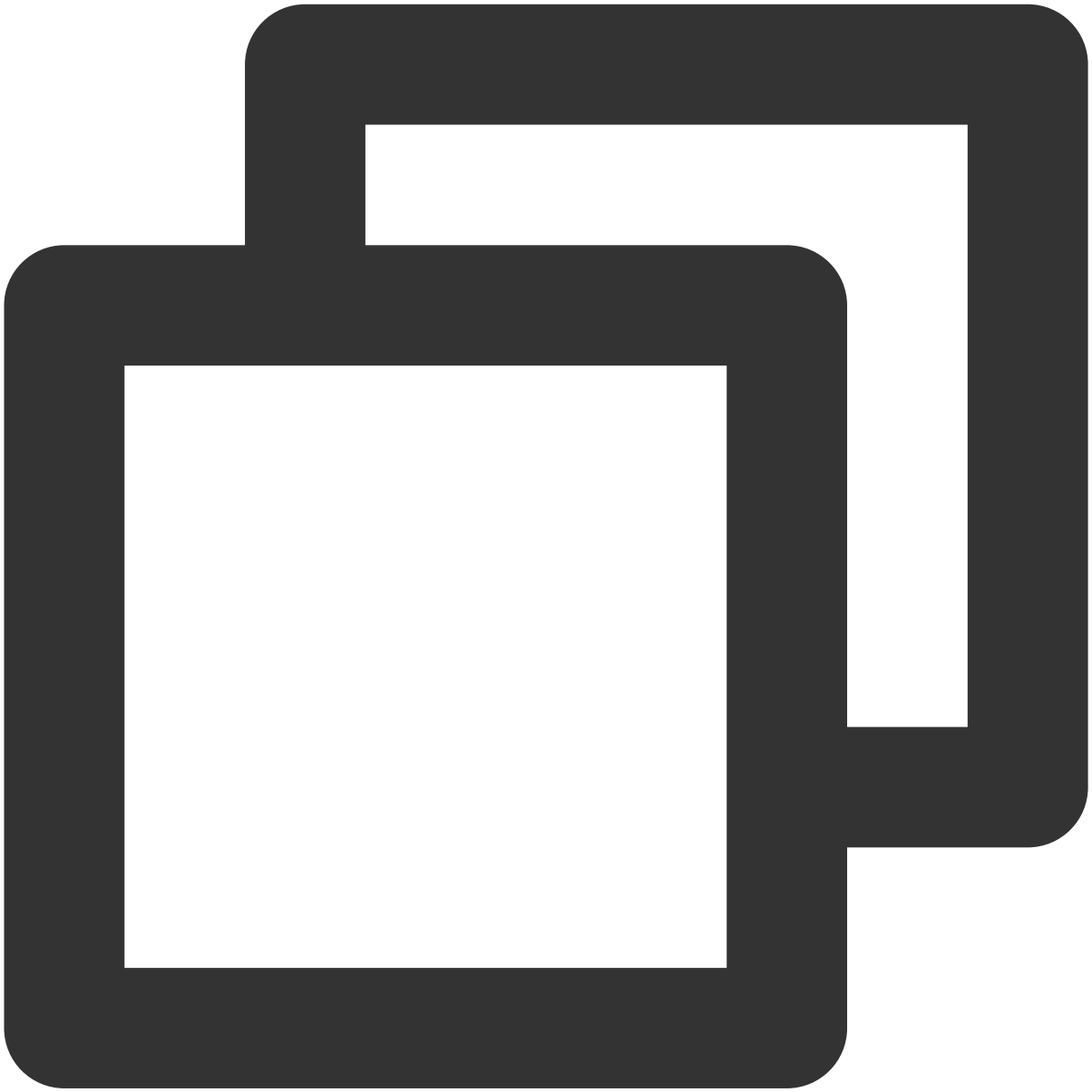
API	操作描述
添加数字水印任务	提交一个嵌入数字水印任务。
提取数字水印任务	提交一个提取数字水印任务。

添加数字水印任务

功能说明

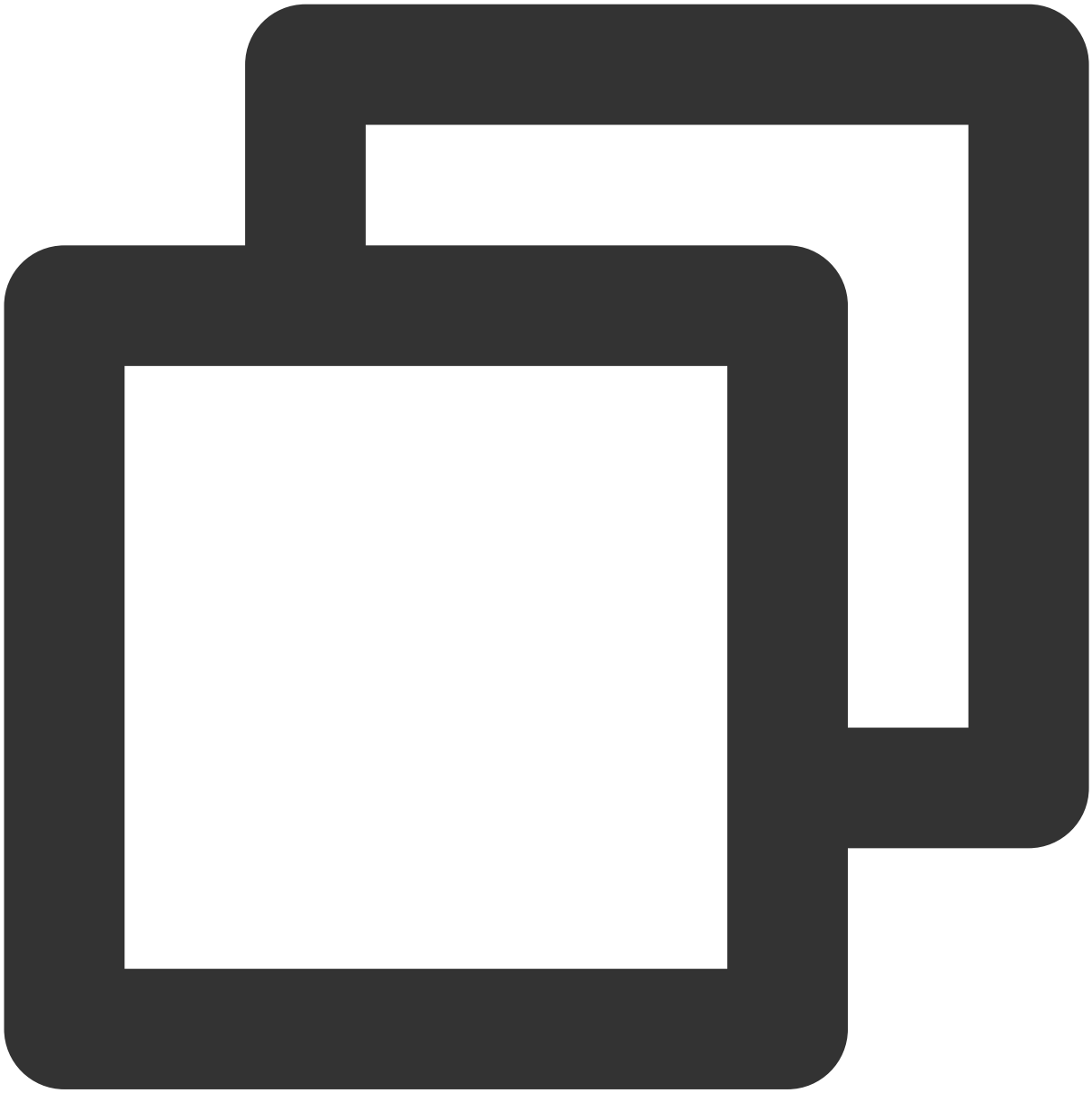
提交一个嵌入数字水印任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaDigitalWatermarkJobs(array $args
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 嵌入数字水印任务 https://www.tencentcloud.com/document/product/436/49047
    $result = $cosClient->createMediaDigitalWatermarkJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'DigitalWatermark',
        'QueueId' => 'p81e648af2aee496885707caxxxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'DigitalWatermark' => array(
                'Message' => '123456789ab',
                'Type' => 'Text',
                'Version' => 'V1',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'DigitalWatermark.mp4',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：DigitalWatermark	String	是
Input	Request	待操作的媒体信息	Container	是

Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Output	Request.Operation	结果输出地址	Container	是
DigitalWatermark	Request.Operation	数字水印配置	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否
UserData	Request.Operation	透传用户信息，可打印的 ASCII 码, 长度不超过1024	String	否

Container 类型 DigitalWatermark 的具体数据类型描述如下：

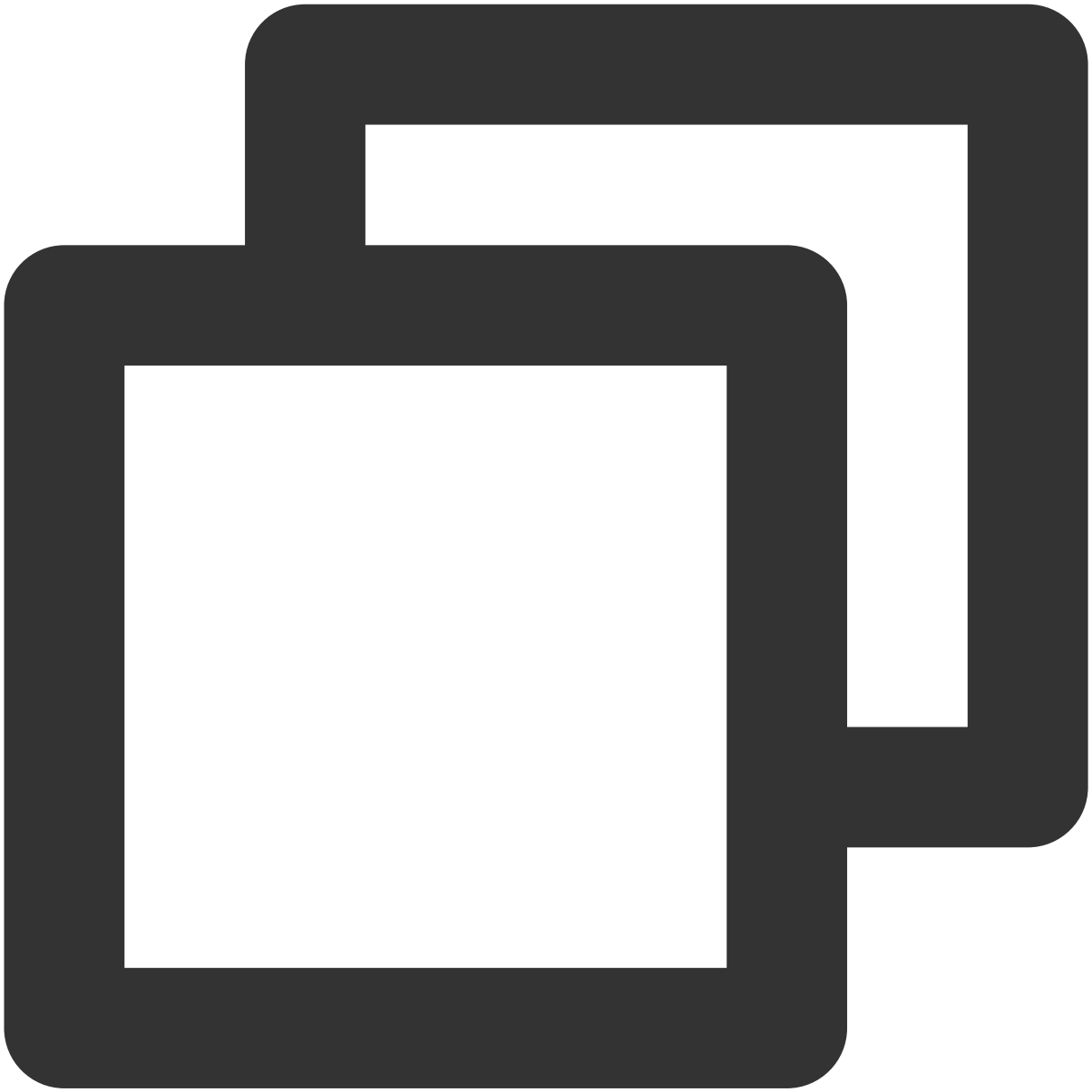
节点名称（关键字）	父节点	描述	类型	是否必选	限制
Message	Request.Operation.DigitalWatermark	数字水印嵌	string	是	长度不超过64个字

		入的字符串信息			符，仅支持中文、英文、数字、_、-和*
Type	Request.Operation.DigitalWatermark	水印类型	String	是	Text
Version	Request.Operation.DigitalWatermark	水印版本	String	是	V1

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

```
                (
                )

        )

    [RequestId] => NjI2N2JlMDFfZmNjYAJSDIOAJDOIAJD9lZDYxOA==
    [ContentType] => application/xml
    [ContentLength] => 828
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-26T17:40:17+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => je1dc03f0c544as09d8as09d8a09dec25a93
                    [Message] =>
                    [Operation] => Array
                        (
                            [DigitalWatermark] => Array
                                (
                                    [Message] => 123456789ab
                                    [Type] => Text
                                    [Version] => V1
                                )

                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => DigitalWatermark.mp4
                                    [Region] => ap-guangzhou
                                )
                            [UserData] => xxx
                            [JobLevel] => 0

                        )

                    [QueueId] => p81e648af2aee49688570s8a90d80a9d86
```

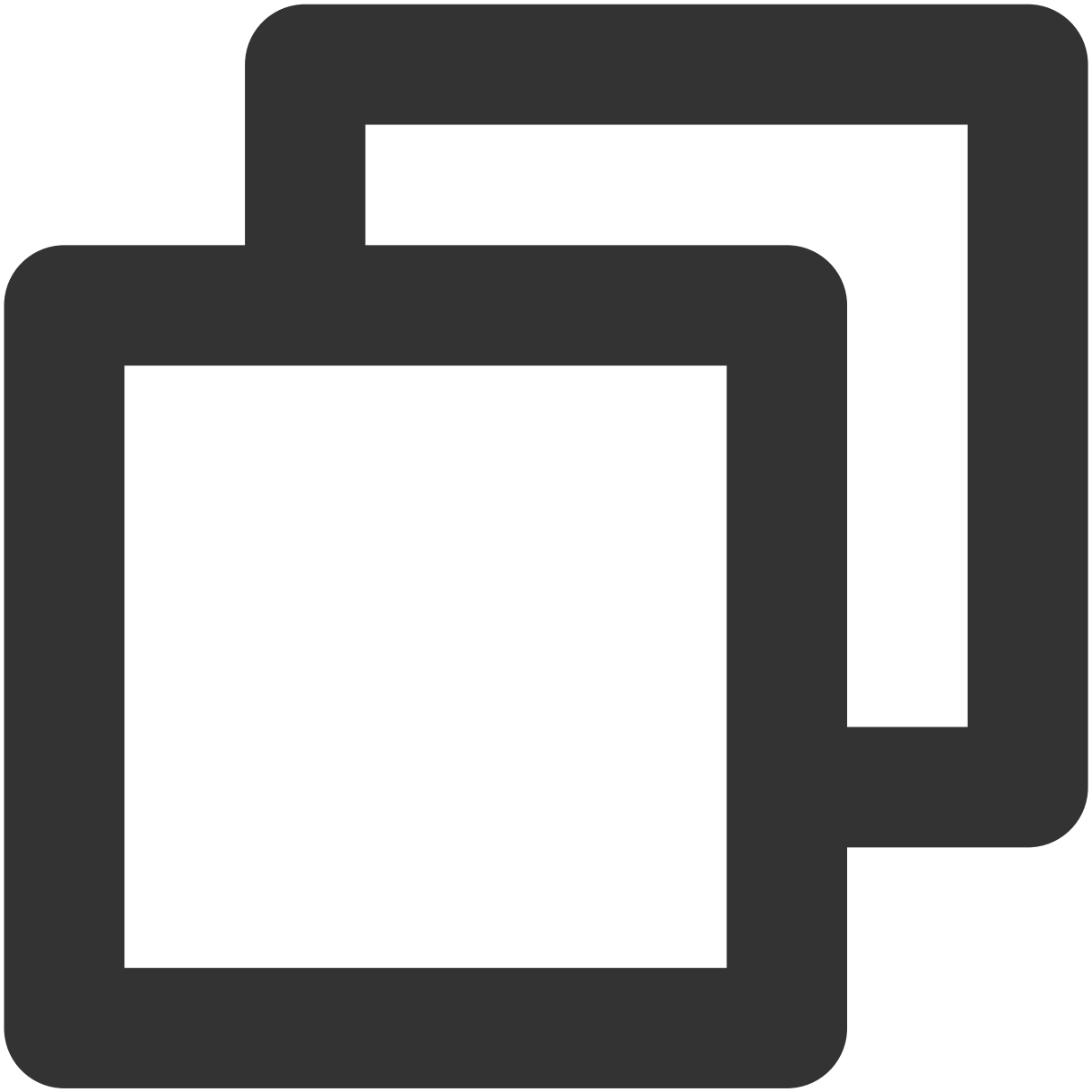
```
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => DigitalWatermark
                )

        )

)
```

提取数字水印任务

功能说明

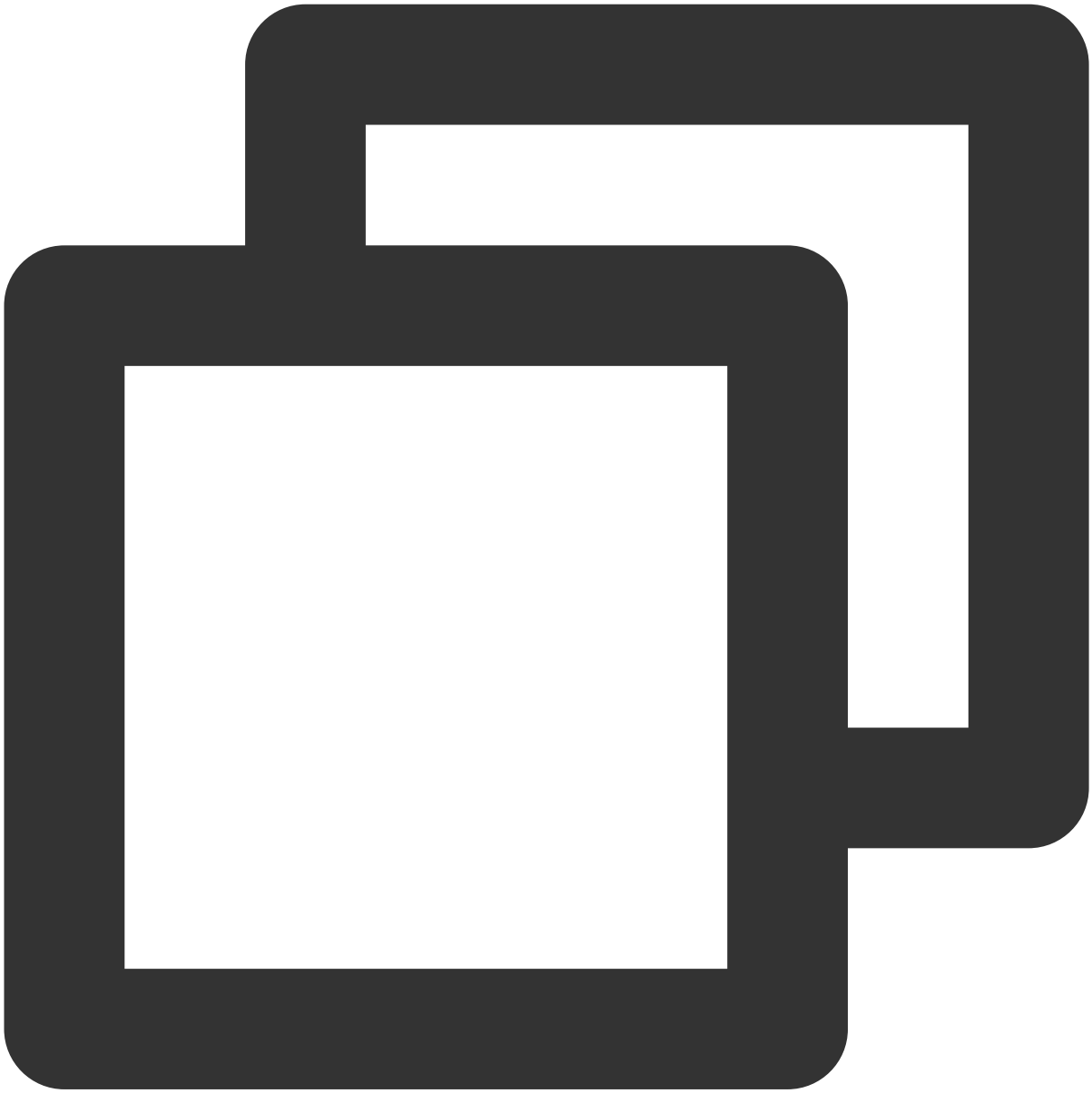
提交一个提取数字水印任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaExtractDigitalWatermarkJobs(arra
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 提取数字水印任务 https://www.tencentcloud.com/document/product/436/49048
    $result = $cosClient->createMediaExtractDigitalWatermarkJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'ExtractDigitalWatermark',
        'QueueId' => 'p81e648af2aee496885707caxxxxxxxxx',
        'Input' => array(
            'Object' => 'DigitalWatermark.mp4'
        ),
        'Operation' => array(
            'ExtractDigitalWatermark' => array(
                'Type' => 'Text',
                'Version' => 'V1',
            ),
//              'UserData' => 'xxx', // 透传用户信息
//              'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：ExtractDigitalWatermark	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否

CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

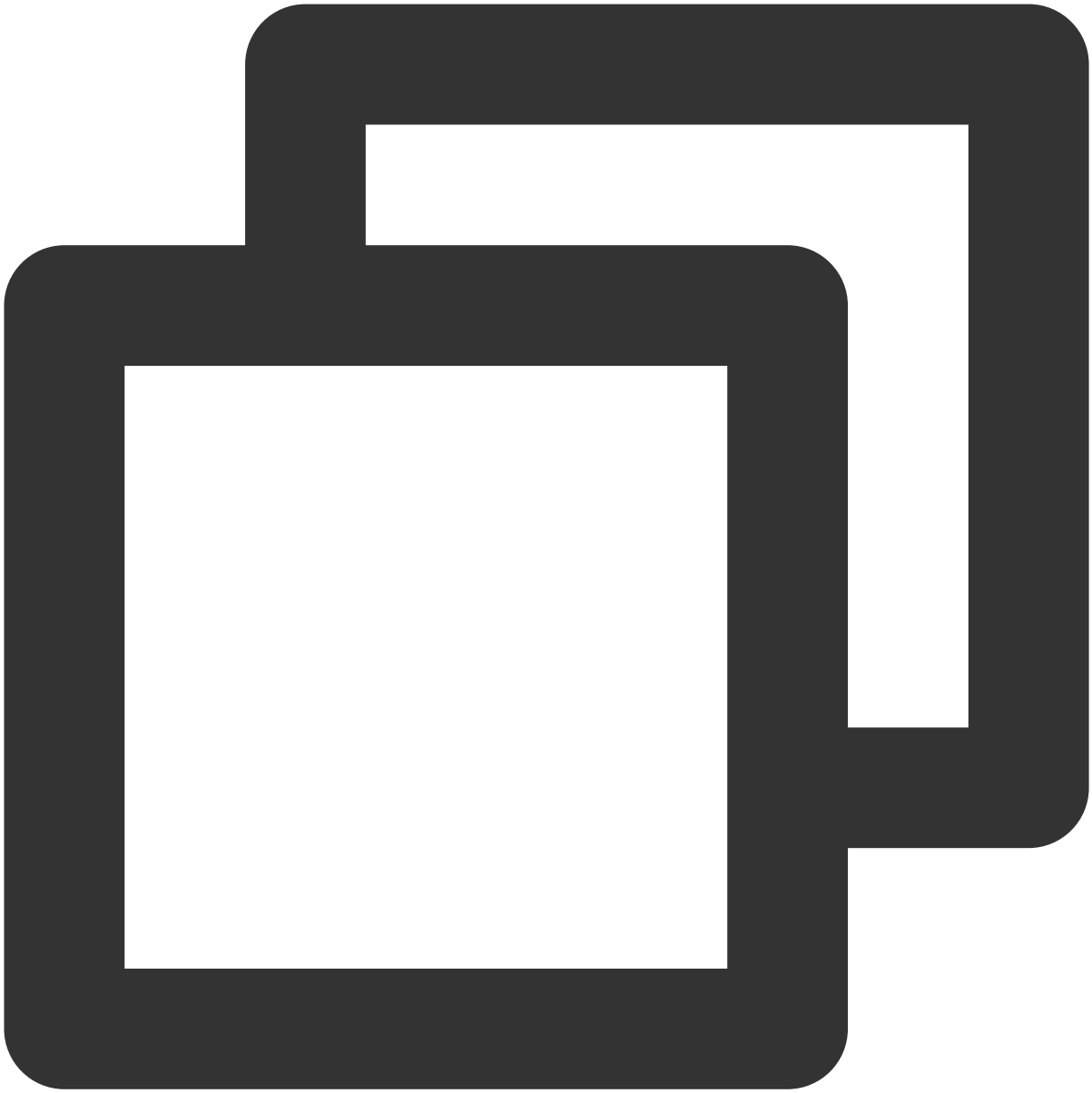
Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
ExtractDigitalWatermark	Request.Operation	提取数字水印配置	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否
UserData	Request.Operation	透传用户信息，可打印的 ASCII 码，长度不超过1024	String	否

Container 类型 ExtractDigitalWatermark 的具体数据类型描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Type	Request.Operation.ExtractDigitalWatermark	水印类型	String	是	Text
Version	Request.Operation.ExtractDigitalWatermark	水印版本	String	是	V1

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2OGI4YmZfZmNjYTNiAOISDAOIDUOMjU4ZjE=
    [ContentType] => application/xml
    [ContentLength] => 680
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-27T11:30:07+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => DigitalWatermark.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => j56038a4cc5da1a8s0d98a90sd806cf06
                    [Message] =>
                    [Operation] => Array
                        (
                            [ExtractDigitalWatermark] => Array
                                (
                                    [Type] => Text
                                    [Version] => V1
                                )
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2aee4z9x8c09z8xc09zbe086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => ExtractDigitalWatermark
                )

        )

)
```

超分辨率任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于超分辨率任务接口的 API 概览和 SDK 示例代码。

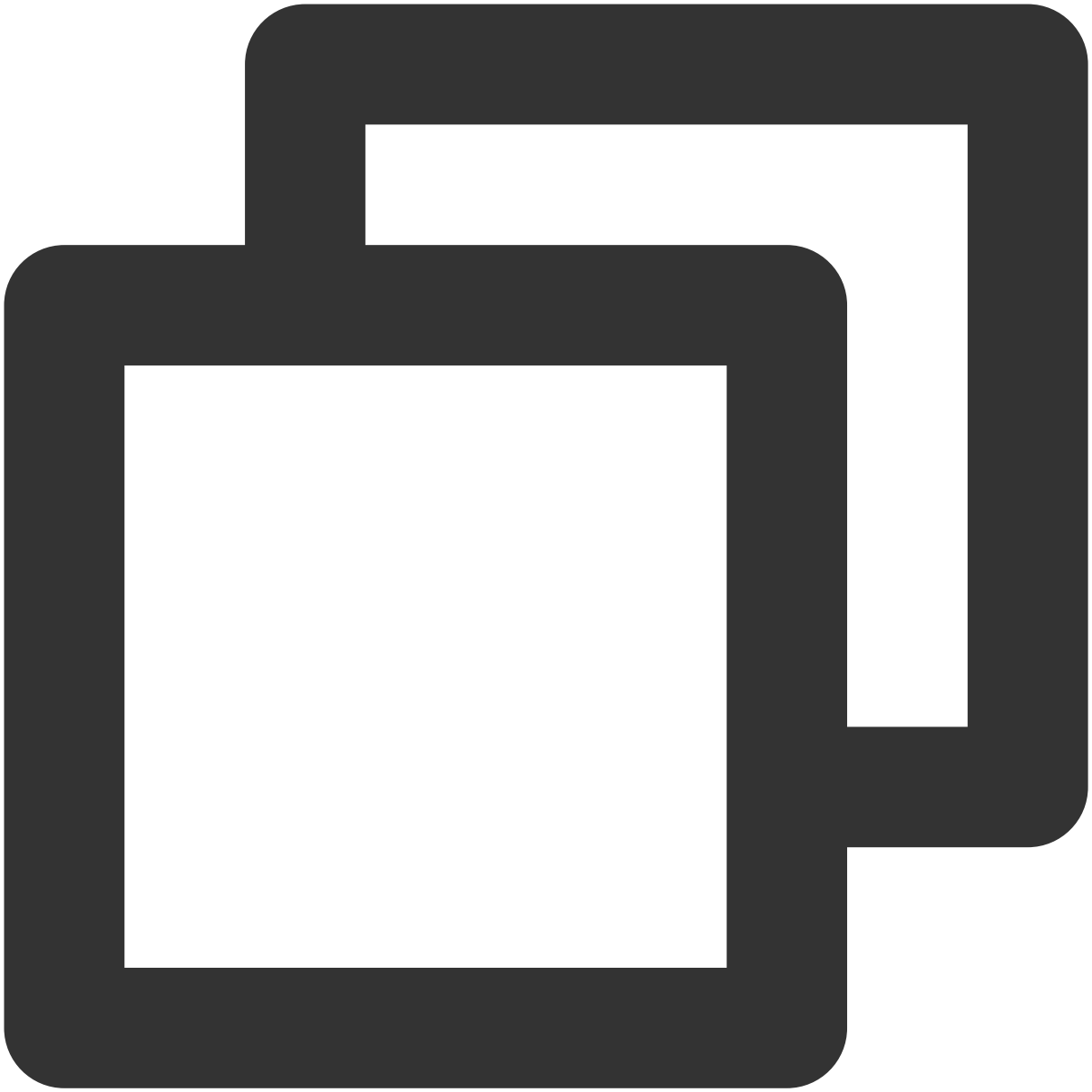
API	操作描述
提交超分辨率任务	提交超分辨率任务

提交超分辨率任务

功能说明

提交超分辨率任务。

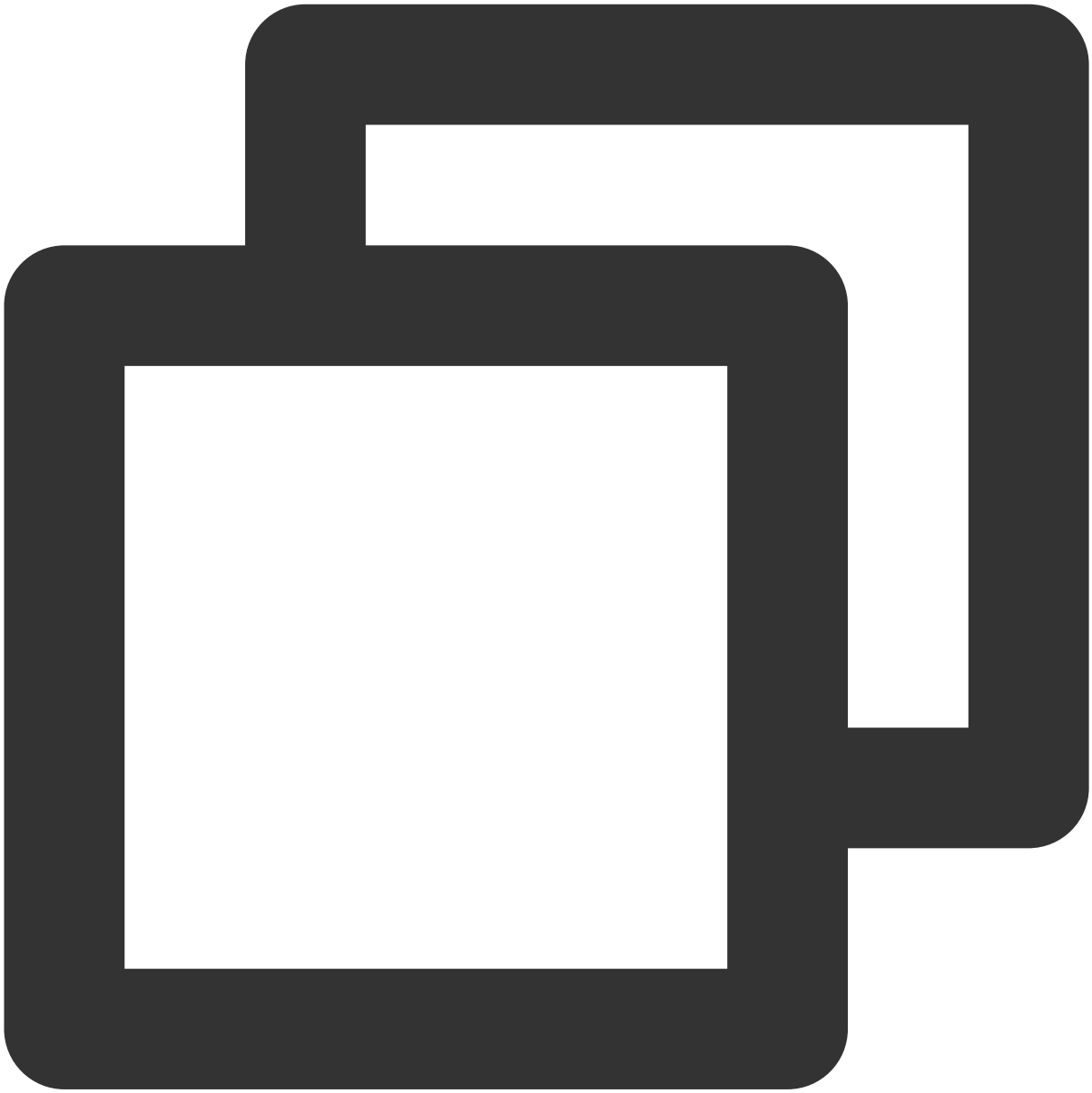
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaSuperResolutionJobs(array $args
```

请求示例

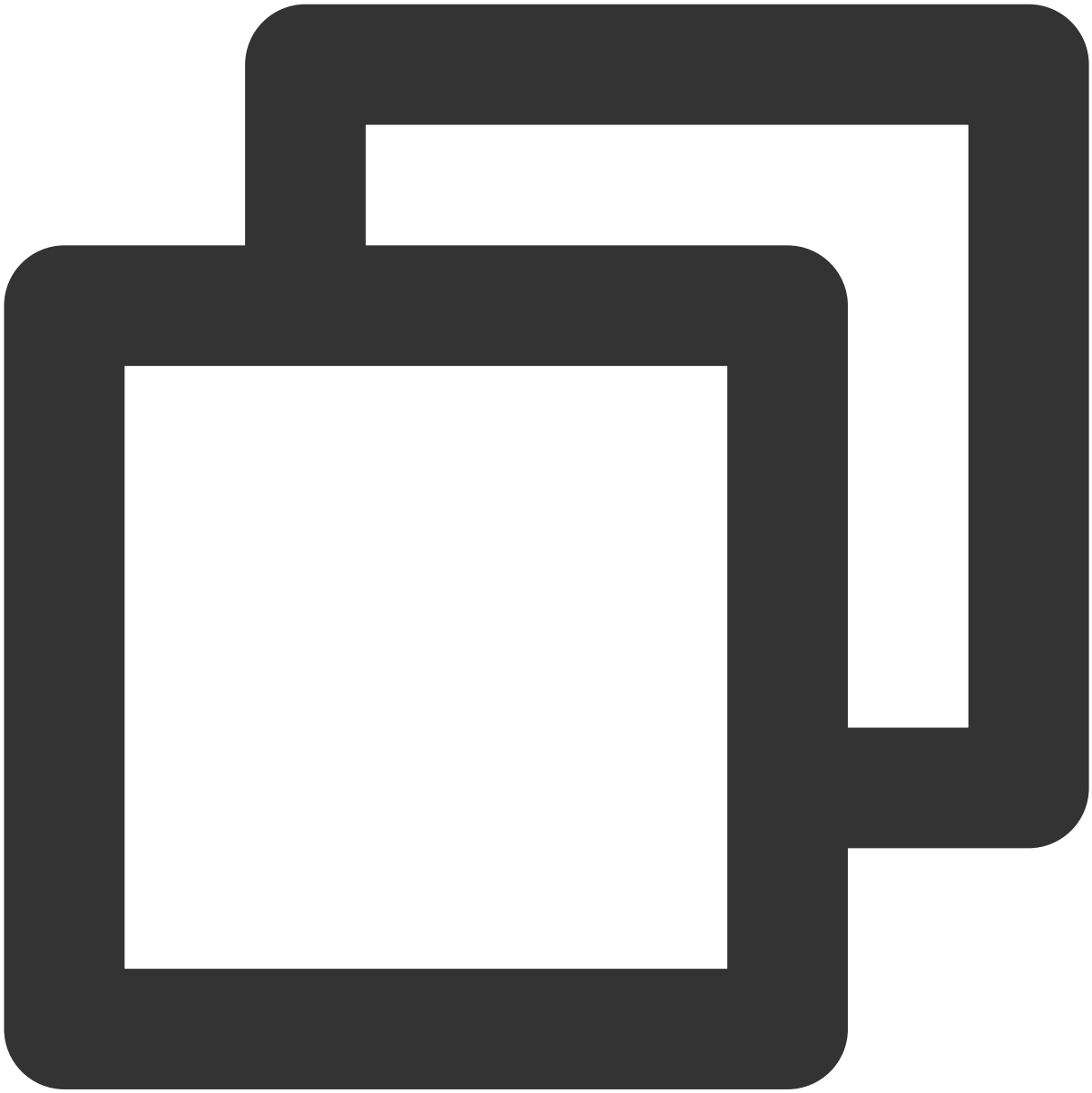
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaSuperResolutionJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'SuperResolution',
        'QueueId' => 'p81e648af2aee49688570xxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'TemplateId' =>'t19ea5e0c0b7054d7b904axxxxxxxxxxx',
            'TranscodeTemplateId' =>'t0b612860a293f41078xxxxxxxxxx',
            'WatermarkTemplateId' =>'t185e2e24551b24259a02xxxxxxxxxx',
            'DigitalWatermark' => array(
                'Message' => 'xxx',
                'Type' => 'Text',
                'Version' => 'V1',
                'IgnoreError' => 'true',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'SuperResolution.flv',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaSuperResolutionJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'SuperResolution',
        'QueueId' => 'p81e648af2aee49688570xxxxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'SuperResolution' => array(
                'Resolution' => '',
                'EnableScaleUp' => '',
            ),
            'Transcode' => array(
                'Tag' => '',
                'Name' => '',
                'Container' => array(
                    'Format' => '',
                ),
                'Video' => array(
                    'Codec' => '',
                    'Width' => '',
                    'Height' => '',
                    'Fps' => '',
                    'Remove' => '',
                    'Profile' => '',
                    'Bitrate' => '',
                    'Crf' => '',
                    'Gop' => '',
                    'Preset' => '',
                    'Bufsize' => '',
                    'Maxrate' => '',
                    'HlsTsTime' => '',
                    'Pixfmt' => '',
                    'LongShortMode' => '',
                ),
                'TimeInterval' => array(
                    'Start' => '',
                    'Duration' => '',
                ),
                'Audio' => array(
                    'Codec' => '',
                    'Samplerate' => '',
                    'Bitrate' => '',
```

```
                    'Channels' => '',
                    'Remove' => '',
                    'KeepTwoTracks' => '',
                    'SwitchTrack' => '',
                    'SampleFormat' => '',
                ),
                'TransConfig' => array(
                    'AdjDarMethod' => '',
                    'IsCheckReso' => '',
                    'ResoAdjMethod' => '',
                    'IsCheckVideoBitrate' => '',
                    'VideoBitrateAdjMethod' => '',
                    'IsCheckAudioBitrate' => '',
                    'AudioBitrateAdjMethod' => '',
                    'DeleteMetadata' => '',
                    'IsHdr2Sdr' => '',
                    'HlsEncrypt' => array(
                        'IsHlsEncrypt' => '',
                        'UriKey' => '',
                    ),
                ),
            ),
            'Watermark' => array(
                'Type' => '',
                'Pos' => '',
                'LocMode' => '',
                'Dx' => '',
                'Dy' => '',
                'StartTime' => '',
                'EndTime' => '',
                'Image' => array(
                    'Url' => '',
                    'Mode' => '',
                    'Width' => '',
                    'Height' => '',
                    'Transparency' => '',
                    'Background' => '',
                ),
                'Text' => array(
                    'FontSize' => '',
                    'FontType' => '',
                    'FontColor' => '',
                    'Transparency' => '',
                    'Text' => '',
                ),
            ),
            'DigitalWatermark' => array(
```

```
                'Message' => '',
                'Type' => '',
                'Version' => '',
                'IgnoreError' => '',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'SuperResolution.flv',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：SuperResolution	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否

版权所有：腾讯云计算（北京）有限责任公司

CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
SuperResolution	Request.Operation	指定超分辨率模板参数，不能与 TemplateId 同时为空	Container	否
TemplateId	Request.Operation	指定的超分辨率模 ID，优先使用模板 ID，不能与 SuperResolution 同时为空	String	否
Transcode	Request.Operation	指定转码模板参数，不能与 TranscodeTemplateId 同时为空	Container	否
TranscodeTemplateId	Request.Operation	指定的转码模板 ID，优先使用模板 ID，不能与 Transcode 同时为空	String	否
Watermark	Request.Operation	指定水印模板参数，同创建水印模板 CreateMediaTemplate 接口中的 Request.Watermark，最多传3个	Container 数组	否
WatermarkTemplateId	Request.Operation	指定的水印模板 ID，可以传多个水印模板 ID，最多传3个，优先使用模板 ID	String 数组	否
DigitalWatermark	Request.Operation	指定数字水印参数	Container	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

说明：

提交超分辨率任务必须传入转码参数。对于超分辨率参数，优先使用 TemplateId，无 TemplateId 时使用 SuperResolution；对于转码参数，优先使用 TranscodeTemplateId，无 TranscodeTemplateId 时使用 Transcode；对于水印参数，可以使用 WatermarkTemplateId 或 Watermark 设置，WatermarkTemplateId 优先级更高。

Container 类型 SuperResolution 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Resolution	Request.Operation.SuperResolution	同创建超分辨率模板 CreateMediaTemplate 接口中的 Request.Resolution	String	是
EnableScaleUp	Request.Operation.SuperResolution	同创建超分辨率模板 CreateMediaTemplate 接口中的 Request.EnableScaleUp	String	否
Version	Request.Operation.SuperResolution	同创建超分辨率模板 CreateMediaTemplate 接口中的 Request.Version	String	否

Container 类型 Transcode 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
TimeInterval	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TimeInterval	Container	否
Container	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Container	Container	否
Video	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Video	Container	否
Audio	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.Audio	Container	否
TransConfig	Request.Operation.Transcode	同创建转码模板 CreateMediaTemplate 接口中的 Request.TransConfig	Container	否
AudioMix	Request.Operation.Transcode	混音参数，详情见 AudioMix	Container 数组	否

Container 类型 DigitalWatermark 的具体数据描述如下：

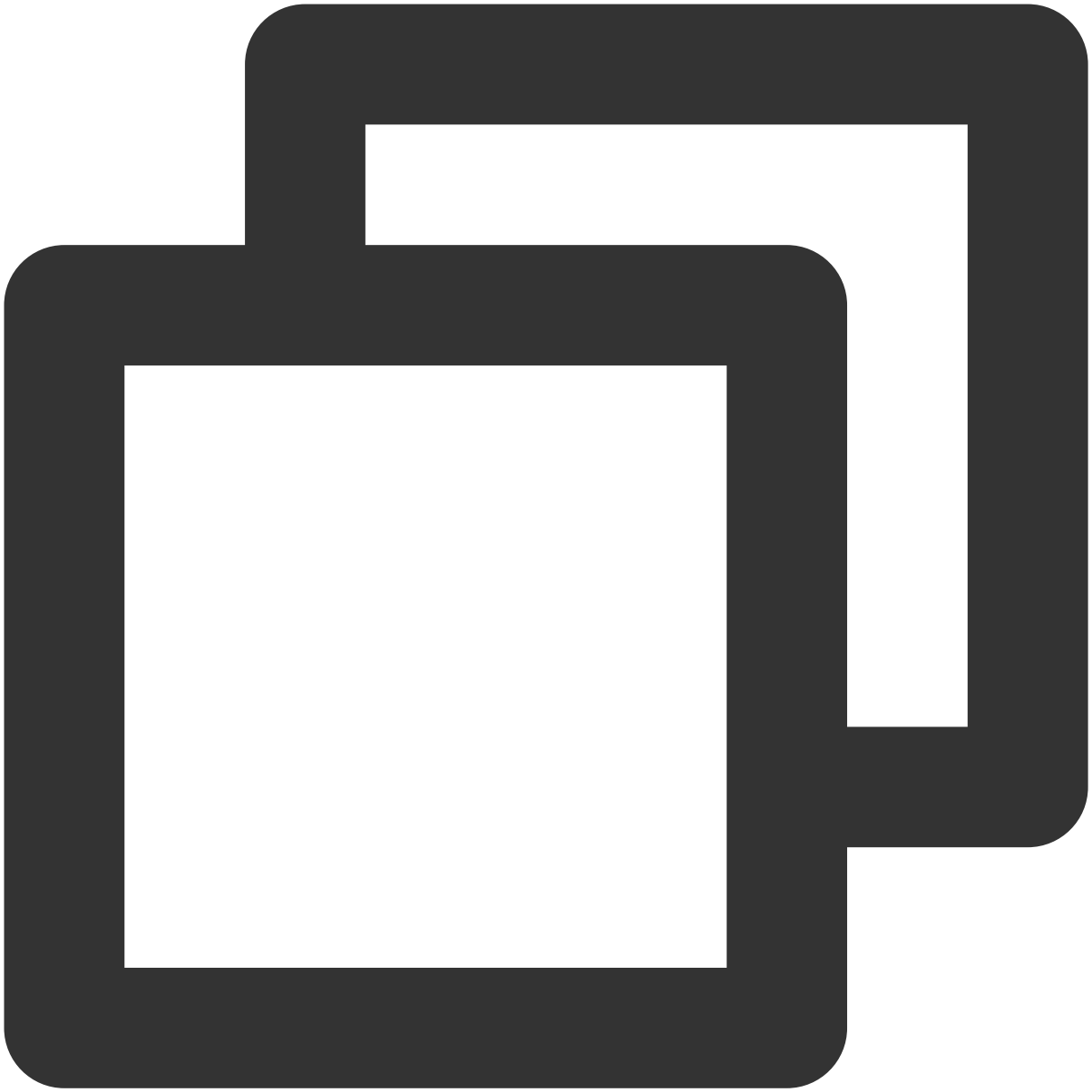
节点名称	父节点	描述	类型	必

（关键字）				选
Message	Request.Operation.DigitalWatermark	数字水印嵌入的字符串信息，长度不超过64个字符，仅支持中文、英文、数字、_、-和*	String	是
Type	Request.Operation.DigitalWatermark	水印类型，当前仅可设置为 Text	String	是
Version	Request.Operation.DigitalWatermark	水印版本，当前仅可设置为 V1	String	是
IgnoreError	Request.Operation.DigitalWatermark	当添加水印失败是否忽略错误继续执行任务，限制为 true/false，默认 false	string	否

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2OGY4ZDdfZmNjYTNiMGAJODIJAOIDJAOMmVmNDU=
    [ContentType] => application/xml
    [ContentLength] => 1170
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-27T16:03:35+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => j89be22b8c60011ec9070fasd8a09d80a9b8
                    [Message] =>
                    [Operation] => Array
                        (
                            [DigitalWatermark] => Array
                                (
                                    [IgnoreError] => true
                                    [Message] => xxx
                                    [State] => Running
                                    [Type] => Text
                                    [Version] => V1
                                )

                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => SuperResolution.flv
                                    [Region] => ap-guangzhou
                                )

                            [TemplateId] => t19ea5e0c0b7b904a8sd09a8sd09a80sd10
                            [TemplateName] => SuperResolution-1
                            [TranscodeTemplateId] => t19ea5e0c0b7b904a8sd09a8sd09a8
```

```
                                [WatermarkTemplateId] => t19ea5e0c0b7b904a8sd09a8asdasd
                                [UserData] => xxx
                                [JobLevel] => 0
                            )

                        [QueueId] => p81e648af2aee49688570asd8a90sd6
                        [StartTime] => -
                        [State] => Submitted
                        [Tag] => SuperResolution
                    )

            )

)
```

视频标签任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于视频标签任务接口的 API 概览和 SDK 示例代码。

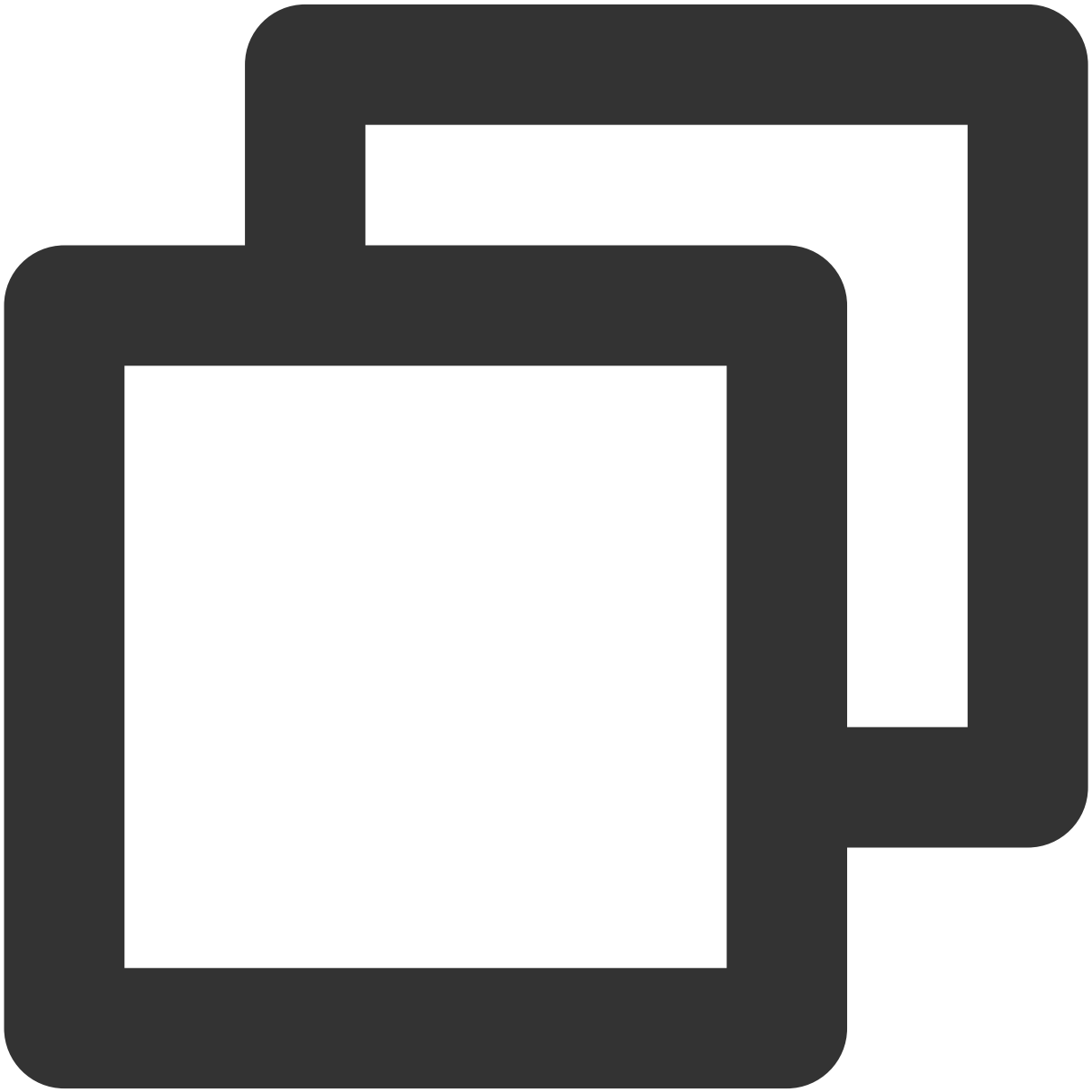
API	操作描述
提交视频标签任务	提交视频标签任务

提交视频标签任务

功能说明

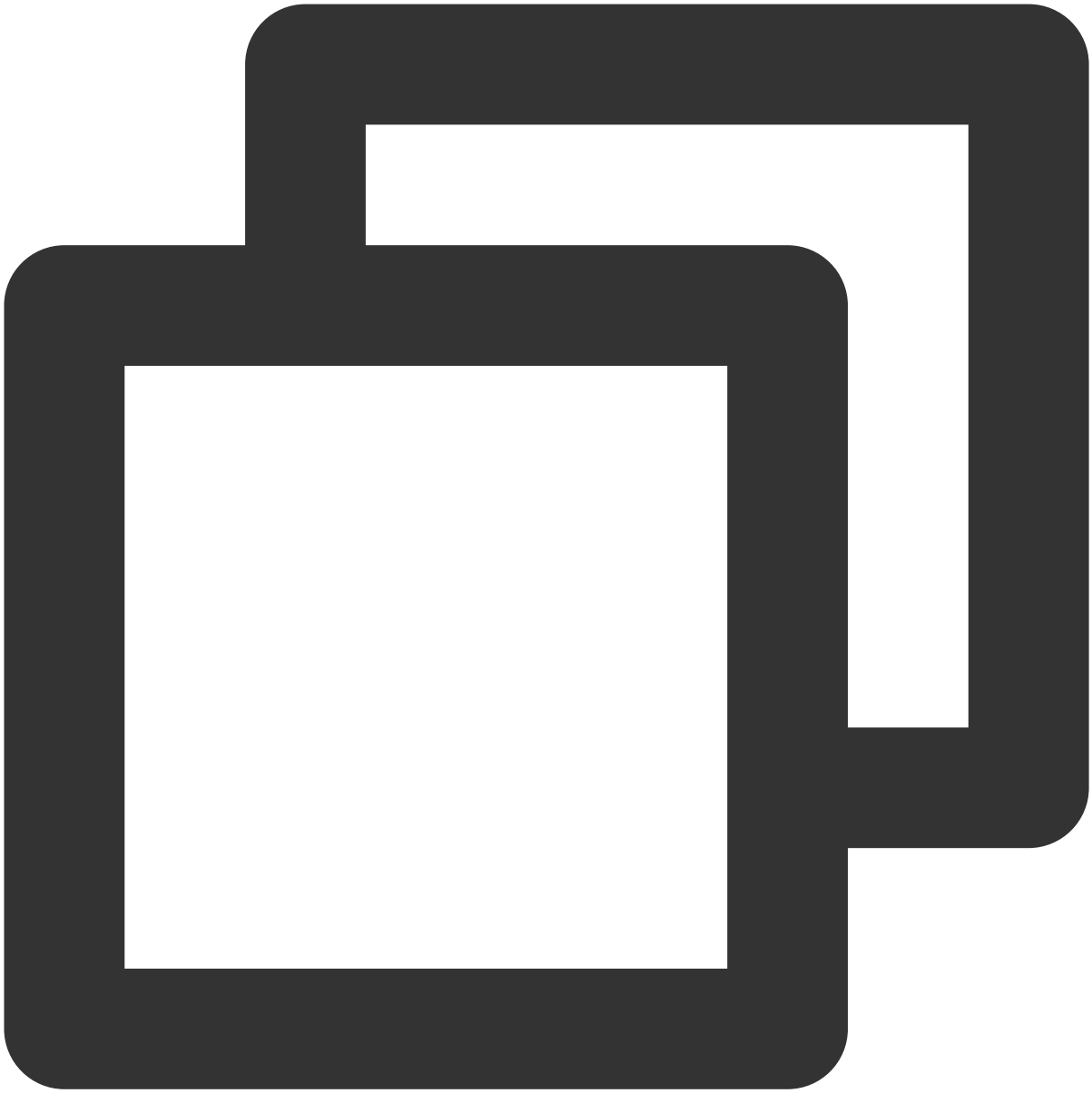
提交视频标签任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaVideoTagJobs(array $args = array
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 提交视频标签任务 https://cloud.tencent.com/document/product/436/67202
    $result = $cosClient->createMediaVideoTagJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'VideoTag',
        'QueueId' => 'p81e648af2aee496885707ca0xxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'VideoTag' => array(
                'Scenario' => 'Stream',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：VideoTag	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否

CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	执行视频标签任务的媒体文件名，目前支持 mp4、avi、mkv、wmv、rmvb、flv、mov 封装格式，视频时长超过30min的视频请 提交工单 处理	String	是

Container 类型 Operation 的具体数据描述如下：

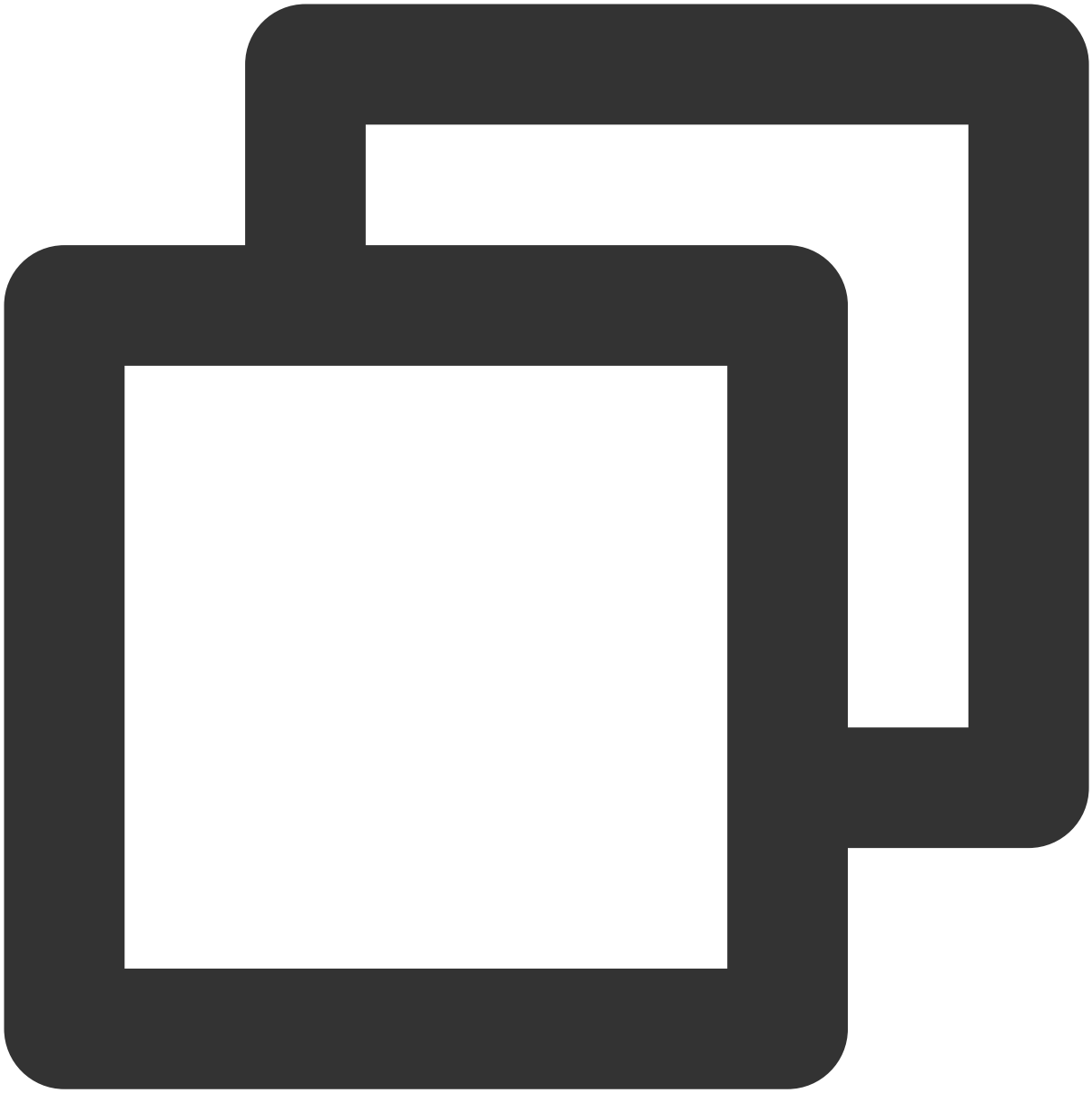
节点名称（关键字）	父节点	描述	类型	是否必选
VideoTag	Request.Operation	指定该 VideoTag 任务参数	Container	是
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否
UserData	Response.JobsDetail.Operation	透传用户信息	String	否

Container 类型 VideoTag 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Scenario	Request.Operation.VideoTag	场景类型，可选择视频标签的运用场景，不同的运用场景使用的算法、输入输出等都会有所差异	string	是	当前版本只适配 Stream 场景

版权所有：腾讯云计算（北京）有限责任公司

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
```

版权所有：腾讯云计算（北京）有限责任公司

```
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
                (
                )

        )

    [RequestId] => NjI2OTI2ZTdfZaOAUSIODUJAIODJAIODViNWQ=
    [ContentType] => application/xml
    [ContentLength] => 610
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-27T19:20:07+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => jfe591374c61b11ec9ad8a098sd09asd8aa
                    [Message] =>
                    [Operation] => Array
                        (
                            [VideoTag] => Array
                                (
                                    [Scenario] => Stream
                                )
                            [UserData] => xxx
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2aeea8d90a8s90d8a0de086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => VideoTag
                )

        )

)
```

版权所有：腾讯云计算（北京）有限责任公司

图片处理任务接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于图片处理任务接口的 API 概览和 SDK 示例代码。

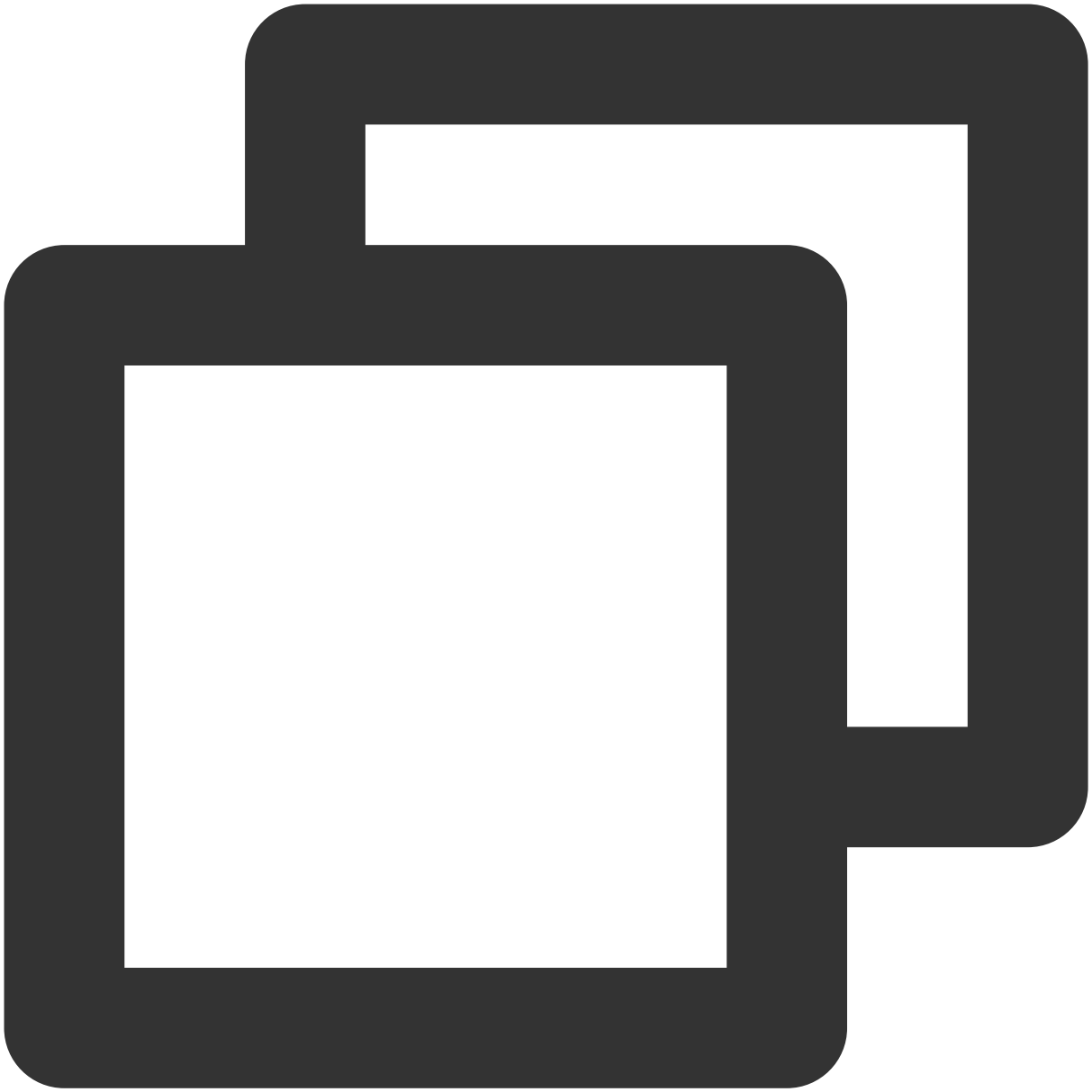
API	操作描述
提交图片处理任务	提交图片处理任务

提交图片处理任务

功能说明

提交一个图片处理任务。

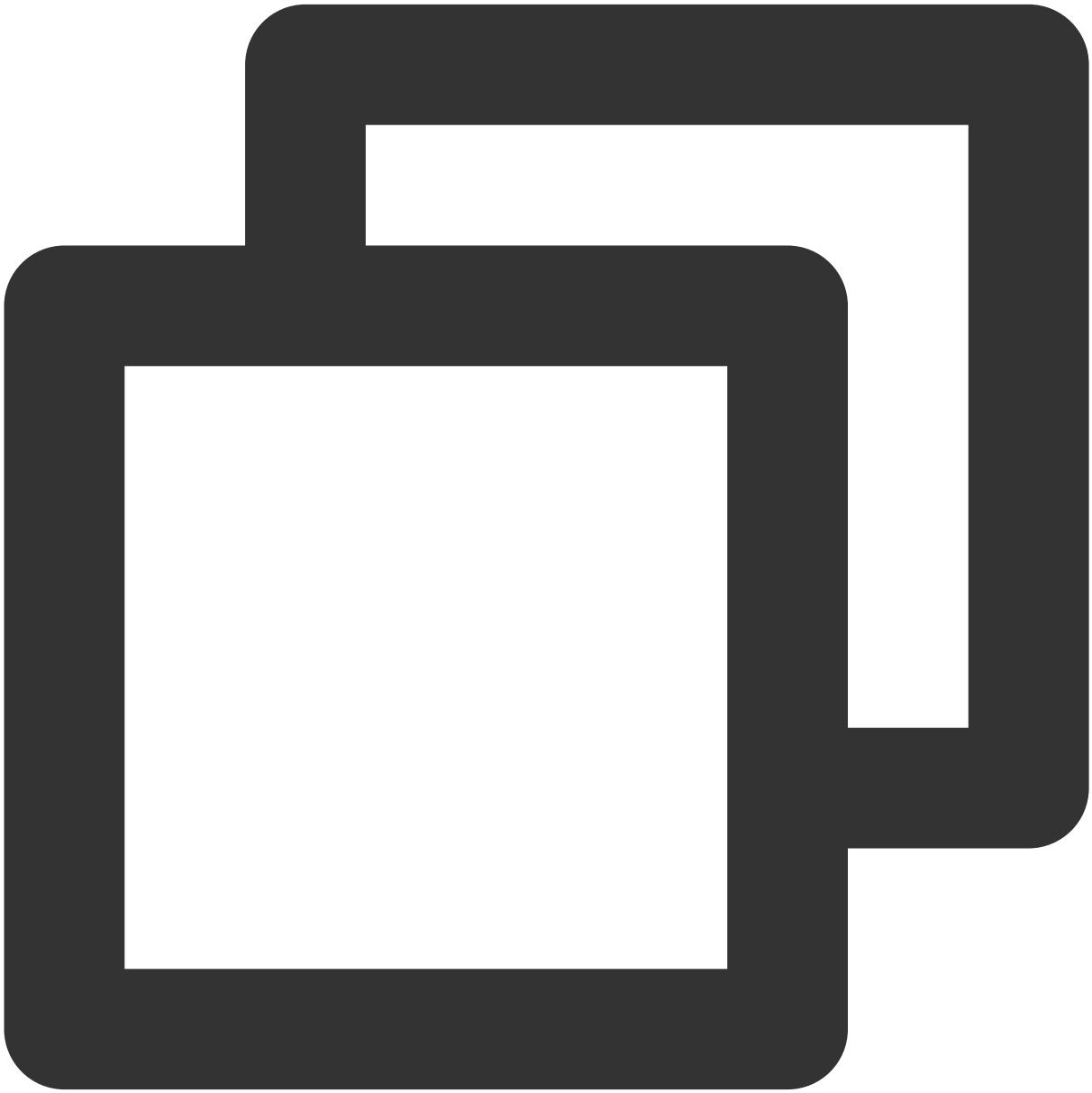
方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaPicProcessJobs(array $args = arr
```

请求示例

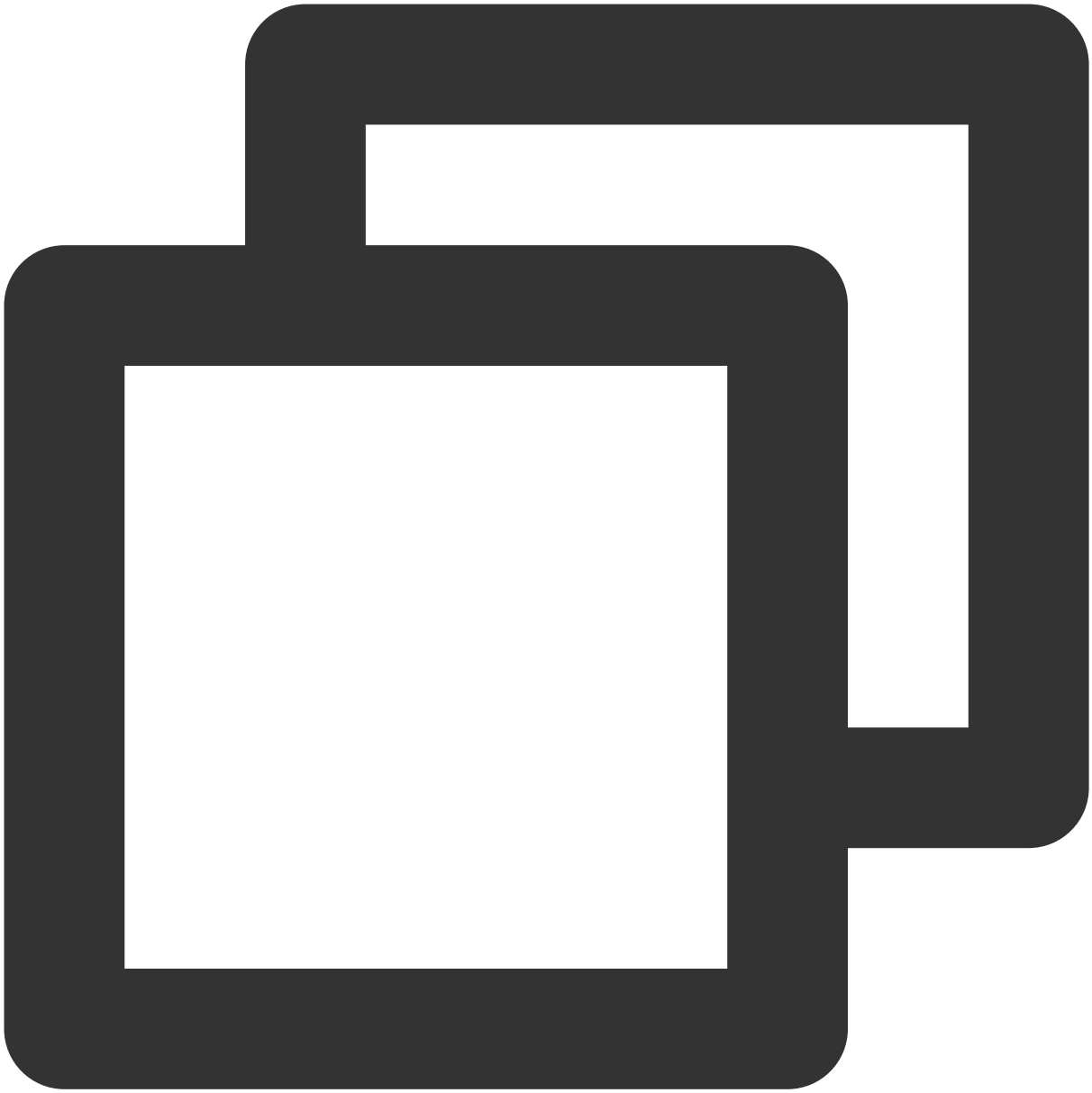
示例一： 使用模板



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaPicProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'PicProcess',
        'QueueId' => 'pcf4d6d9e5e734asd0as8d09as8d09a8d0',
        'Input' => array(
            'Object' => 'test01.png'
        ),
        'Operation' => array(
            'TemplateId' => 't1648745f76c354e8ad8a09sd890ad80a8d',
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'picprocess.jpg',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

示例二： 自定义参数



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    $result = $cosClient->createMediaPicProcessJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'PicProcess',
        'QueueId' => 'pcf4d6d9e5e734asd0as8d09as8d09a8d0',
        'Input' => array(
            'Object' => 'test01.png'
        ),
        'Operation' => array(
            'PicProcess' => array(
                'IsPicInfo' => '',
                'ProcessRule' => '',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'picprocess.jpg',
            ),
//              'UserData' => 'xxx', // 透传用户信息
//              'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高,
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：PicProcess	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	Container	是

QueueId	Request	任务所在的队列 ID	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	String	否
CallBack	Request	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
PicProcess	Request.Operation	指定该任务的参数，同创建图片处理模板 CreateMediaTemplate 接口中的 Request.PicProcess	Container	否
TemplateId	Request.Operation	指定的模板 ID	String	否
Output	Request.Operation	结果输出地址	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
JobLevel	Request.Operation	任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0	String	否

注意：

优先使用 TemplateId，无 TemplateId 时使用对应任务类型的参数。

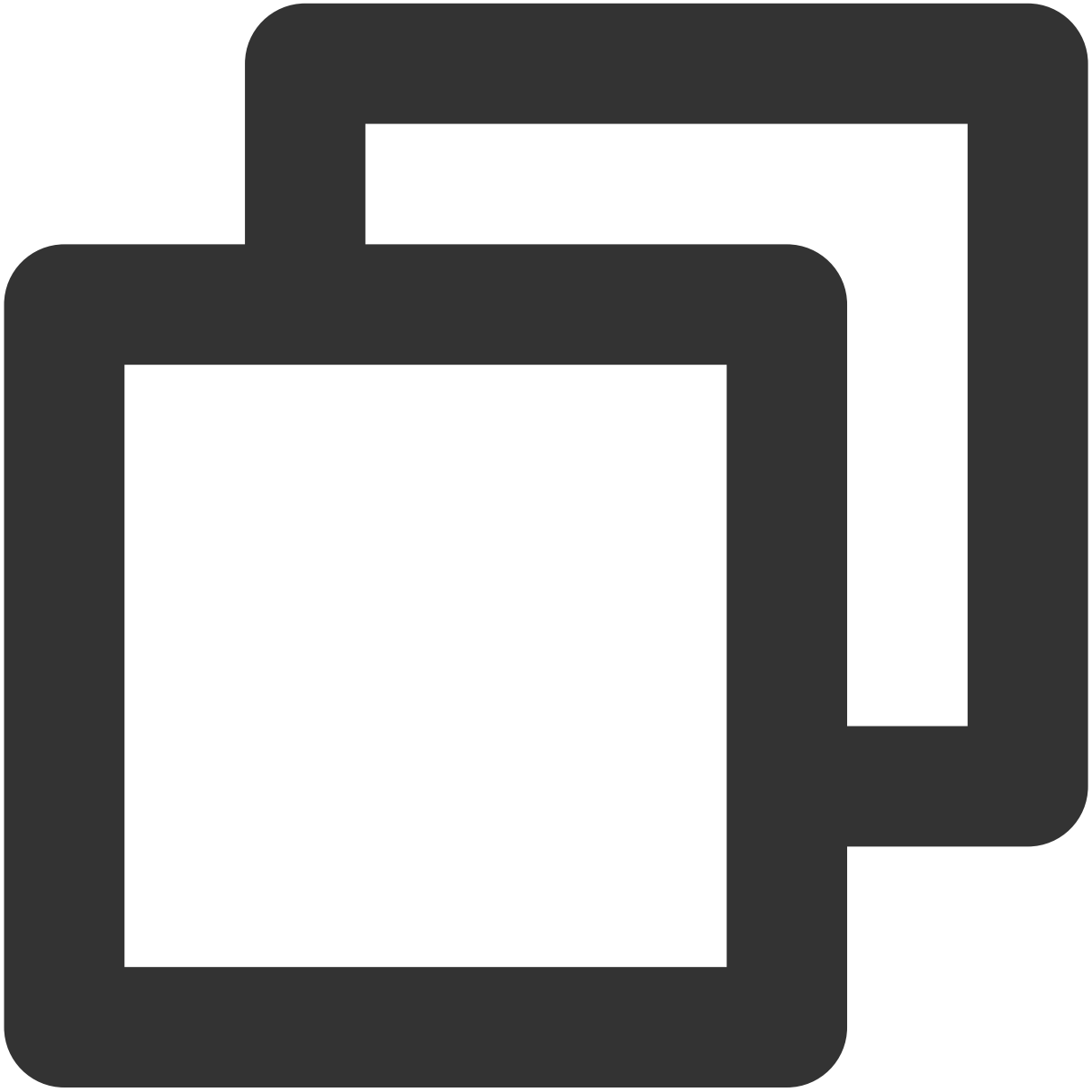
Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选

版权所有：腾讯云计算（北京）有限责任公司

Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	结果文件的名字	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
```

```
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
                (
                )

        )

    [RequestId] => NjI2N2E4NTRfNzgwYzAJODIAUSIODUAU3ZQ==
    [ContentType] => application/xml
    [ContentLength] => 795
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/pic_jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-26T16:07:48+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => test01.png
                            [Region] => ap-guangzhou
                        )

                    [JobId] => cf62e1d50c5sa98d09a8d0a6cf06
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => picprocess.jpg
                                    [Region] => ap-guangzhou
                                )

                            [TemplateId] => t1648745f76asd70as8d90as8d0aa8a09
                            [TemplateName] => PicProcess-1
                            [UserData] => xxx
                            [UserData] => 0
```

```
                        )

                    [QueueId] => pcf4d6d9e5e734d0b82eas90d8a90sdd0
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => PicProcess
                )

        )

)
```

转封装任务接口

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于转封装任务接口的 API 概览和 SDK 示例代码。

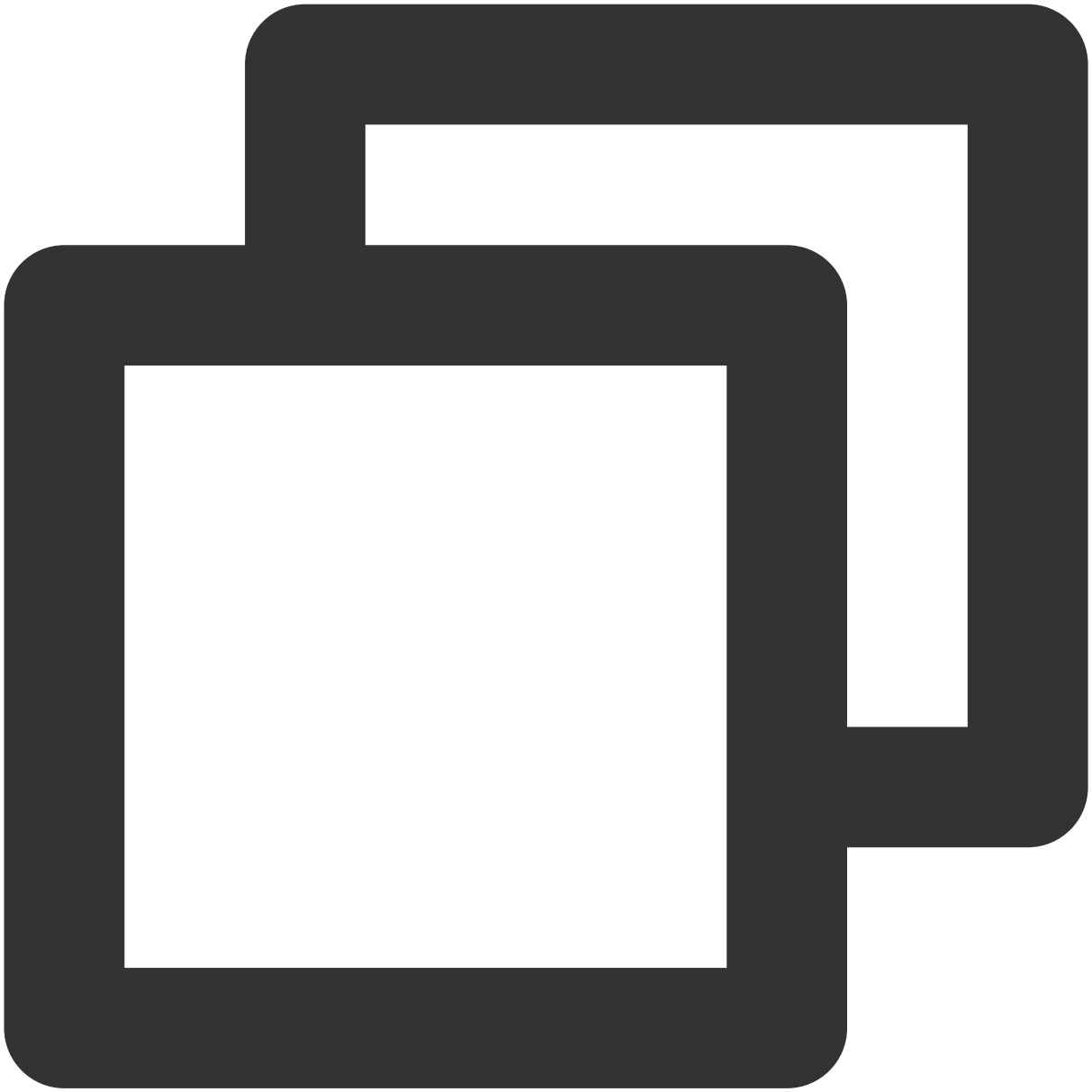
API	操作描述
提交转封装任务	提交转封装任务

提交转封装任务

功能说明

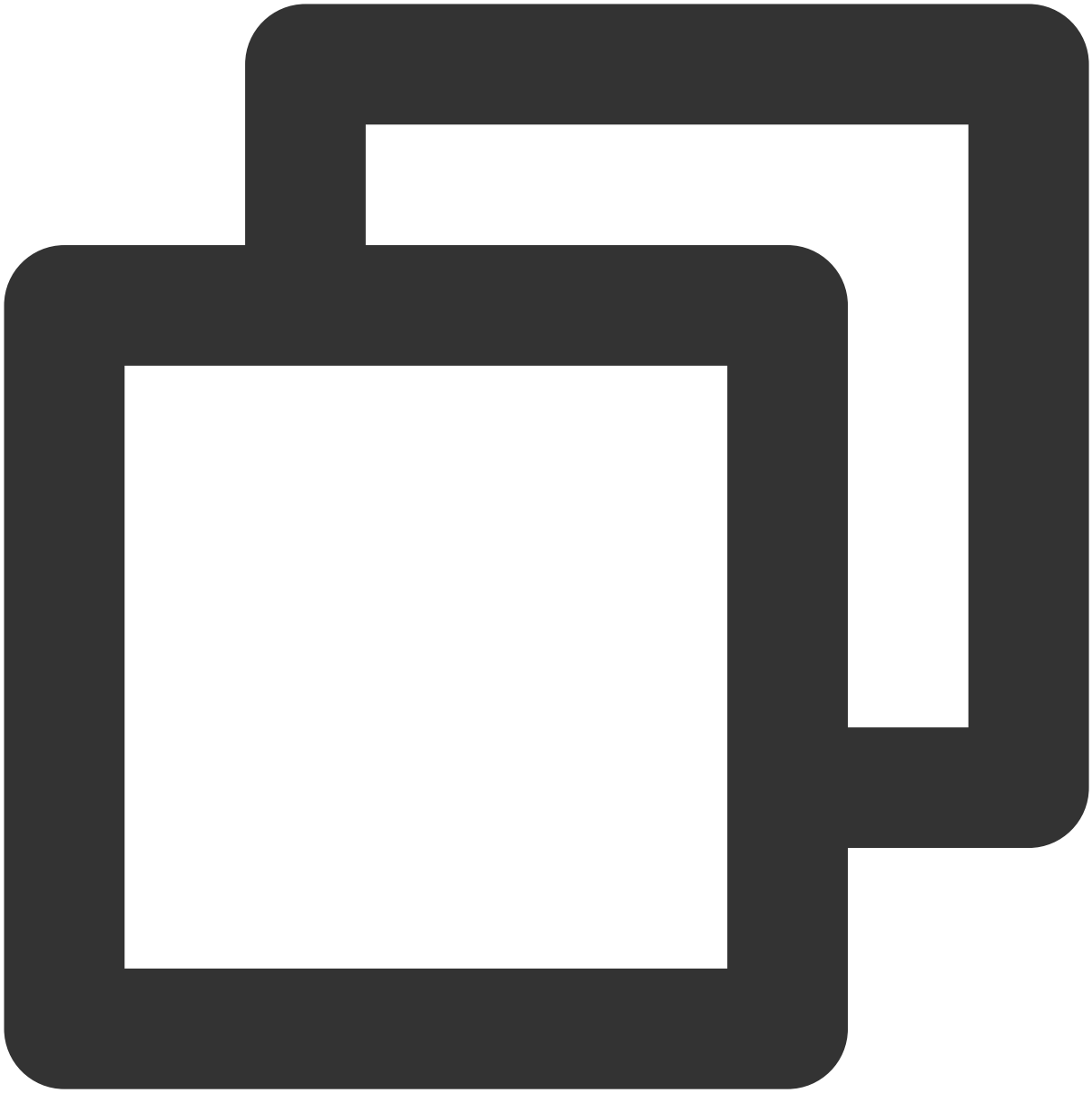
提交转封装任务。

方法原型



```
public Guzzle\\Service\\Resource\\Model createMediaSegmentJobs(array $args = array(
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 提交转封装任务 https://www.tencentcloud.com/document/product/436/49053
    $result = $cosClient->createMediaSegmentJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'Segment',
        'QueueId' => 'p81e648af2aee496885707ca0xxxxxxxxx',
        'Input' => array(
            'Object' => 'video01.mp4'
        ),
        'Operation' => array(
            'Segment' => array(
                'Format' => 'mkv',
                'Duration' => '5',
                'HlsEncrypt' => array(
                    'IsHlsEncrypt' => 'false',
                    'UriKey' => '',
                ),
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000',
                'Object' => 'Segment-trans${Number}.mkv',
            ),
//            'UserData' => 'xxx', // 透传用户信息
//            'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，
        ),
        'CallBack' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	创建任务的 Tag：Segment	String	是
Input	Request	待操作的媒体信息	Container	是

Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID	String	是
CallBack	Request	回调地址	String	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	媒体文件名	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Segment	Request.Operation	指定转封装参数	Container	是
Output	Request.Operation	结果输出地址	Container	是

Container 类型 Segment 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Format	Request.Operation.Segment	封装格式	String	是	aac、mp3、flac、mp4、ts、mkv、avi、hls、m3u8
Duration	Request.Operation.Segment	转封装时长，单位：秒	String	否	不小于5的整数
HlsEncrypt	Request.Operation.Segment	hls 加密配置	Container	否	无, 只有当封装格式为 hls 时生效

Container 类型 HlsEncrypt 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
IsHlsEncrypt	Request.Operation.Segment.HlsEncrypt	是否开启	String	否	false	1. true/false 2. Segment.Format

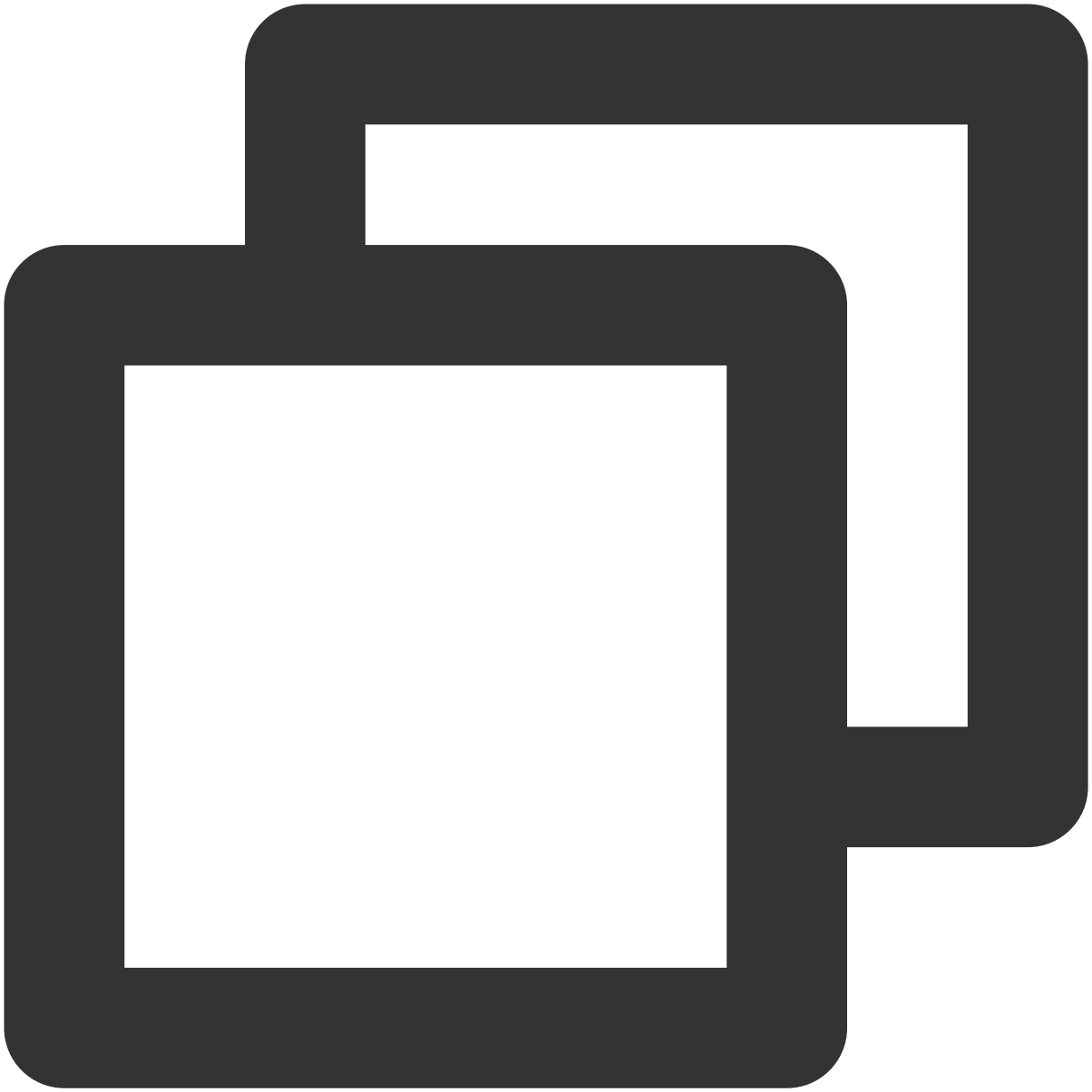
版权所有：腾讯云计算（北京）有限责任公司

		HLS 加密				为 HLS 时支持加密
UriKey	Request.Operation.Segment.HlsEncrypt	HLS 加密的 key	String	否	无	当 IsHlsEncrypt 为 true 时，该参数才有意义

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域	String	是
Bucket	Request.Operation.Output	存储结果的存储桶	String	是
Object	Request.Operation.Output	输出结果的文件名，如果设置了Duration, 且 Format 不为 HLS 或 m3u8 时, 文件名必须包含${Number}参数作为自定义转封装后每一小段音/视频流的输出序号	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [Body] => GuzzleHttp\\Psr7\\Stream Object
        (
            [stream:GuzzleHttp\\Psr7\\Stream:private] => Resource id #88
            [size:GuzzleHttp\\Psr7\\Stream:private] =>
            [seekable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [readable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [writable:GuzzleHttp\\Psr7\\Stream:private] => 1
            [uri:GuzzleHttp\\Psr7\\Stream:private] => php://temp
            [customMetadata:GuzzleHttp\\Psr7\\Stream:private] => Array
```

版权所有：腾讯云计算（北京）有限责任公司

```
                (
                )

        )

    [RequestId] => NjI2YTA2MzZfZAODIJAIODJAIODZF8xNjc2YTA=
    [ContentType] => application/xml
    [ContentLength] => 866
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
    [Response] => Array
        (
            [JobsDetail] => Array
                (
                    [Code] => Success
                    [CreationTime] => 2022-04-28T11:12:54+0800
                    [EndTime] => -
                    [Input] => Array
                        (
                            [BucketId] => examplebucket-1250000000
                            [Object] => video01.mp4
                            [Region] => ap-guangzhou
                        )

                    [JobId] => j184996a2c6a111ec954767a8248f19aa
                    [Message] =>
                    [Operation] => Array
                        (
                            [Output] => Array
                                (
                                    [Bucket] => examplebucket-1250000000
                                    [Object] => Segment-trans${Number}.mkv
                                    [Region] => ap-guangzhou
                                )

                            [Segment] => Array
                                (
                                    [Duration] => 10
                                    [Format] => mkv
                                    [HlsEncrypt] => Array
                                        (
                                            [IsHlsEncrypt] => false
                                            [UriKey] =>
                                        )

                                )
                            [UserData] => xxx
```

```
                            [JobLevel] => 0
                        )

                    [QueueId] => p81e648af2az8x7ca0898s0dasbe086
                    [StartTime] => -
                    [State] => Submitted
                    [Tag] => Segment
                )

        )

)
```

模板接口
搜索模板接口

最近更新时间：2024-03-13 10:25:35

简介

本文档提供关于搜索模板接口的 API 概览和 SDK 示例代码。

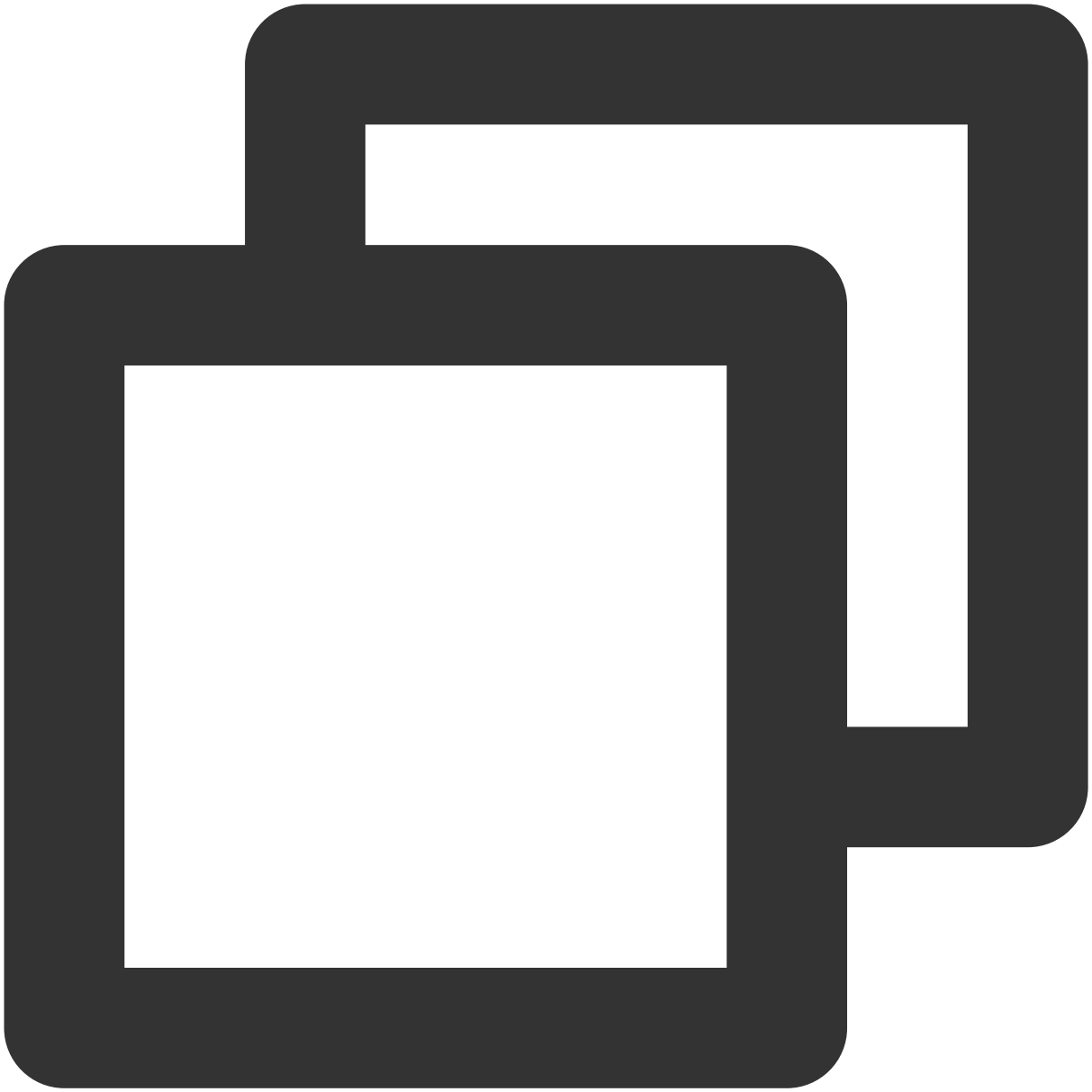
API	操作描述
搜索动图模板	用于搜索动图模板。
搜索截图模板	用于搜索截图模板。
搜索水印模板	用于搜索水印模板。
搜索转码模板	用于搜索转码模板。
搜索拼接模板	用于搜索拼接模板。
搜索极速高清转码模板	用于搜索极速高清转码模板。
搜索精彩集锦模板	用于搜索精彩集锦模板。
搜索人声分离模板	用于搜索人声分离转码模板。
搜索视频增强模板	用于搜索视频增强模板。
搜索图片处理模板	用于搜索图片处理模板。
搜索超分辨率模板	用于搜索超分辨率模板。

搜索模板

功能说明

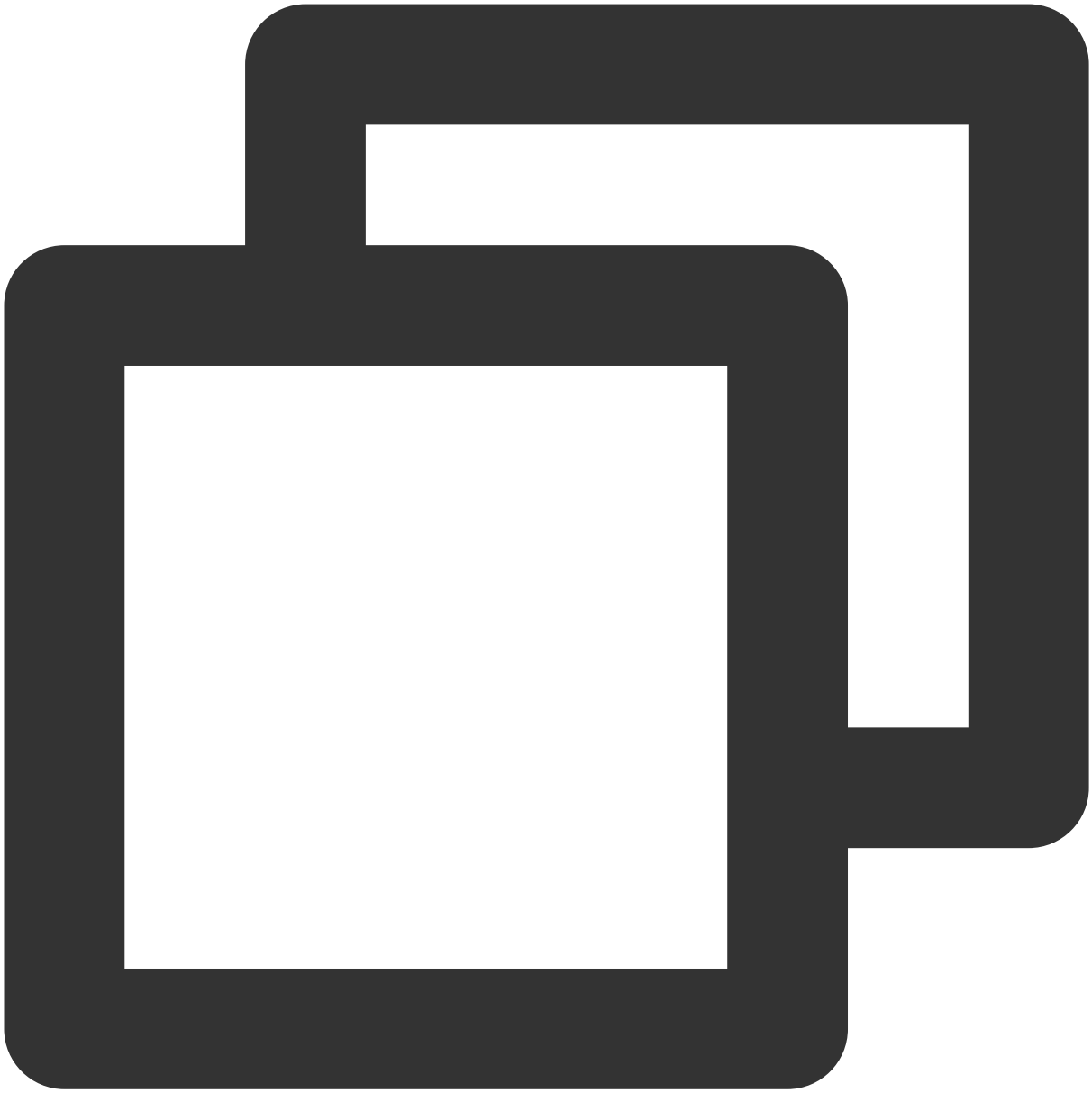
用于搜索模板，支持 Animation、Snapshot、Watermark、Transcode、Concat、HighSpeedHd、VideoMontage、VoiceSeparate、VideoProcess、PicProcess、SuperResolution 多种模板类型。

方法原型



```
public Guzzle\\Service\\Resource\\Model describeMediaTemplates(array $args = array(
```

请求示例



```
<?php

require dirname(__FILE__) . '/../vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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:
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

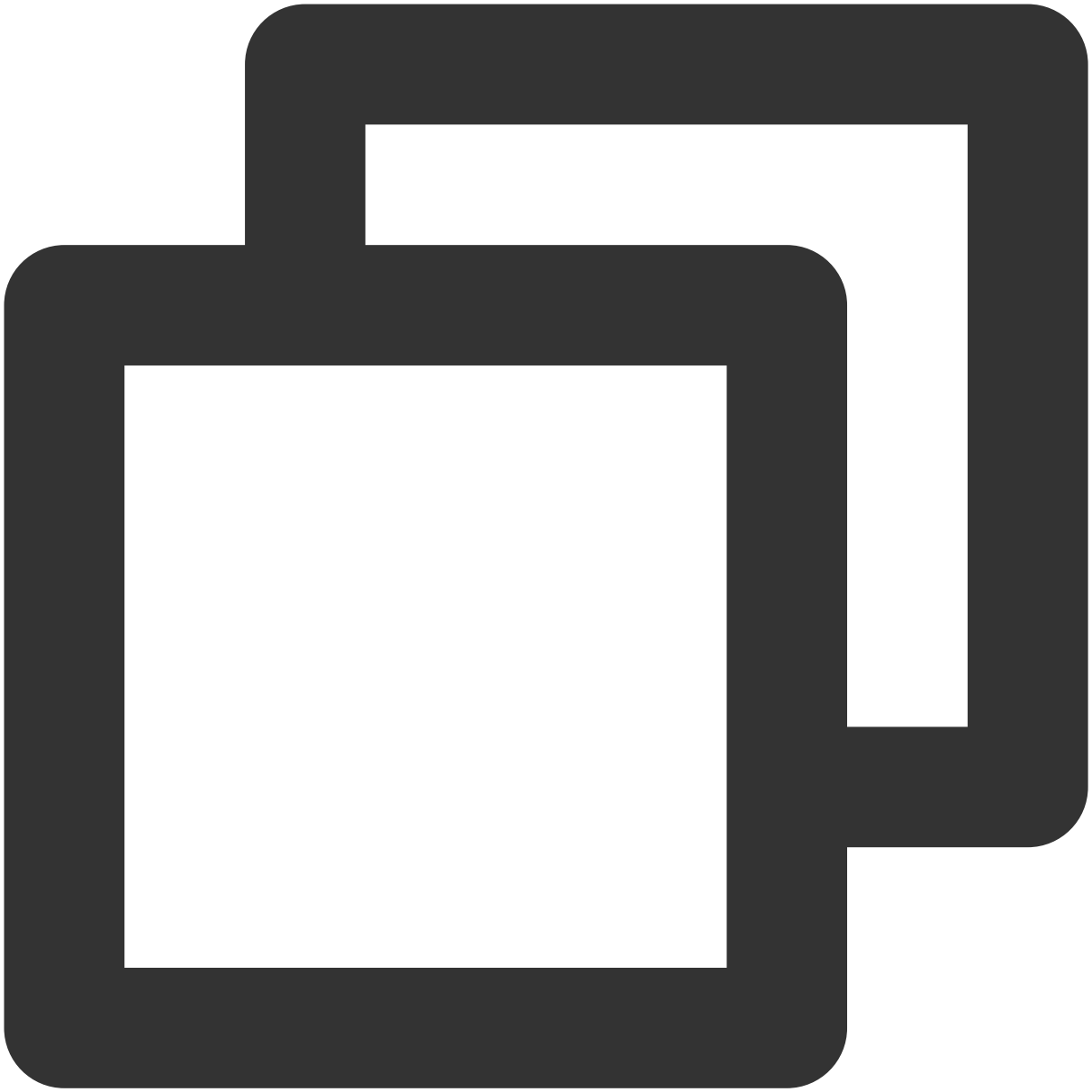
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // 查询模板列表
    $result = $cosClient->describeMediaTemplates(array(
        'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，
        'Tag' => '', // 模板 Tag：Animation、Snapshot、Watermark、Transcode、Concat、
        'Category' => 'Custom',
        'Ids' => '',
        'Name' => '',
        'PageNumber' => '',
        'PageSize' => '',
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	必选
Tag	无	模板 Tag	String	是
Category	无	Official，Custom，默认值：Custom	String	否
Ids	无	模板 ID，以 `,` 符号分割字符串	String	否
Name	无	模板名称前缀	String	否
PageNumber	无	第几页	Integer	否
PageSize	无	每页个数	Integer	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjJhOTRlZDFfNzgwYzdkAOIUDOAIUDTA4ZWE=
    [ContentType] => application/xml
    [ContentLength] => 782
    [TotalCount] => 1
    [PageNumber] => 1
    [PageSize] => 10
    [TemplateList] => Array
        (
            [0] => Array
```

```
                (
                    [TemplateId] => t1456ea89fcbdf45basd7fa98sd97a5
                    [Name] => VoiceSeparate-Name
                    [State] => Normal
                    [Tag] => VoiceSeparate
                    [CreateTime] => 2021-12-27T22:19:15+0800
                    [UpdateTime] => 2021-12-27T22:19:15+0800
                    [BucketId] => examplebucket-125000000
                    [Category] => Custom
                    [VoiceSeparate] => Array
                        (
                            [AudioMode] => AudioAndBackground
                            [AudioConfig] => Array
                                (
                                    [Codec] => mp3
                                    [Samplerate] => 44100
                                    [Bitrate] => 128
                                    [Channels] => 2
                                )

                        )

                )

        )

    [Bucket] => examplebucket-125000000
    [Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
)
```

文件处理
多文件打包压缩

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于多文件打包压缩接口的 API 概览和 SDK 示例代码。

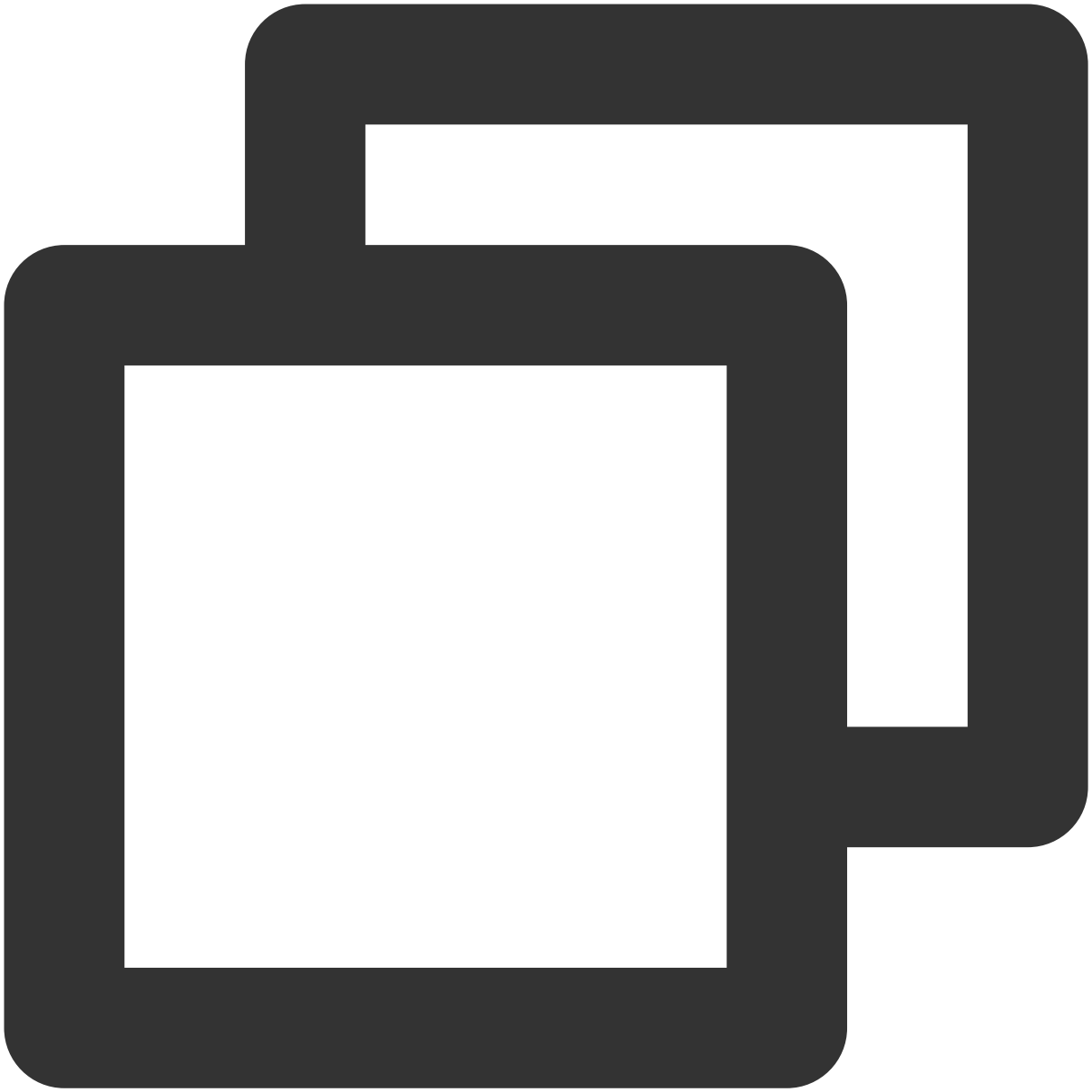
API	操作描述
提交多文件打包压缩任务	多文件打包压缩功能可以将您的多个文件，打包为 zip 等压缩包格式，以提交任务的方式进行多文件打包压缩，异步返回打包后的文件，该接口属于 POST 请求。
查询多文件打包压缩结果	本接口用于主动查询指定的多文件打包压缩任务结果。

提交多文件打包压缩任务

功能说明

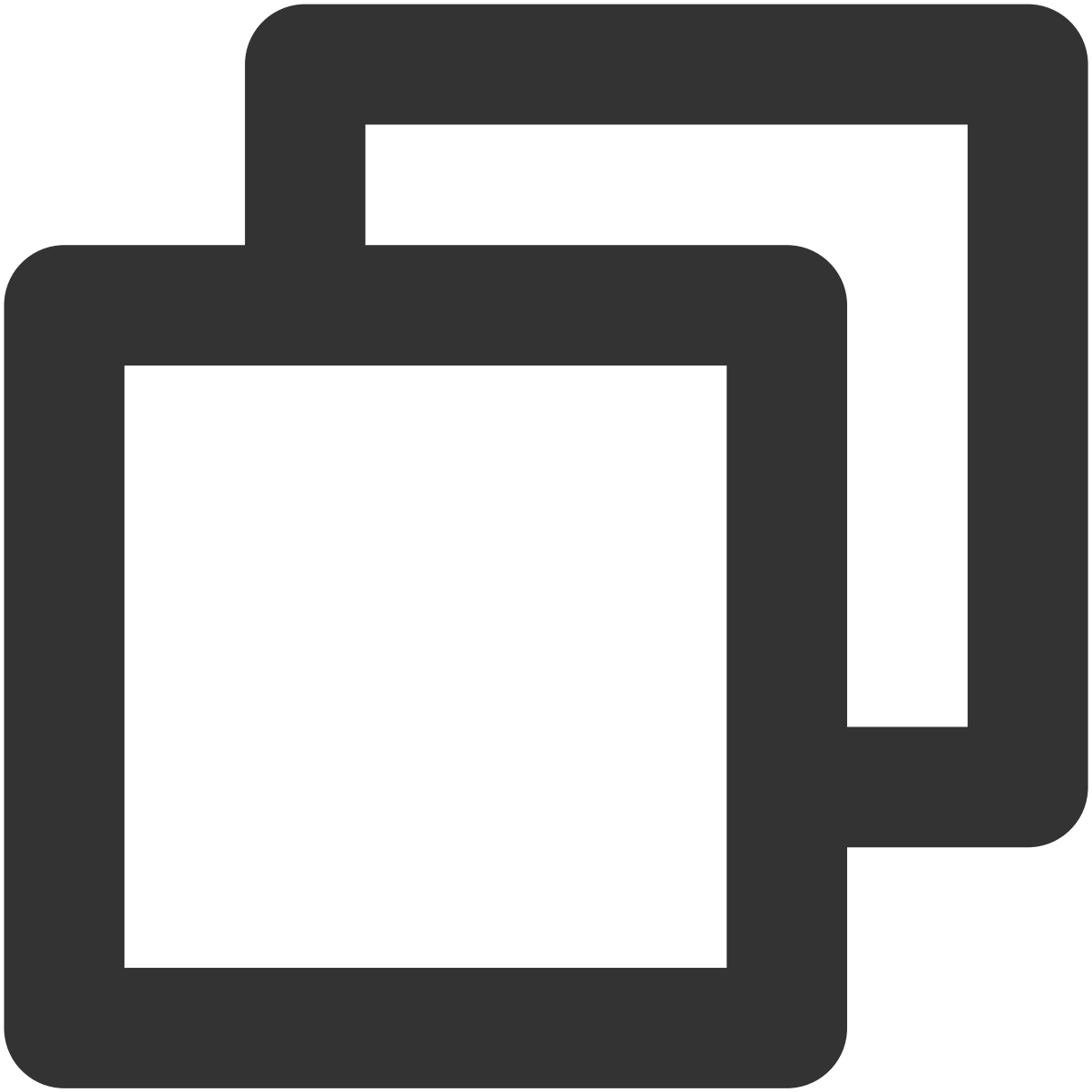
多文件打包压缩功能可以将您的多个文件，打包为 zip 等压缩包格式，以提交任务的方式进行多文件打包压缩，异步返回打包后的文件，该接口属于 POST 请求。

方法原型



```
public Guzzle\\Service\\Resource\\Model createFileCompressJobs(array $args = array(
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53995 提交多文件打包压缩任务-异
    $result = $cosClient->createFileCompressJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'FileCompress',
//        'QueueId' => 'pcc3ae89sa9d807fs89dg789sdg',
        'Operation' => array(
            'UserData' => 'xxx',
            'FileCompressConfig' => array(
                'Flatten' => '0',
                'Format' => 'zip',
//                'UrlList' => 'test/index.csv',
//                'Prefix' => 'test/',
                'Keys' => array(
                    'object1', // 待压缩桶文件
                    'object2', // 待压缩桶文件
                    'object3', // 待压缩桶文件
                ),
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
                'Object' => 'output/test.zip',
            ),
        ),
//        'CallBackFormat' => '',
//        'CallBackType' => '',
//        'CallBack' => '',
//        'CallBackMqConfig' => array(
//            'MqRegion' => '',
//            'MqMode' => '',
//            'MqName' => '',
//        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，多文件打包压缩默认为：FileCompress。	String	是
Operation	Request	包含文件打包压缩的处理规则。	Container	是
QueueId	Request	任务所在的队列 ID。	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情请参见 CallBackMqConfig	Container	否

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileCompressConfig	Request.Operation	指定文件打包压缩的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
Output	Request.Operation	指定打包压缩后的文件保存的地址信息。	Container	是

Container 类型 FileCompressConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型
Flatten	Request.Operation.FileCompressConfig	文件打包时，是否需要去除源文件已有的目录结构，有效值： 0：不需要去除目录结构，打包后压缩包中的文件会保留原有的目录结构；	String

版权所有：腾讯云计算（北京）有限责任公司

		1：需要，打包后压缩包内的文件会去除原有的目录结构，所有文件都在同一层级。 例如：源文件 URL 为 `https://domain/source/test.mp4`，则源文件路径为 source/test.mp4，如果为 1，则 ZIP 包中该文件路径为 test.mp4；如果为0， ZIP 包中该文件路径为 source/test.mp4。	
Format	Request.Operation.FileCompressConfig	打包压缩的类型，有效值：zip、tar、tar.gz。	String
UrlList	Request.Operation.FileCompressConfig	支持将需要打包的文件整理成索引文件，后台将根据索引文件内提供的文件url，打包为一个压缩包文件。索引文件需要保存在当前存储桶中，本字段需要提供索引文件的对象地址，例如：/test/index.csv。 索引文件格式：仅支持 CVS文件，一行一条 URL（仅支持本存储桶文件），如有多列字段，默认取第一列作为URL。最多不超过10000个文件, 总大小不超过50G, 否则会导致任务失败。	String
Prefix	Request.Operation.FileCompressConfig	支持对存储桶中的某个前缀进行打包，如果需要对某个目录进行打包，需要加/，例如test目录打包，则值为：test/。最多不超过10000个文件，总大小不超过50G，否则会导致任务失败。	String
Key	Request.Operation.FileCompressConfig	支持对存储桶中的多个文件进行打包，个数不能超过 1000, 总大小不超过50G，否则会导致任务失败。	String 数组

注意：

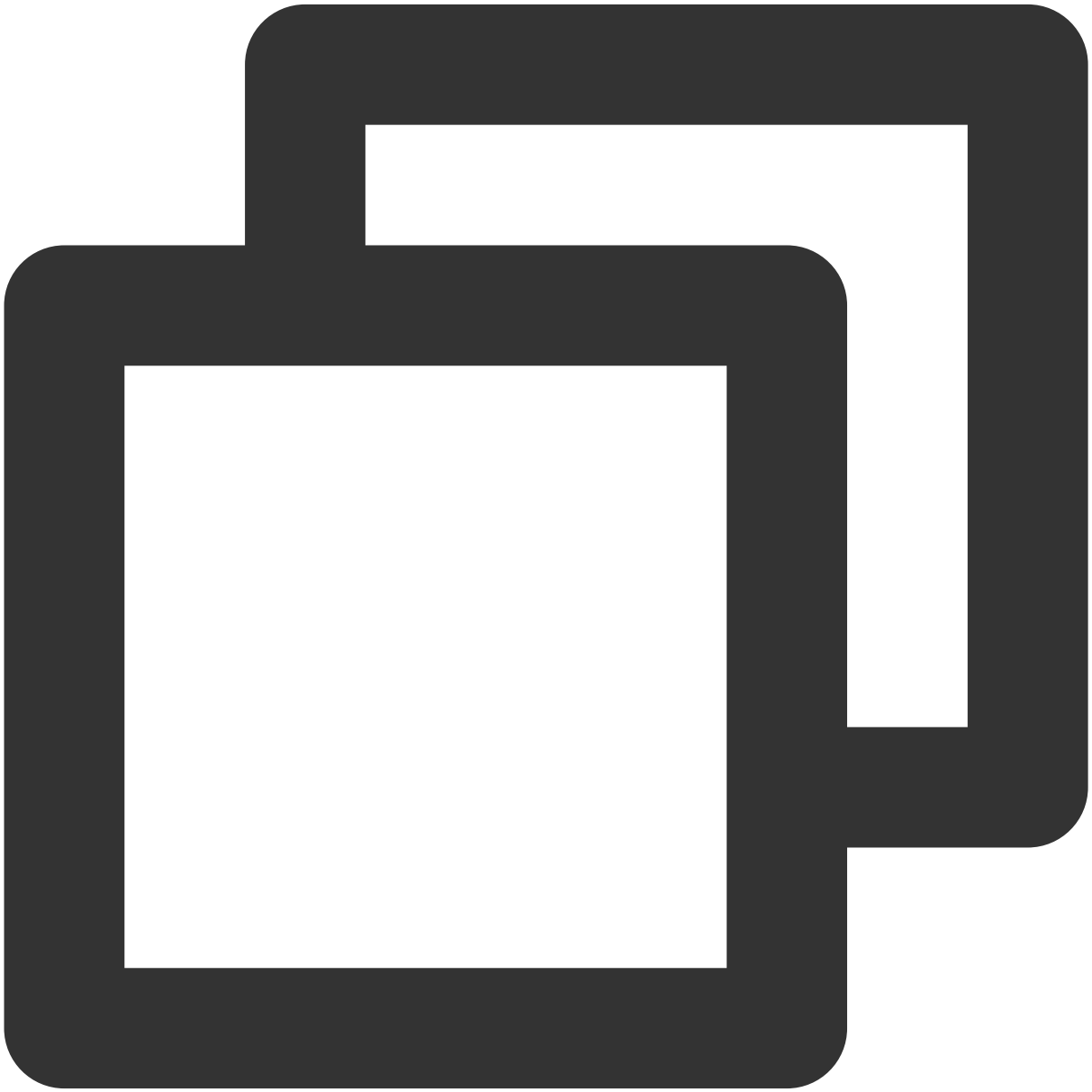
UrlList、Prefix、Key 三者仅能选择一个，不能都为空，也不会同时生效。如果填了多个，会按优先级 UrlList > Prefix > Key 取最高优先级执行。

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域。	String	是
Bucket	Request.Operation.Output	保存压缩后文件的存储桶。	String	是

Object	Request.Operation.Output	压缩后文件的文件名	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 827
    [JobsDetail] => Array
```

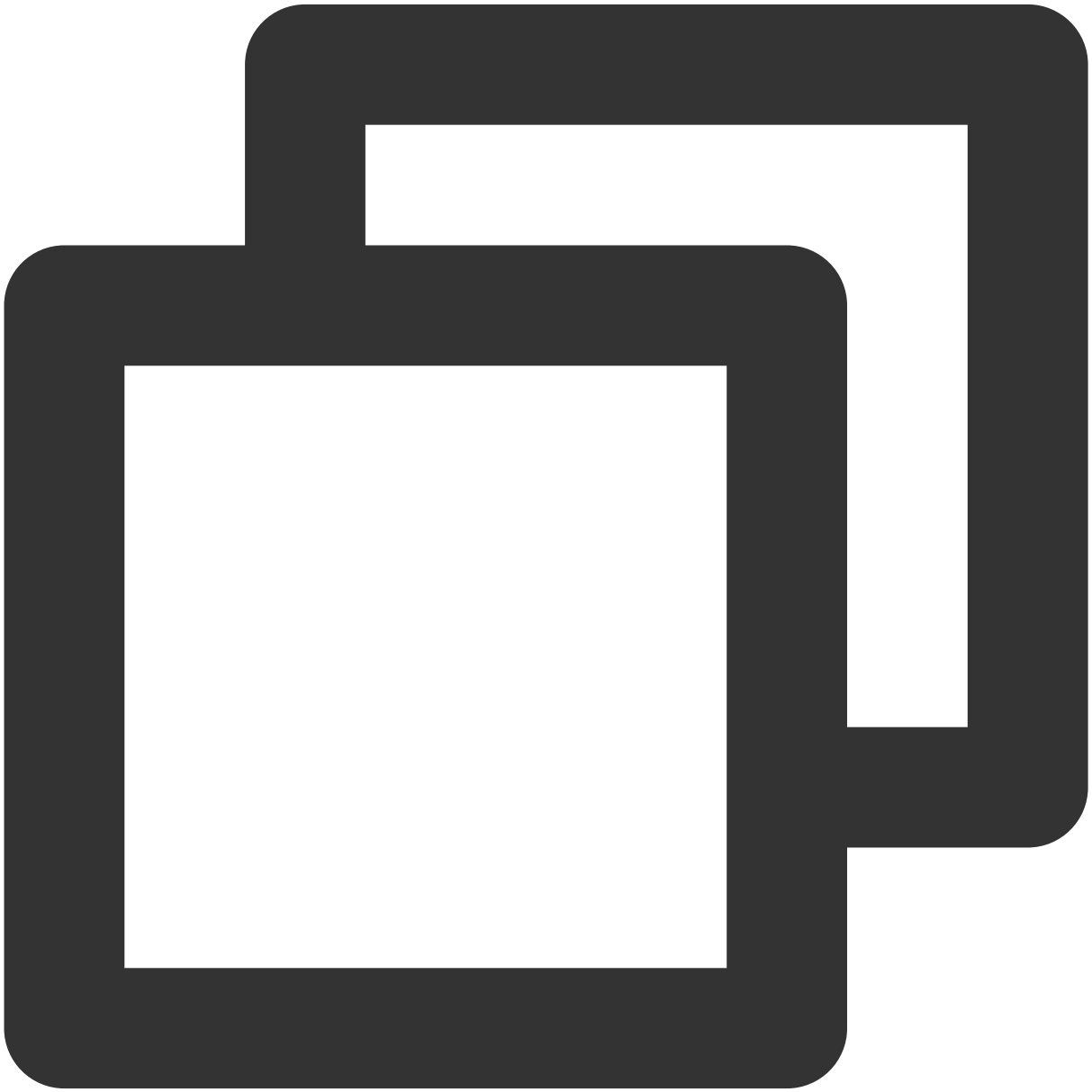
```
        (
            [Progress] => 0
            [Code] => Success
            [Message] =>
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
            [Tag] => FileCompress
            [State] => Submitted
            [CreationTime] => 2023-01-31T16:17:21+0800
            [StartTime] => -
            [EndTime] => -
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [Output] => Array
                        (
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                            [Object] => tmp/test.zip
                        )

                    [FileCompressConfig] => Array
                        (
                            [Flatten] => 0
                            [Format] => zip
                            [UrlList] =>
                            [Prefix] =>
                            [Key] => Array
                                (
                                    [0] => test1.png
                                    [1] => test2.jpeg
                                )

                        )

                )

        )

    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs
)
```

查询多文件打包压缩结果

功能说明

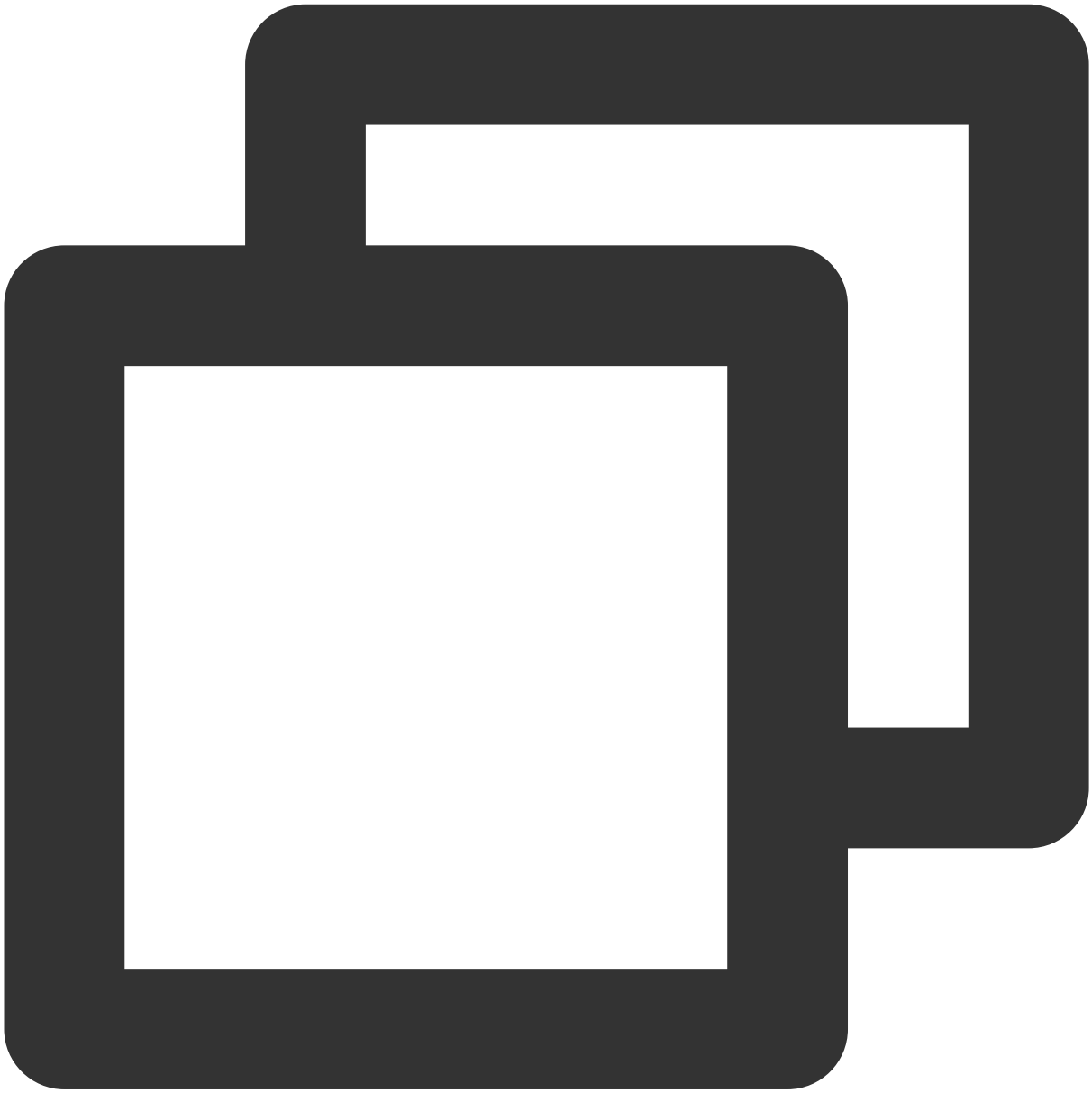
本接口用于主动查询指定的多文件打包压缩任务结果。

方法原型



```
public Guzzle\\Service\\Resource\\Model getFileCompressResult(array $args = array()
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
```

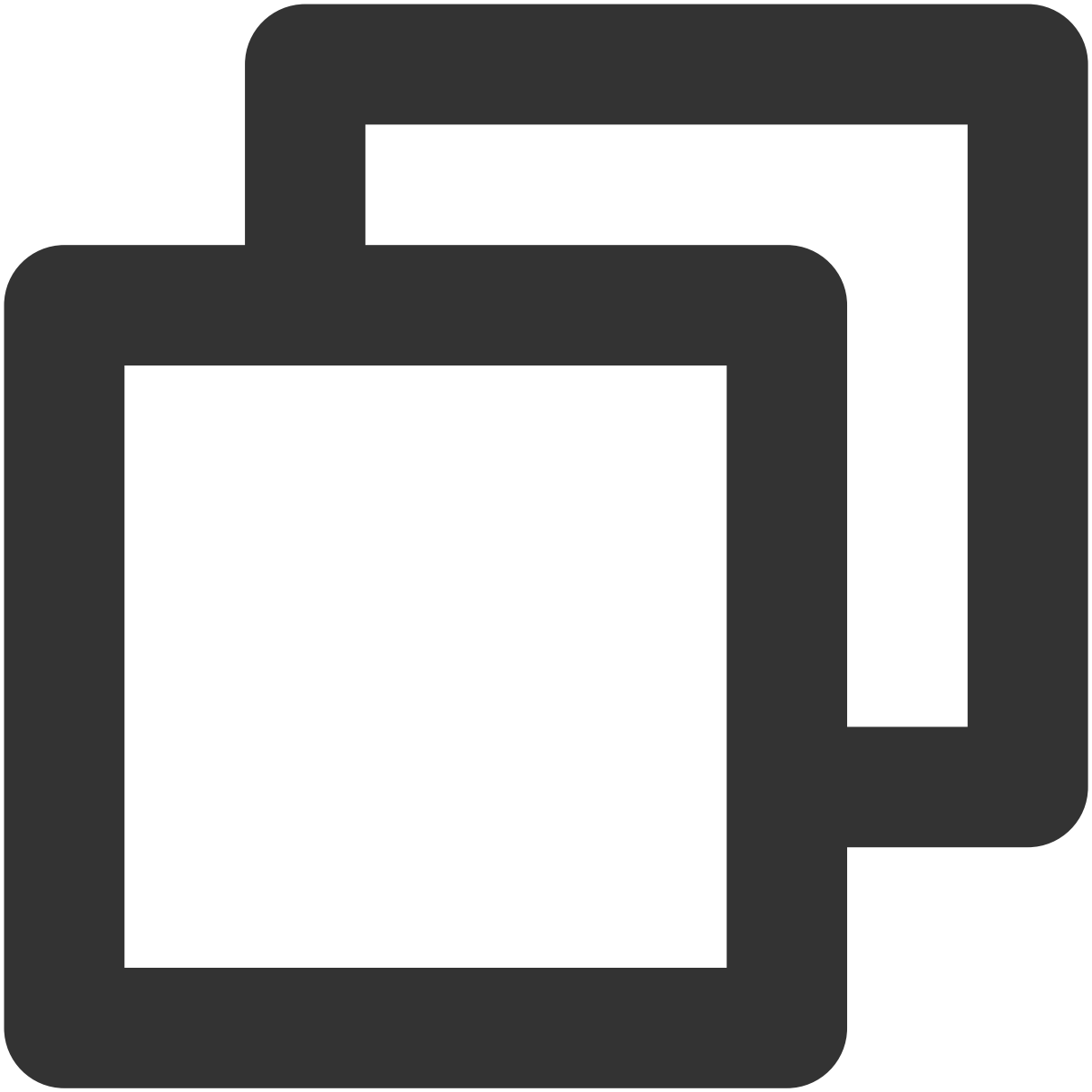
```
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53996 查询多文件打包压缩结果
    $result = $cosClient->getFileCompressResult(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => '', // jobId
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	jobId	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 1093
    [JobsDetail] => Array
        (
            [Progress] => 100
            [Code] => Success
            [Message] => success
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
```

```
            [Tag] => FileCompress
            [State] => Success
            [CreationTime] => 2023-01-31T16:17:21+0800
            [StartTime] => 2023-01-31T16:17:21+0800
            [EndTime] => 2023-01-31T16:17:22+0800
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [Output] => Array
                        (
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                            [Object] => tmp/test.zip
                        )

                    [FileCompressConfig] => Array
                        (
                            [Flatten] => 0
                            [Format] => zip
                            [UrlList] =>
                            [Prefix] =>
                            [Key] => Array
                                (
                                    [0] => test1.png
                                    [1] => test2.jpeg
                                )

                        )

                    [FileCompressResult] => Array
                        (
                            [CompressFileCount] => 2
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                            [Object] => tmp/test.zip
                        )

                )

        )

    [Key] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs/f
)
```

版权所有：腾讯云计算（北京）有限责任公司

文件哈希值计算

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于文件哈希值计算接口的 API 概览和 SDK 示例代码。

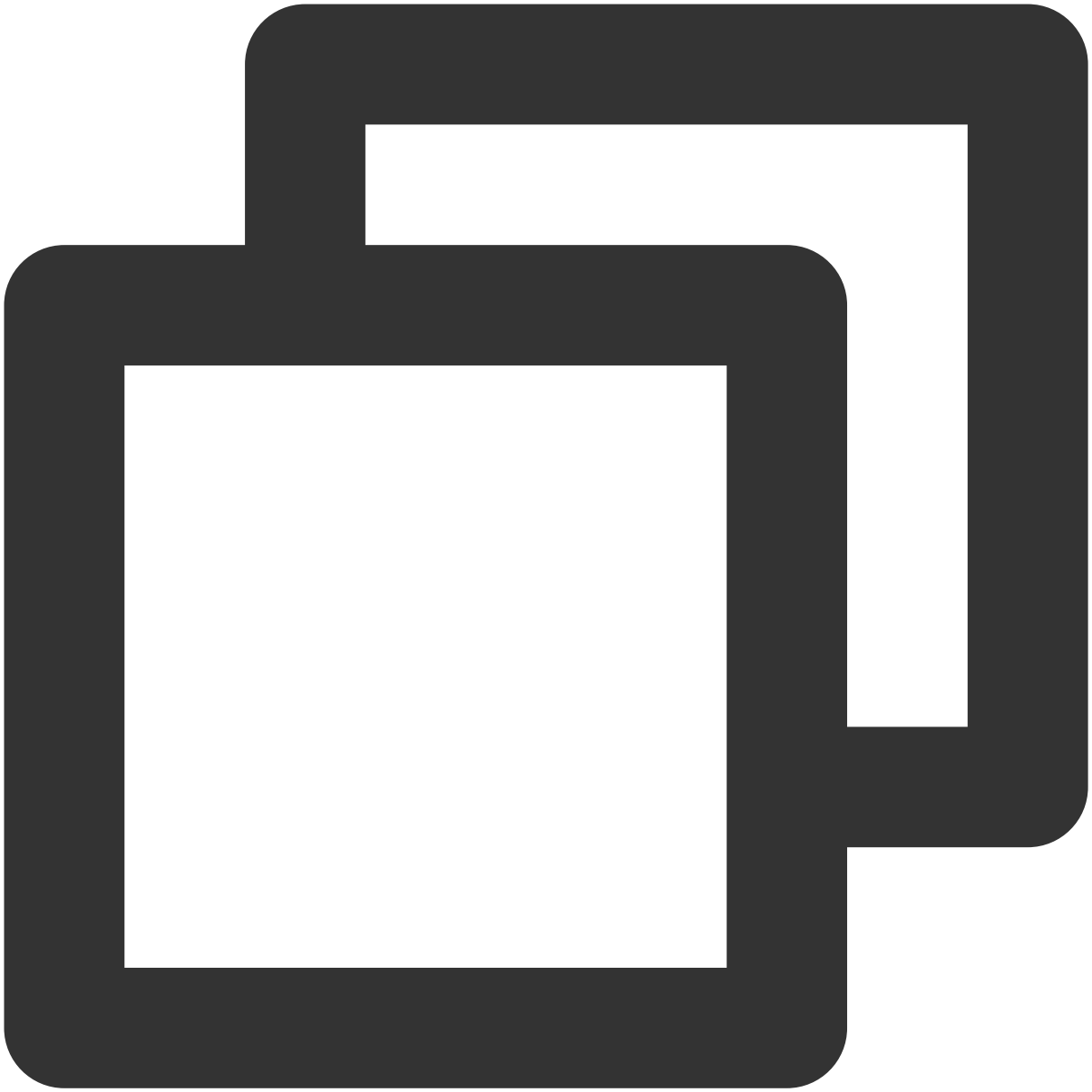
API	操作描述
哈希值计算同步请求	以同步请求的方式进行文件哈希值计算，实时返回计算得到的哈希值，该接口属于 GET 请求。
提交哈希值计算任务	以提交任务的方式进行文件哈希值计算，异步返回计算得到的哈希值，该接口属于 POST 请求。
查询哈希值计算结果	本接口用于主动查询指定的文件哈希值计算任务结果。

哈希值计算同步请求

功能说明

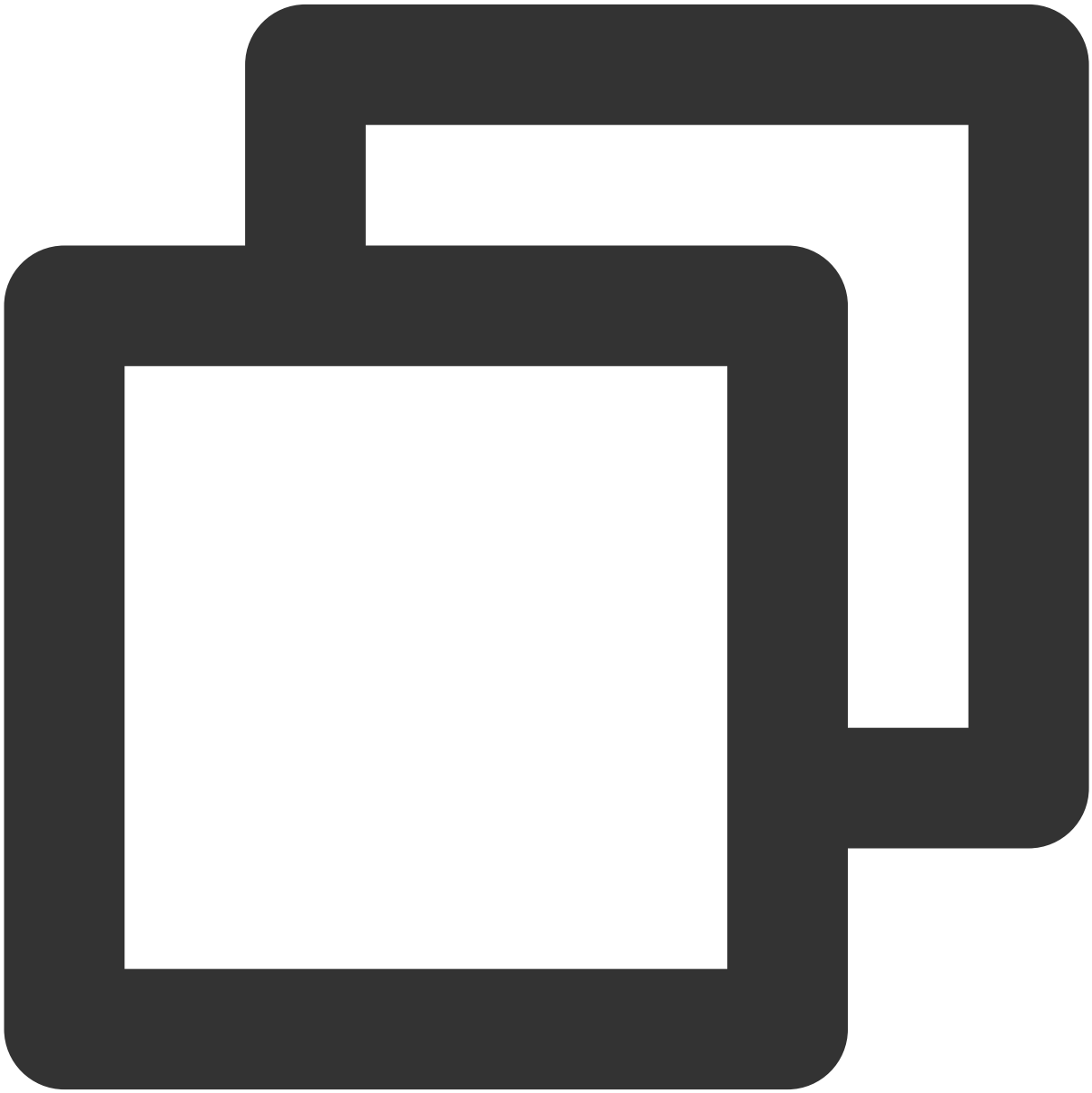
以同步请求的方式进行文件哈希值计算，实时返回计算得到的哈希值，该接口属于 GET 请求。

方法原型



```
public Guzzle\\Service\\Resource\\Model FileJobs4Hash(array $args = array());
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

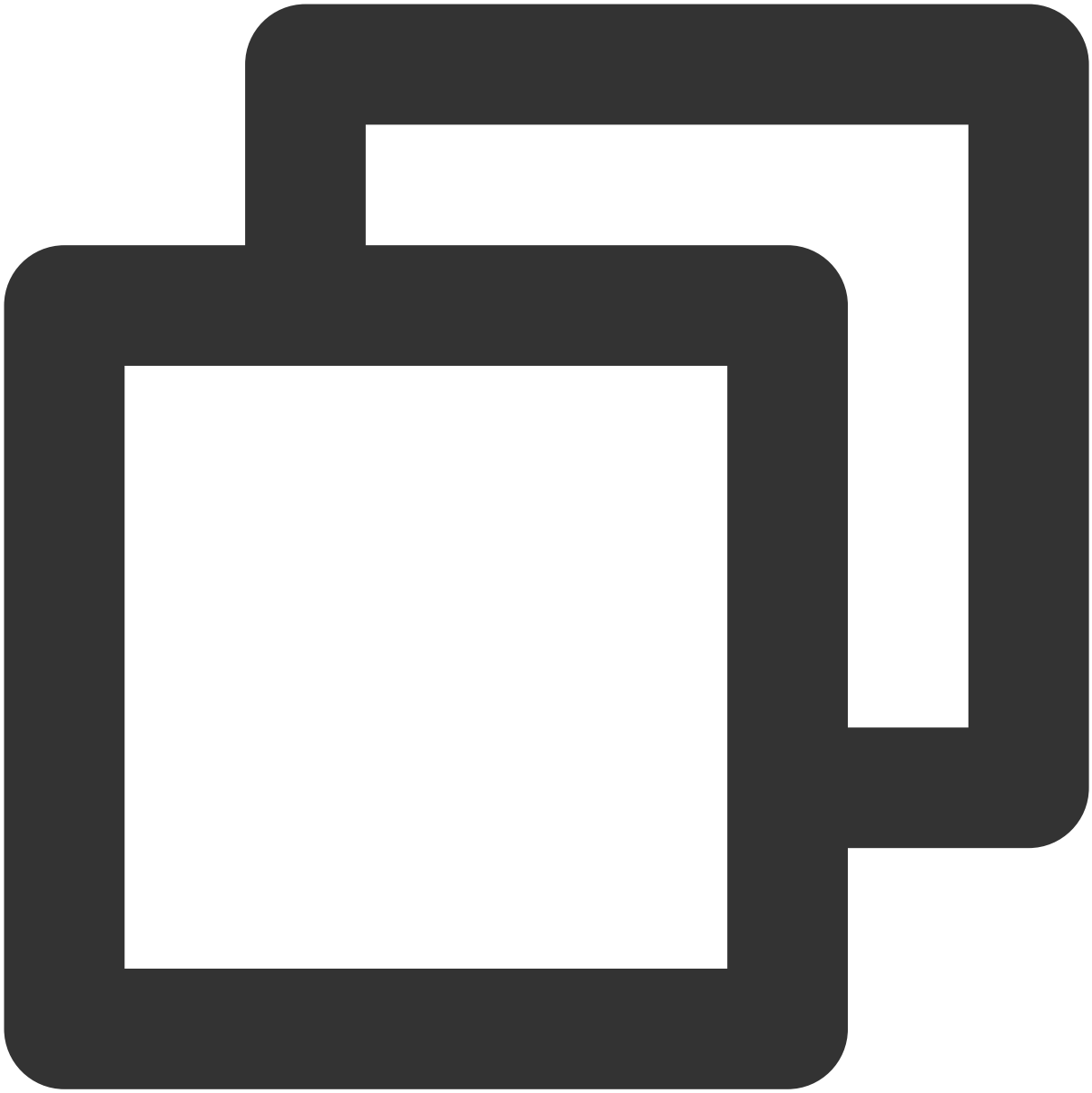
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53990 哈希值计算同步请求
    $result = $cosClient->FileJobs4Hash(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => 'for-test.mp4', // 桶文件
        'Type' => 'md5', // 支持的哈希算法类型，有效值：md5、sha1、sha256
//        'AddToHeader' => 'true', // 是否将计算得到的哈希值，自动添加至文件的自定义 header
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
    /**
     * 可能出现的 Exception
     * 1. Error Message: file processing is not active yet, please apply for file p
     *    解决方式：开通文件处理服务
     */
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	桶文件	是
Type	String	支持的哈希算法类型，有效值：md5、sha1、sha256	是
AddToHeader	String	是否将计算得到的哈希值，自动添加至文件的自定义 header，格式为：x-cos-meta-md5/sha1/sha256; 有效值： true、false，不填则默认为 false。	否

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI
    [ContentType] => application/xml
    [ContentLength] => 421
    [FileHashCodeResult] => Array
        (
            [Etag] => "1f3ede740849eaa2e79e9b269ea3db2a"
            [LastModified] => 2023-01-29T11:31:02+0000
            [MD5] => 1f3ede740849eaa2e79e9b269ea3db2a
            [FileSize] => 42120
```

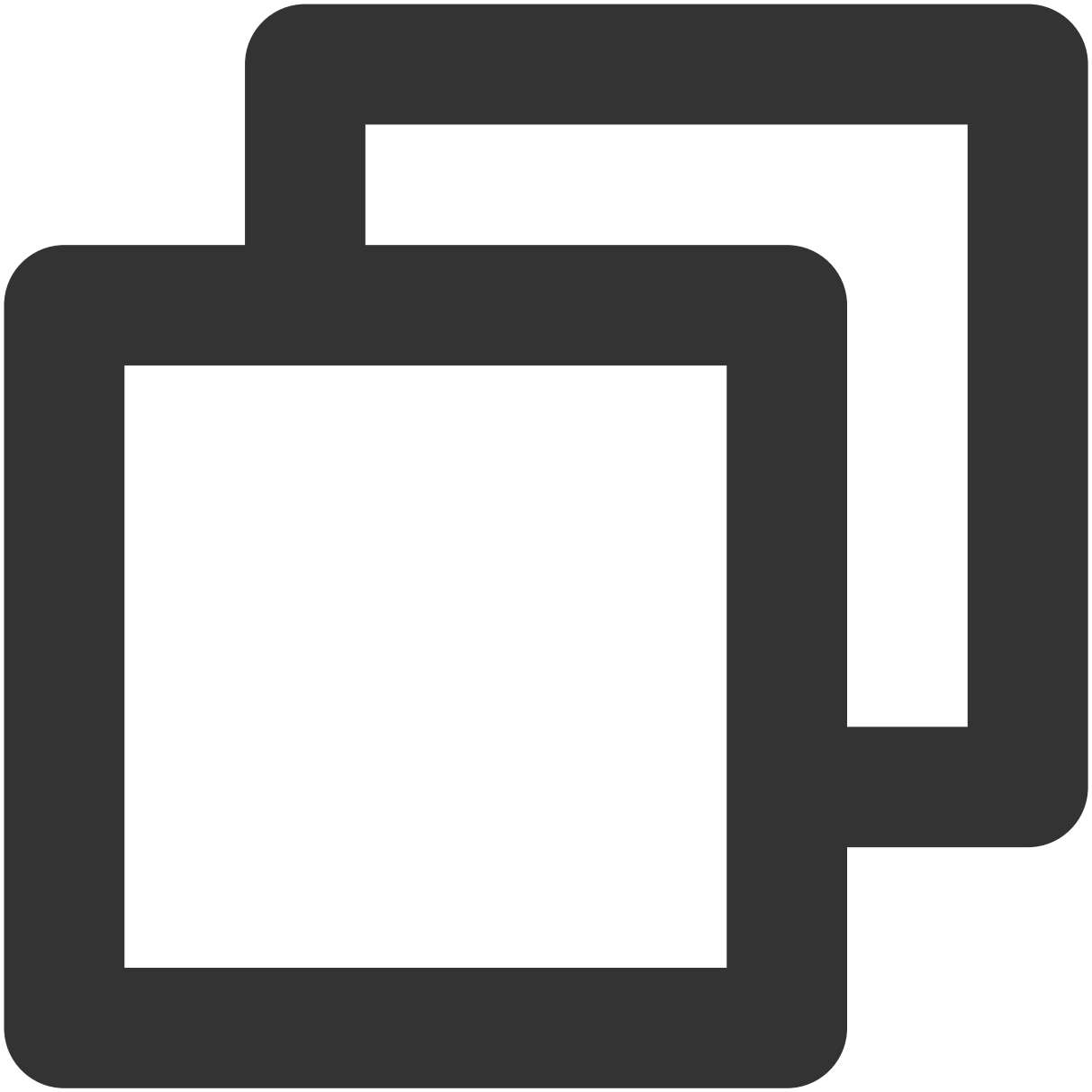
```
        )

    [Input] => Array
        (
            [Region] => ap-guangzhou
            [Bucket] => examplebucket-1250000000
            [Object] => sound01.mp3
        )

    [Key] => sound01.mp3
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sound01.mp
)
```

提交哈希值计算任务

功能说明

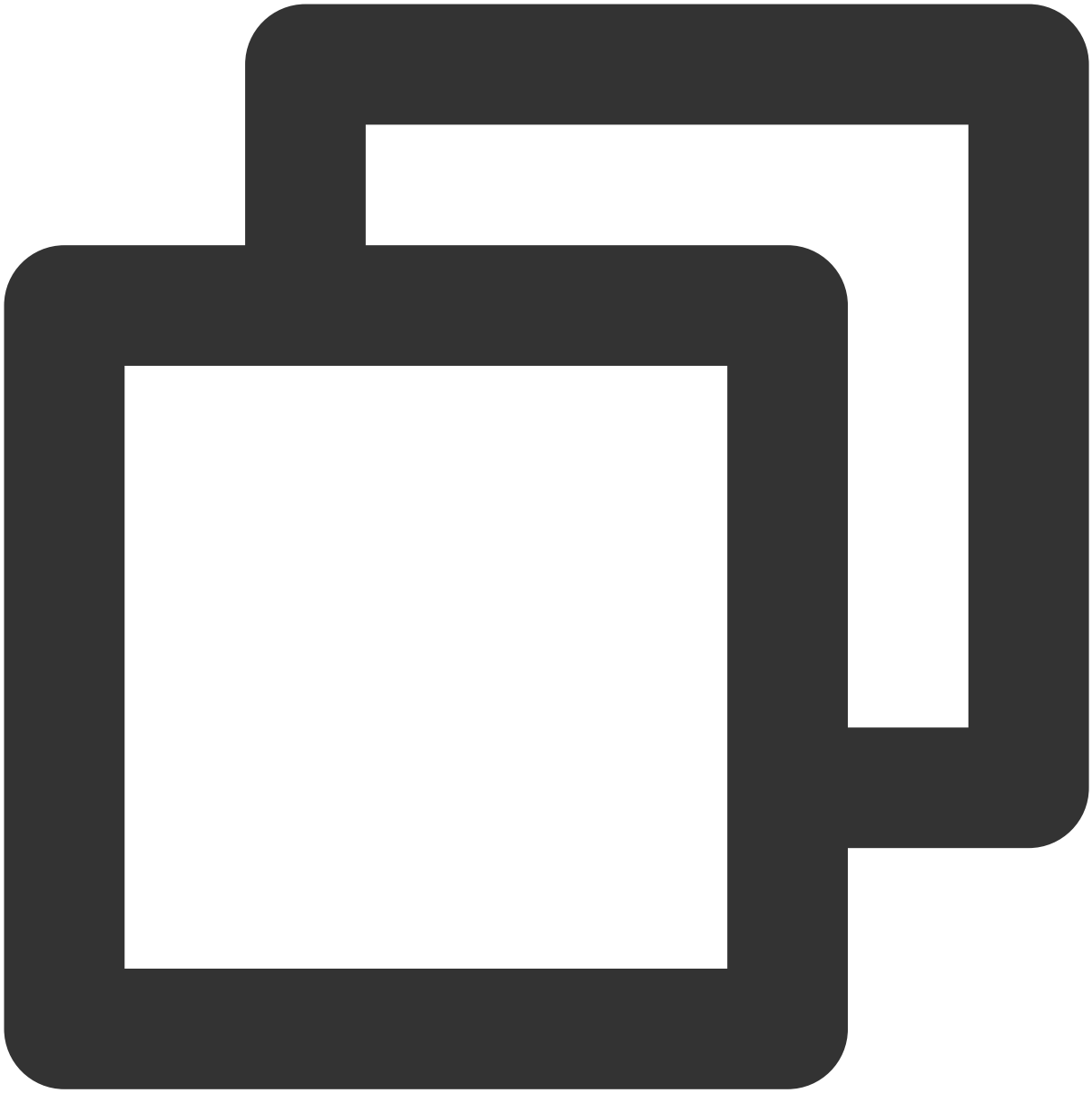
以提交任务的方式进行文件哈希值计算，异步返回计算得到的哈希值，该接口属于 POST 请求。

方法原型



```
public Guzzle\\Service\\Resource\\Model createFileHashCodeJobs(array $args = array(
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53991 提交哈希值计算任务-异步
    $result = $cosClient->createFileHashCodeJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'FileHashCode',
//        'QueueId' => 'pcc3ae89sa9d807fs89dg789sdg',
        'Input' => array(
            'Object' => 'test.mp4',
        ),
        'Operation' => array(
            'UserData' => 'xxx',
            'FileHashCodeConfig' => array(
                'Type' => 'MD5',
                'AddToHeader' => 'true',
            ),
        ),
//        'CallBackFormat' => '',
//        'CallBackType' => '',
//        'CallBack' => '',
//        'CallBackMqConfig' => array(
//            'MqRegion' => '',
//            'MqMode' => '',
//            'MqName' => '',
//        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，哈希值计算默认为：FileHashCode。	String	是
Input	Request	包含待操作的文件信息。	Container	是

Operation	Request	包含哈希值计算的处理规则。	Container	是
QueueId	Request	任务所在的队列 ID。	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	文件名，取值为文件在当前存储桶中的完整名称。	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileHashCodeConfig	Request.Operation	指定哈希值计算的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否

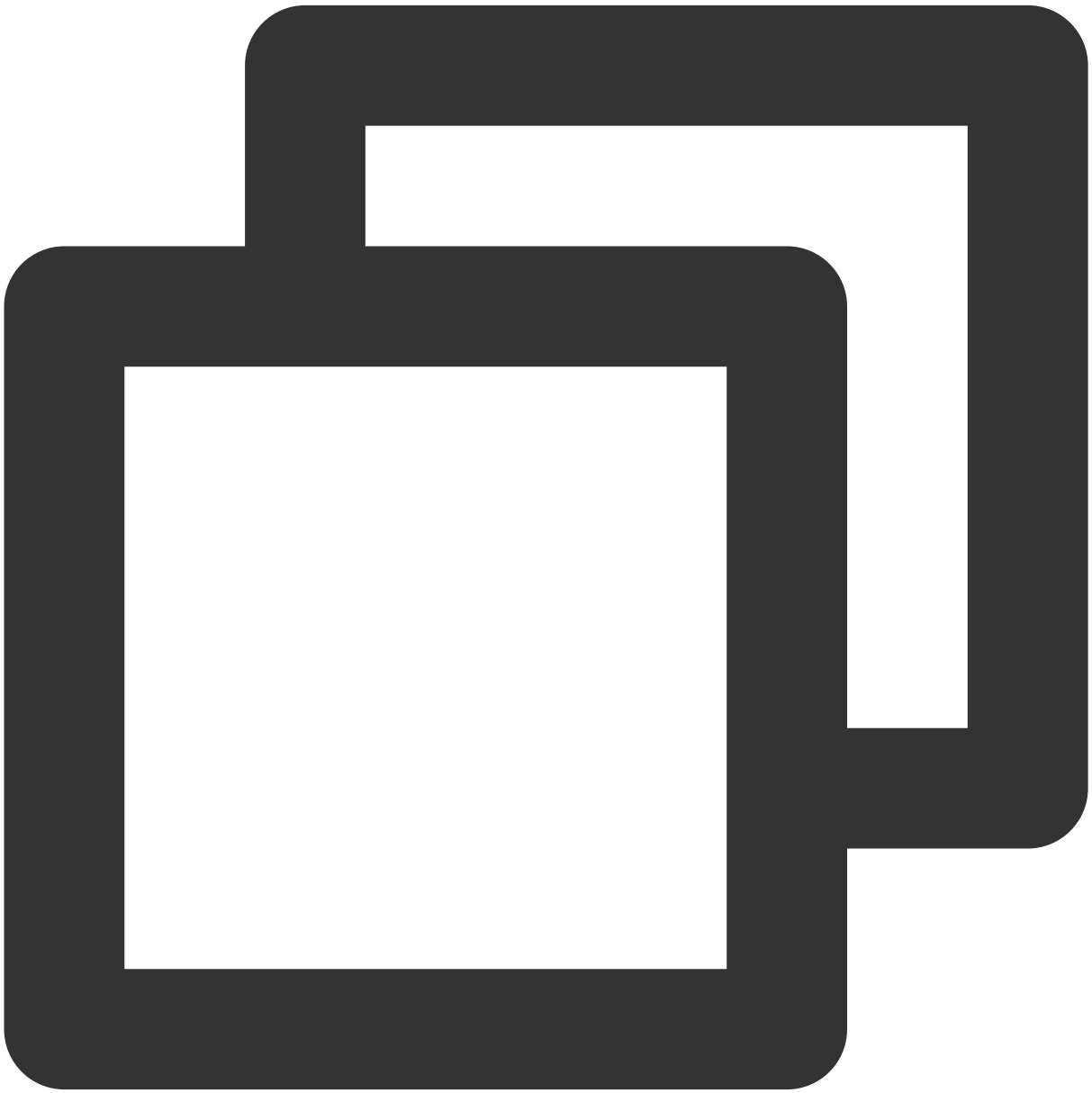
Container 类型 FileHashCodeConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Type	Request.Operation.FileHashCodeConfig	哈希值的算法类型，有效值：MD5、SHA1、SHA256。	String	是
AddToHeader	Request.Operation.FileHashCodeConfig	是否将计算得到的哈希值添加至文件自定义header, 有	String	否

版权所有：腾讯云计算（北京）有限责任公司

		效值：true、false，默认值为 false。 自定义header根据 `Type` 的值变化，例如 `Type` 值为MD5时，自定义header为 x-cos-meta-md5。		

返回结果示例



```
GuzzleHttp\\Command\\Result Object
```

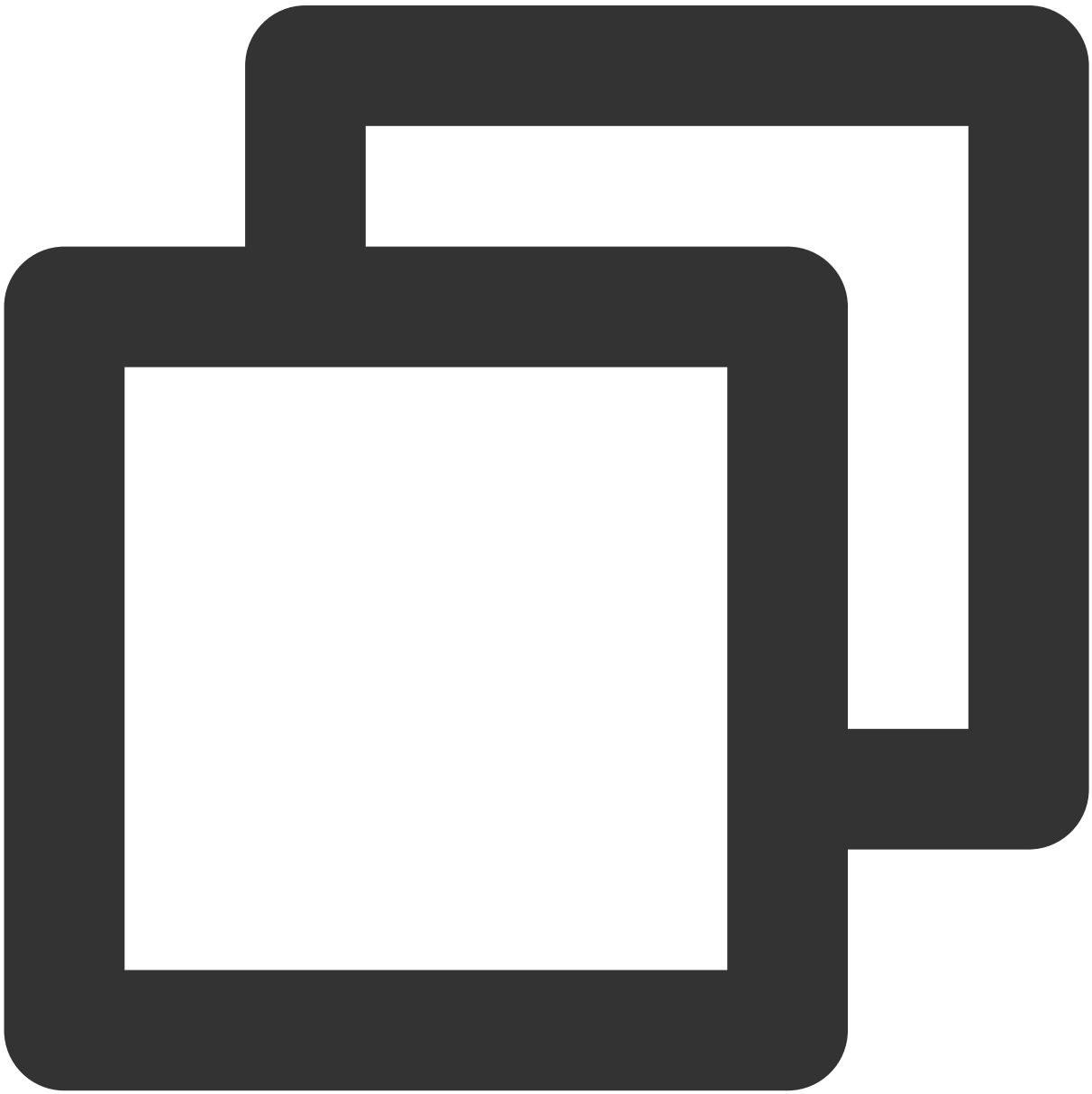
```
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 738
    [JobsDetail] => Array
        (
            [Progress] => 0
            [Code] => Success
            [Message] =>
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
            [Tag] => FileHashCode
            [State] => Submitted
            [CreationTime] => 2023-01-31T15:49:33+0800
            [StartTime] => -
            [EndTime] => -
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Input] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [Object] => sound01.mp3
                )

            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [FileHashCodeConfig] => Array
                        (
                            [Type] => MD5
                            [AddToHeader] => true
                        )

                )

        )

    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs
)
```

查询哈希值计算结果

功能说明

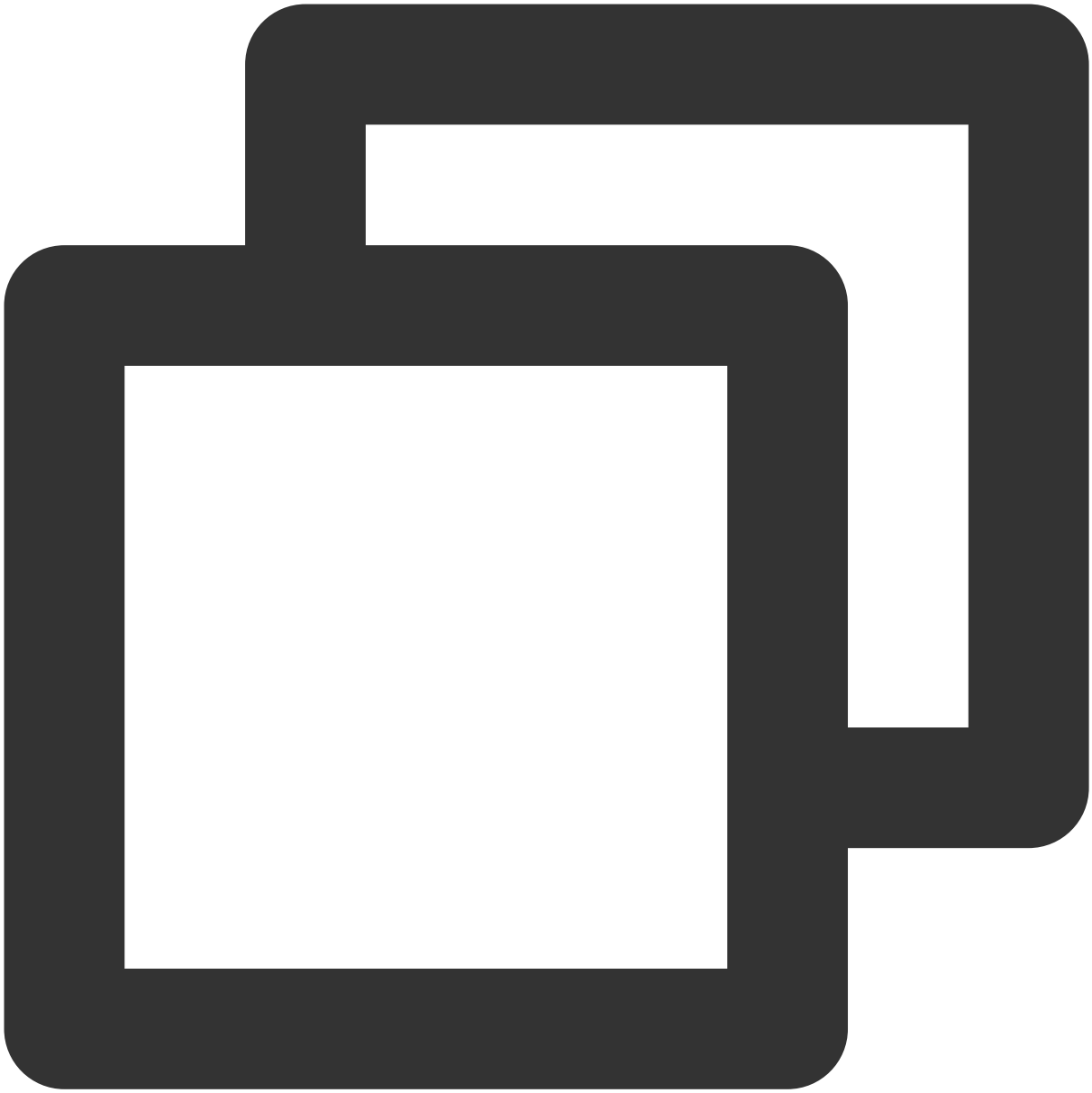
本接口用于主动查询指定的文件哈希值计算任务结果。

方法原型



```
public Guzzle\\Service\\Resource\\Model getFileHashCodeResult(array $args = array()
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

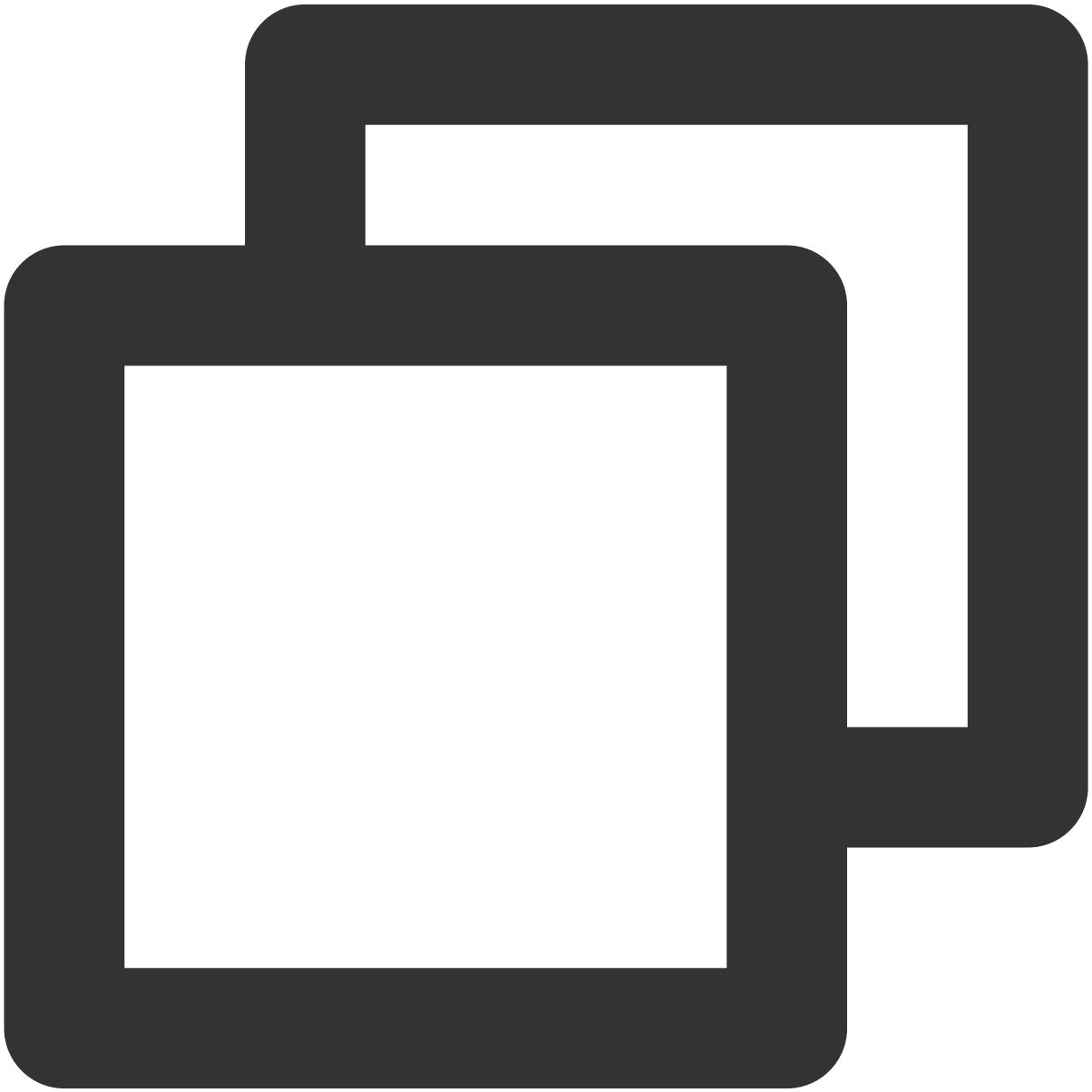
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53992 查询哈希值计算结果
    $result = $cosClient->getFileHashCodeResult(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => '', // jobId
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	jobId	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 1048
    [JobsDetail] => Array
        (
            [Progress] => 100
            [Code] => Success
            [Message] => success
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
```

```
            [Tag] => FileHashCode
            [State] => Success
            [CreationTime] => 2023-01-31T15:49:33+0800
            [StartTime] => 2023-01-31T15:49:33+0800
            [EndTime] => 2023-01-31T15:49:34+0800
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Input] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [Object] => sound01.mp3
                )

            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [FileHashCodeConfig] => Array
                        (
                            [Type] => MD5
                            [AddToHeader] => true
                        )

                    [FileHashCodeResult] => Array
                        (
                            [MD5] => 1f3ede740849eaa2e79e9b269ea3db2a
                            [LastModified] => 2023-01-29T11:31:02+0000
                            [Etag] => "1f3ede740849eaa2e79e9b269ea3db2a"
                            [FileSize] => 42120
                        )

                )

        )

    [Key] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs/f
)
```

文件解压

最近更新时间：2024-03-13 10:25:33

简介

本文档提供关于文件解压接口的 API 概览和 SDK 示例代码。

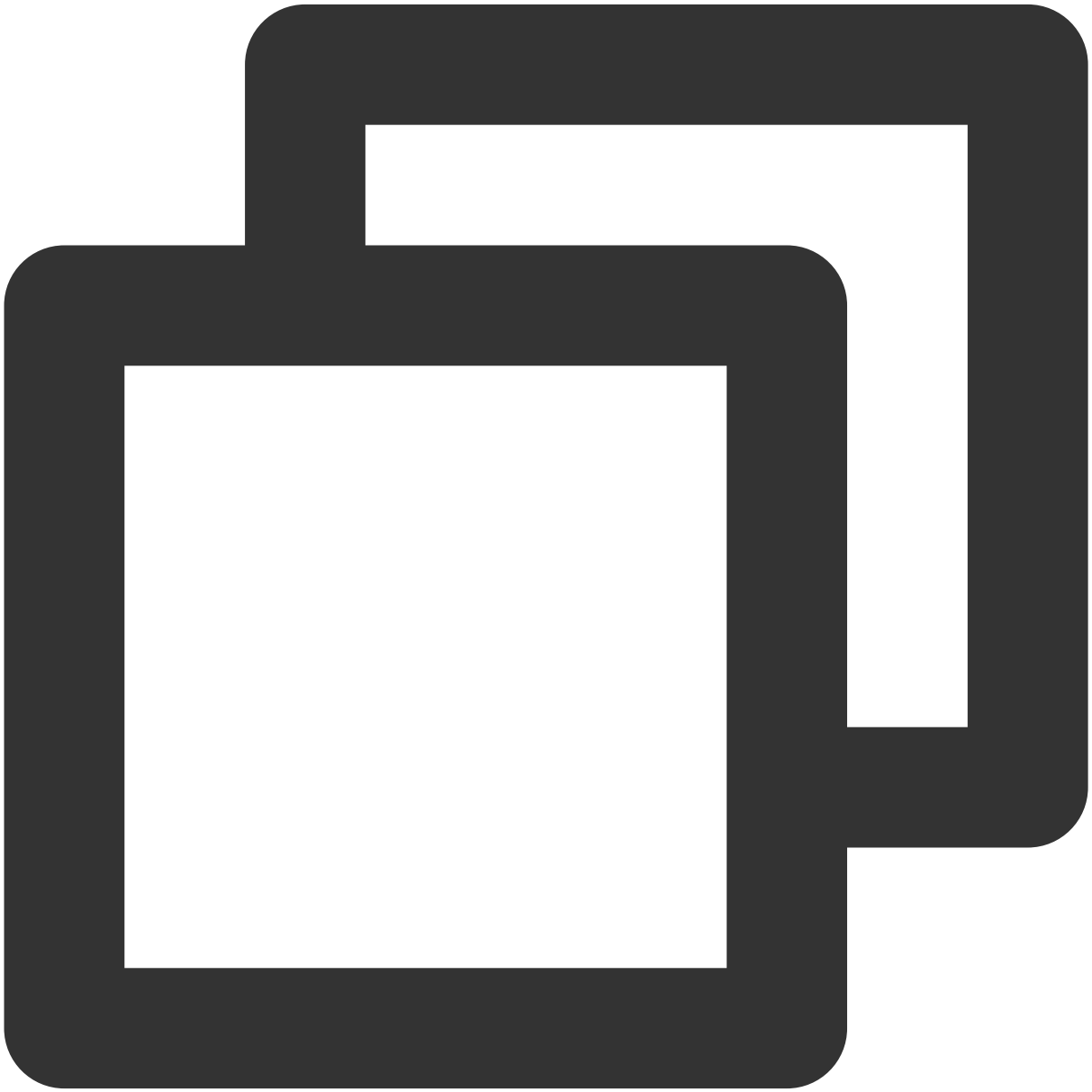
API	操作描述
提交文件解压任务	以提交任务的方式进行文件解压缩，异步返回压缩包内被解压出来的文件，该接口属于 POST 请求。
查询文件解压结果	本接口用于主动查询指定的文件解压任务结果。

提交文件解压任务

功能说明

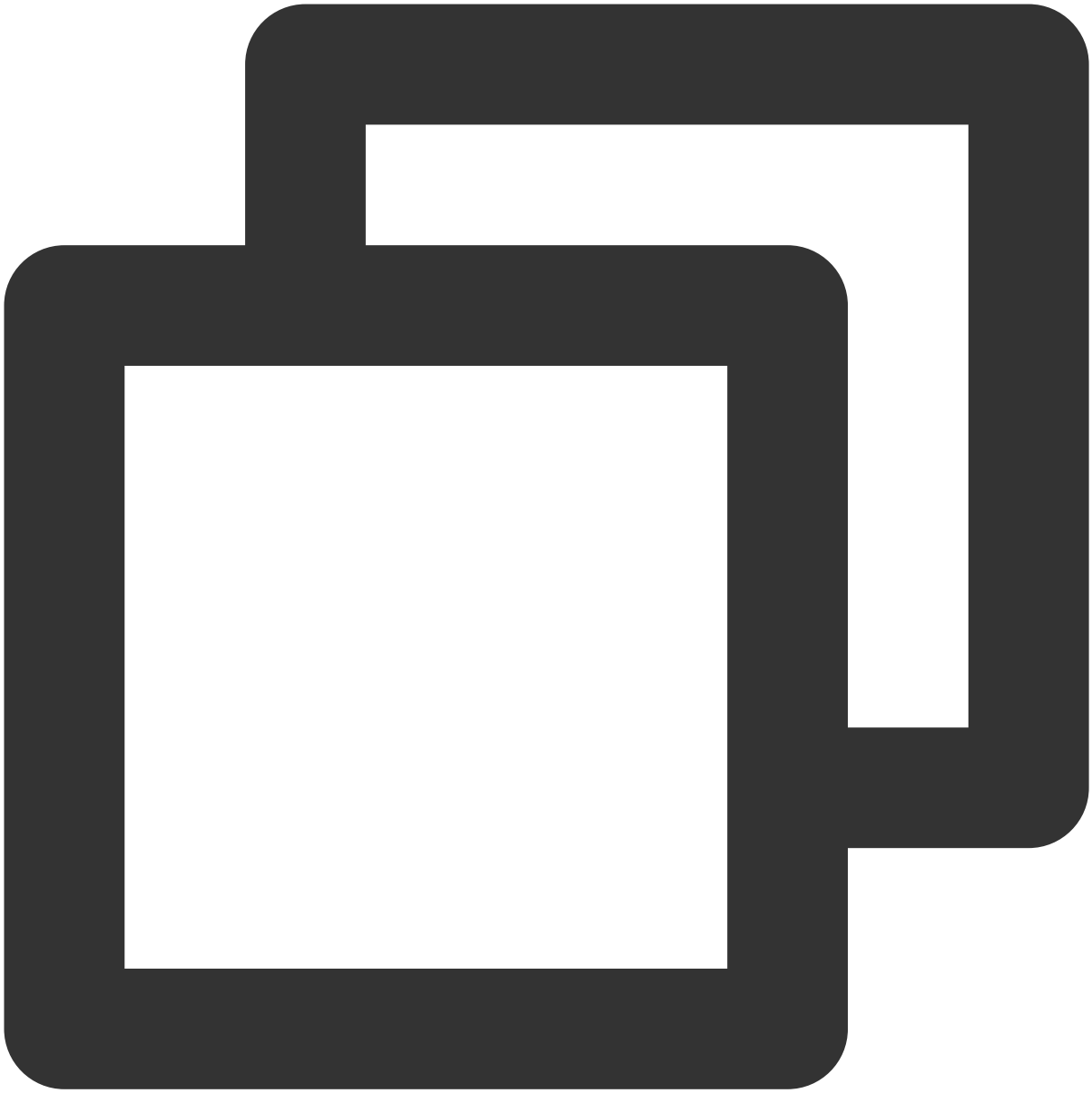
以提交任务的方式进行文件解压缩，异步返回压缩包内被解压出来的文件，该接口属于 POST 请求。

方法原型



```
public Guzzle\\Service\\Resource\\Model createFileUncompressJobs(array $args = arra
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为http
```

```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53993 提交文件解压任务-异步
    $result = $cosClient->createFileUncompressJobs(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Tag' => 'FileUncompress',
//        'QueueId' => 'pcc3ae89sa9d807fs89dg789sdg',
        'Input' => array(
            'Object' => 'test.zip',
        ),
        'Operation' => array(
            'UserData' => 'xxx',
            'FileUncompressConfig' => array(
                'Prefix' => 'prefix',
                'PrefixReplaced' => '1',
            ),
            'Output' => array(
                'Region' => $region,
                'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Ap
            ),
        ),
//        'CallBackFormat' => '',
//        'CallBackType' => '',
//        'CallBack' => '',
//        'CallBackMqConfig' => array(
//            'MqRegion' => '',
//            'MqMode' => '',
//            'MqName' => '',
//        ),
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Tag	Request	表示任务的类型，文件解压默认为：	String	是

		FileUncompress。		
Input	Request	包含待操作的文件信息。	Container	是
Operation	Request	包含文件解压的处理规则。	Container	是
QueueId	Request	任务所在的队列 ID。	String	是
CallBackFormat	Request	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式。	String	否
CallBackType	Request	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型。	String	否
CallBack	Request	任务回调的地址，优先级高于队列的回调地址。	String	否
CallBackMqConfig	Request	任务回调 TDMQ 配置，当 CallBackType 为 TDMQ 时必填。详情见 CallBackMqConfig	Container	否

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Object	Request.Input	文件名，取值为文件在当前存储桶中的完整名称。	String	是

Container 类型 Operation 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
FileUncompressConfig	Request.Operation	指定文件解压的处理规则。	Container	是
UserData	Request.Operation	透传用户信息, 可打印的 ASCII 码, 长度不超过1024	String	否
Output	Request.Operation	指定解压后的文件保存的存储桶信息。	Container	是

Container 类型 FileUncompressConfig 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否

版权所有：腾讯云计算（北京）有限责任公司

				必选
Prefix	Request.Operation.FileUncompressConfig	指定解压后输出文件的前缀，不填则默认保存在存储桶根路径。	String	否
PrefixReplaced	Request.Operation.FileUncompressConfig	指定解压后的文件路径是否需要替换前缀，有效值： 0：不添加额外的前缀，解压缩将保存在Prefix指定的路径下（不会保留压缩包的名称，仅将压缩包内的文件保存至指定的路径）。 1：以压缩包本身的名称作为前缀，解压缩将保存在Prefix指定的路径下。 2：以压缩包完整路径作为前缀，此时如果不指定Prefix，就是解压到压缩包所在的当前路径（包含压缩包本身名称）。 默认值为0。	String	否

说明：

假设压缩包名为 test.zip，压缩包内的文件为“图片.jpg”，存储在了存储桶 A 的 123 目录下，则压缩包的完整路径为：123/test.zip。

对该压缩包进行解压缩，指定解压缩到存储桶A，并指定 `Prefix` 为 456，则不同的 `PrefixReplaced` 值对应解压后的文件保存情况如下：

0：“图片.jpg”会保存在 456 目录下，“图片.jpg”的完整路径为 456/图片.jpg；

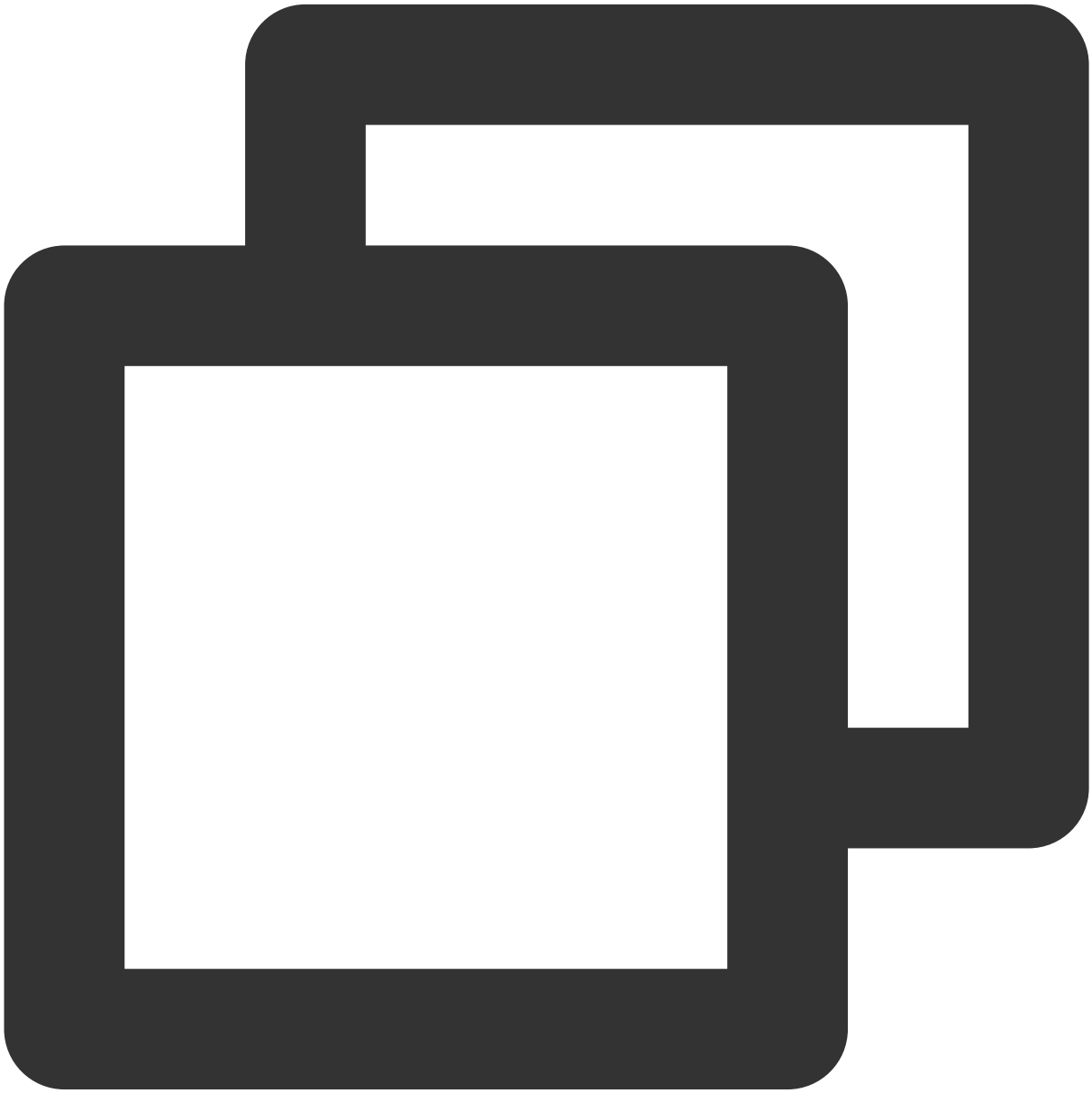
1：“图片.jpg”会以 test 为前缀保存在 456 目录下，“图片.jpg”的完整路径为 456/test/图片.jpg；

2：“图片.jpg”会以 123/test 为前缀保存在 456 目录下，“图片.jpg”的完整路径为 456/123/test/图片.jpg。

Container 类型 Output 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Region	Request.Operation.Output	存储桶的地域。	String	是
Bucket	Request.Operation.Output	保存解压后文件的存储桶。	String	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 872
    [JobsDetail] => Array
        (
            [Progress] => 0
            [Code] => Success
```

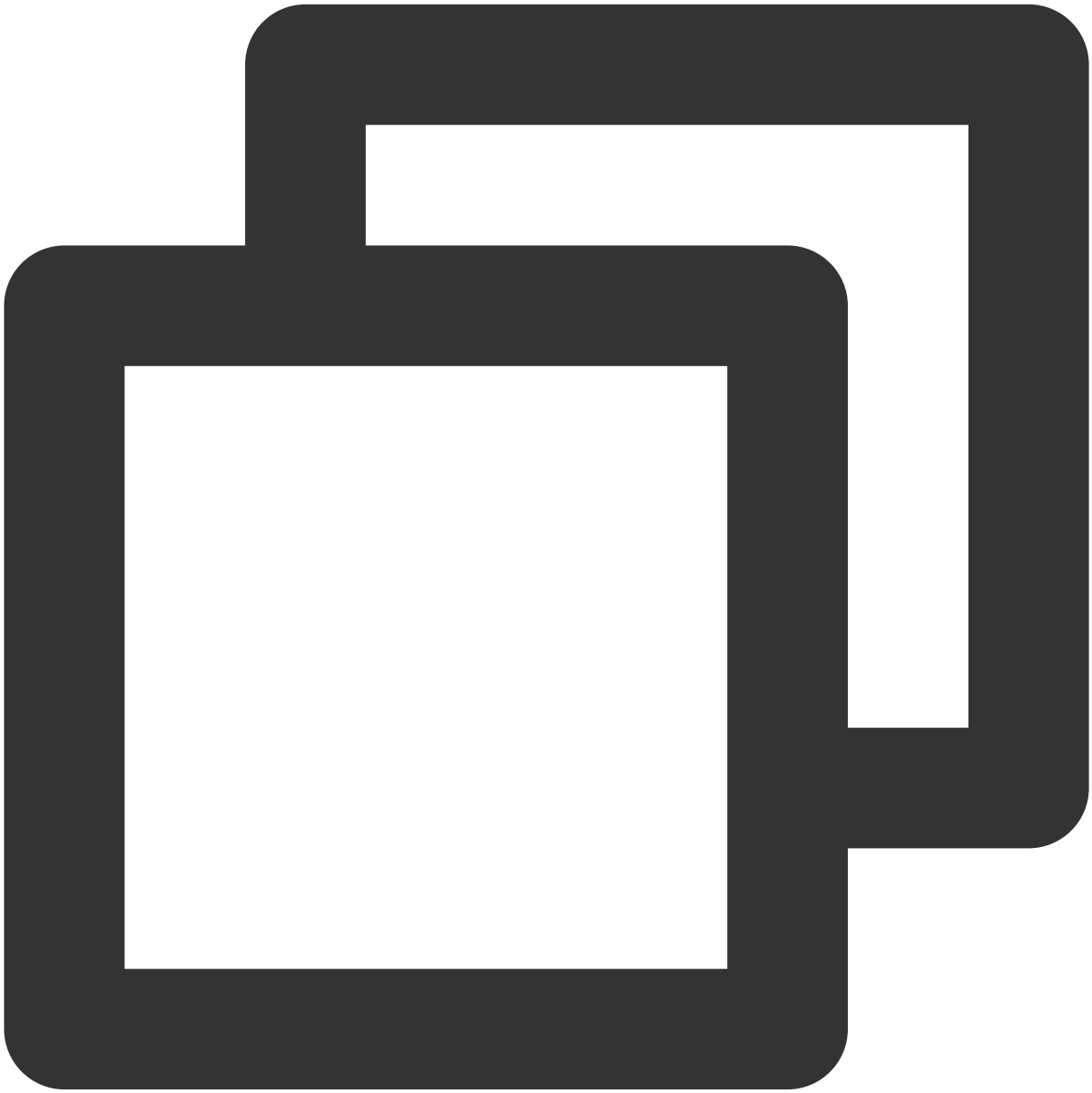
```
            [Message] =>
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
            [Tag] => FileUncompress
            [State] => Submitted
            [CreationTime] => 2023-01-31T16:05:31+0800
            [StartTime] => -
            [EndTime] => -
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Input] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [Object] => test.zip
                )

            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [Output] => Array
                        (
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                        )

                    [FileUncompressConfig] => Array
                        (
                            [Prefix] => prefix
                            [PrefixReplaced] => 1
                        )

                )

        )

    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs
)
```

查询文件解压结果

功能说明

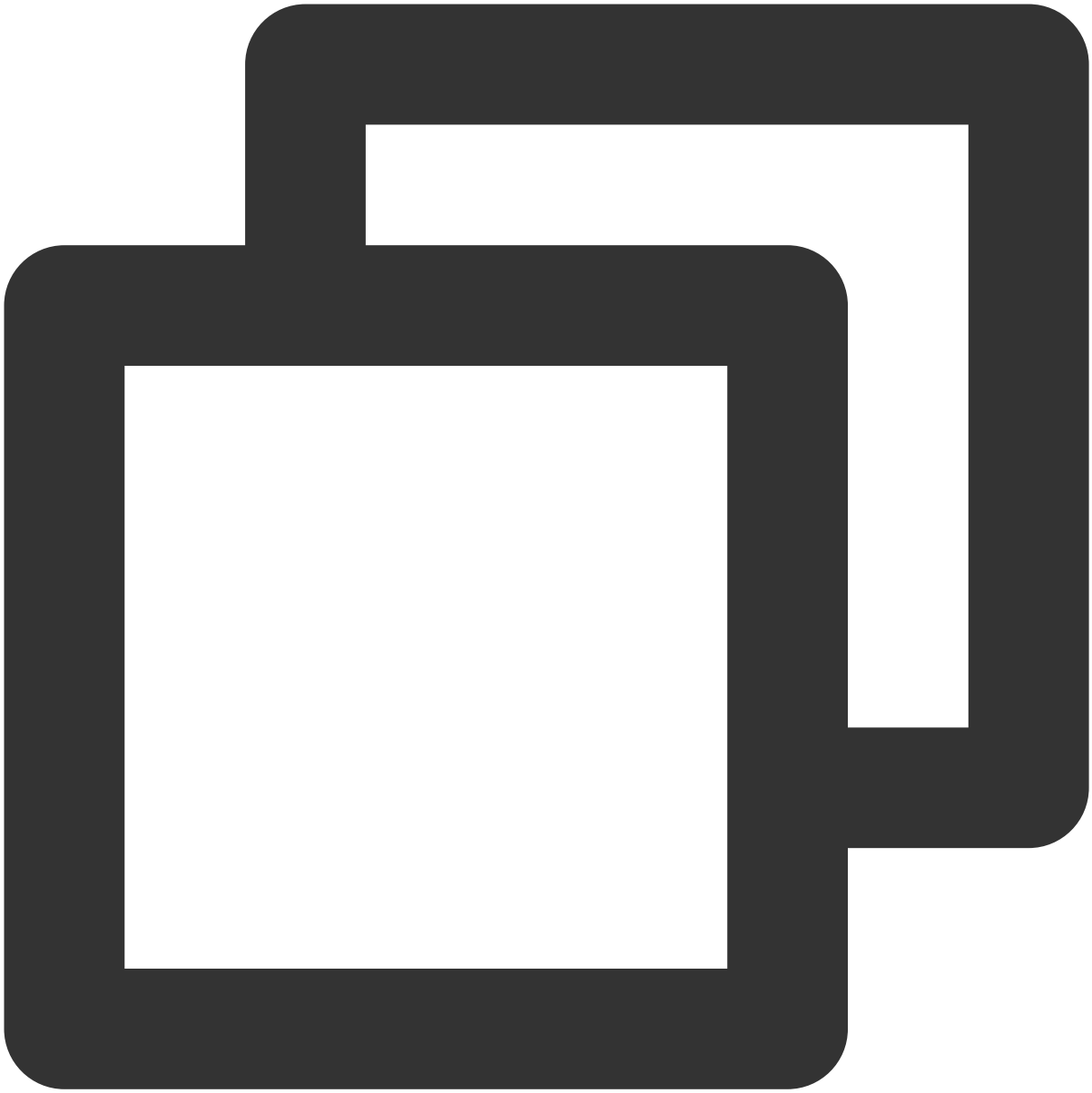
本接口用于主动查询指定的文件解压任务结果。

方法原型



```
public Guzzle\\Service\\Resource\\Model getFileUncompressResult(array $args = array
```

请求示例



```
<?php

require dirname(__FILE__, 2) . '/vendor/autoload.php';

$secretId = "SECRETID"; //替换为用户的 secretId，请登录访问管理控制台进行查看和管理，https:/
$secretKey = "SECRETKEY"; //替换为用户的 secretKey，请登录访问管理控制台进行查看和管理，http
$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，http
$cosClient = new Qcloud\\Cos\\Client(
    array(
        'region' => $region,
        'schema' => 'https', //协议头部，默认为 http
```

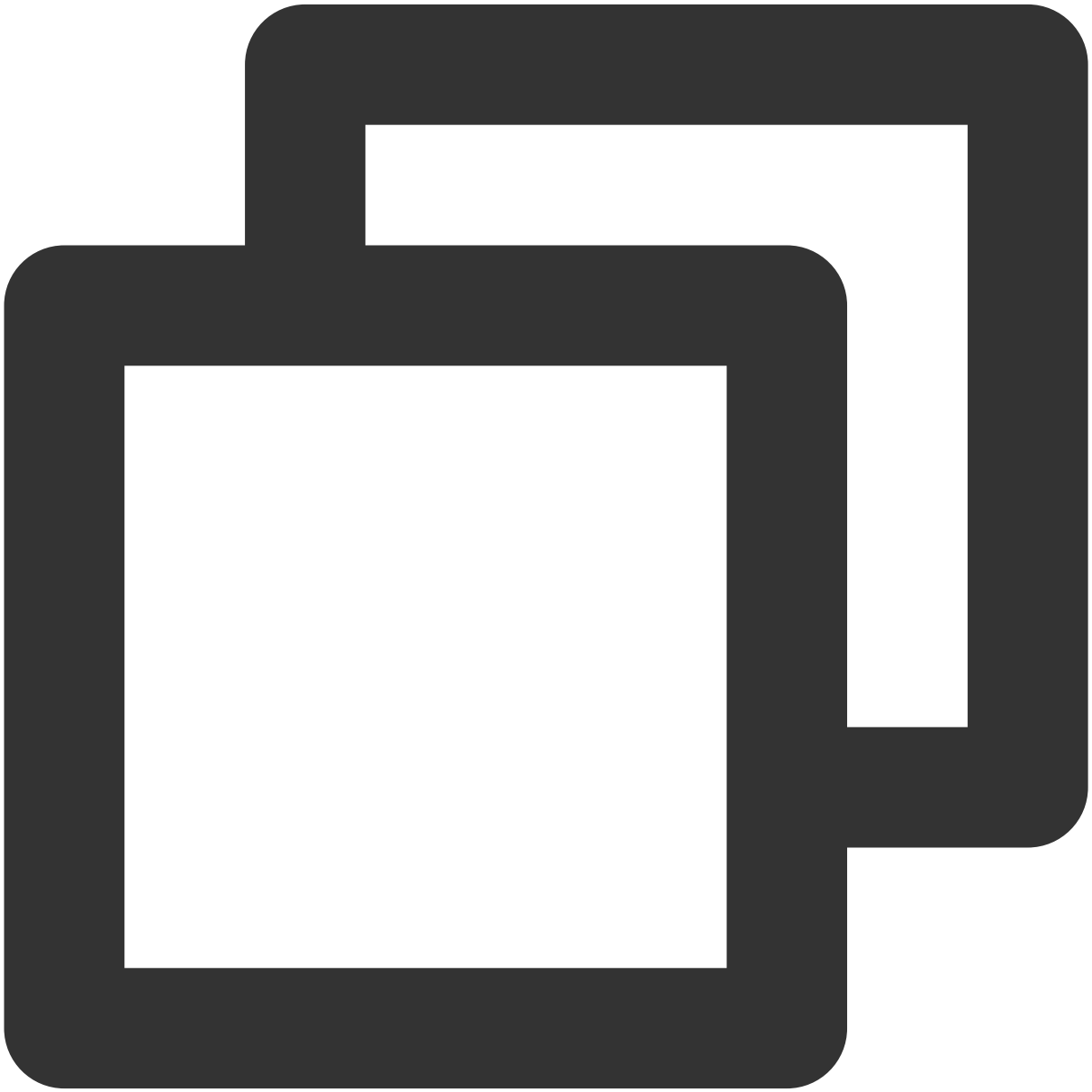
```
        'credentials'=> array(
            'secretId'  => $secretId ,
            'secretKey' => $secretKey)));
try {
    // https://www.tencentcloud.com/document/product/436/53994 查询文件解压结果
    $result = $cosClient->getFileUncompressResult(array(
        'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成
        'Key' => '', // jobId
    ));
    // 请求成功
    print_r($result);
} catch (\\Exception $e) {
    // 请求失败
    echo($e);
}
```

参数说明

Request 中的具体数据描述如下：

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	jobId	是

返回结果示例



```
GuzzleHttp\\Command\\Result Object
(
    [RequestId] => NjNkOGM2NmVfZTYxNmY5MSAASJPOJCIANSIDNAI=
    [ContentType] => application/xml
    [ContentLength] => 1092
    [JobsDetail] => Array
        (
            [Progress] => 100
            [Code] => Success
            [Message] => success
            [JobId] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
```

```
            [Tag] => FileUncompress
            [State] => Success
            [CreationTime] => 2023-01-31T16:05:31+0800
            [StartTime] => 2023-01-31T16:05:31+0800
            [EndTime] => 2023-01-31T16:05:31+0800
            [QueueId] => pcc3ae89sa9d807fs89dg789sdg
            [Input] => Array
                (
                    [BucketId] => examplebucket-1250000000
                    [Region] => ap-guangzhou
                    [Object] => test.zip
                )

            [Operation] => Array
                (
                    [JobLevel] => 0
                    [UserData] => xxx
                    [Output] => Array
                        (
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                        )

                    [FileUncompressConfig] => Array
                        (
                            [Prefix] => prefix
                            [PrefixReplaced] => 1
                        )

                    [FileUncompressResult] => Array
                        (
                            [Region] => ap-guangzhou
                            [Bucket] => examplebucket-1250000000
                            [FileCount] => 6
                        )

                )

        )

    [Key] => fcd32dbdaa13b11esa9ds8g0d98gd0h85
    [Bucket] => examplebucket-1250000000
    [Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/file_jobs/f
)
```

Python SDK
快速入门

最近更新时间：2024-03-13 11:32:21

下载与安装

相关资源

对象存储的 XML Python SDK 源码下载地址：XML Python SDK。
SDK 快速下载地址：XML Python SDK。
演示示例 Demo 下载地址：XML Python Demo。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：Python SDK 常见问题。

说明

如果您在使用 XML 版本 SDK 时遇到函数或方法不存在等错误，请先将 XML 版本 SDK 升级到最新版再重试。

环境依赖

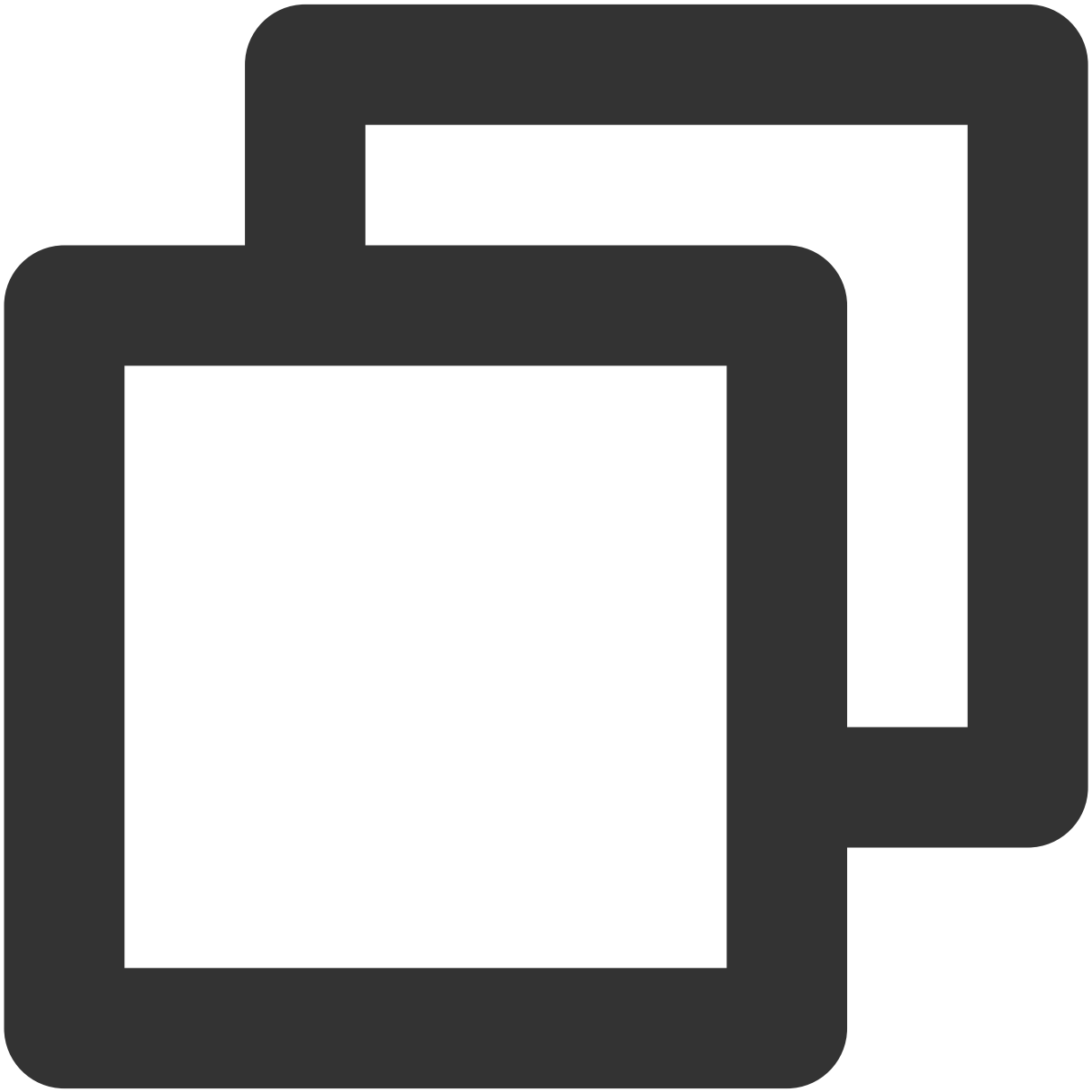
对象存储的 XML Python SDK 目前可以支持 Python 2.7 以及 Python 3.4 及以上。

说明

关于文章中出现的 SecretId、SecretKey、Bucket、Region 等名称的含义和获取方式请参见 COS 术语信息。

安装 SDK

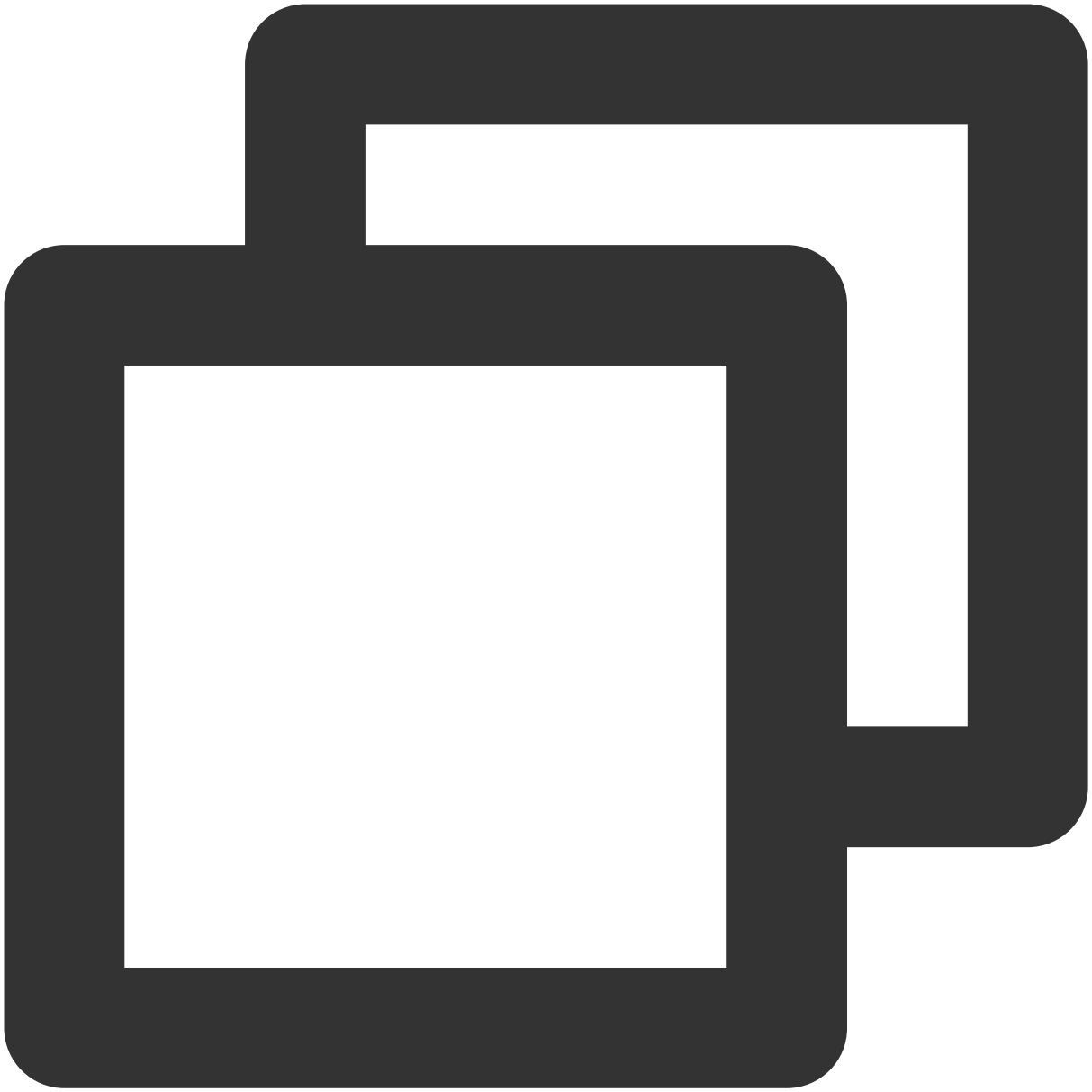
安装 SDK 有三种安装方式：pip 安装、手动安装和离线安装。
使用 pip 安装（推荐）



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

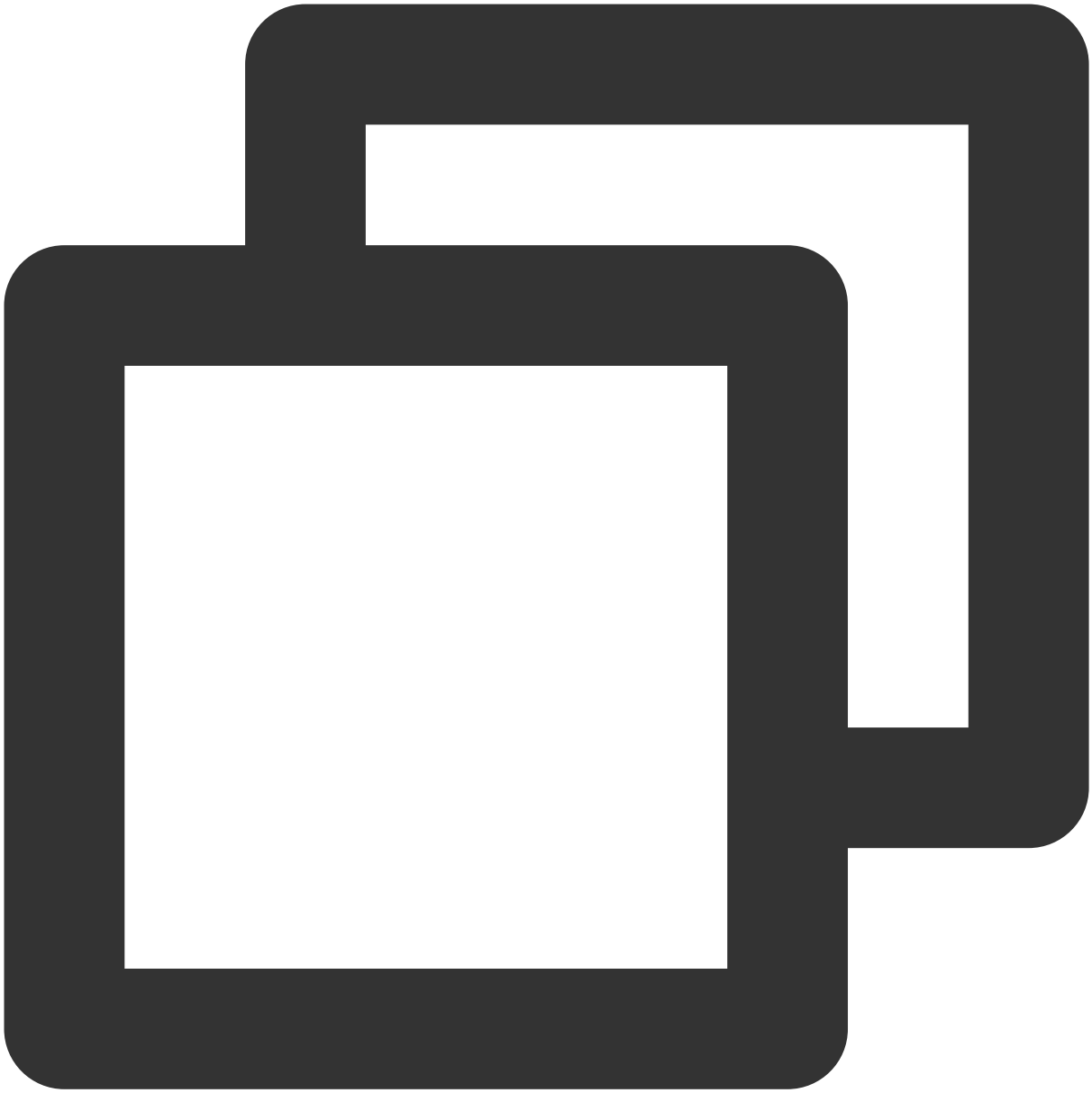
手动安装

从 XML Python SDK 下载源码，通过 setup 手动安装，执行以下命令：



```
python setup.py install
```

离线安装



```
# 在有外网的机器下运行如下命令
mkdir cos-python-sdk-packages
pip download cos-python-sdk-v5 -d cos-python-sdk-packages
tar -czvf cos-python-sdk-packages.tar.gz cos-python-sdk-packages
# 将安装包拷贝到没有外网的机器后运行如下命令
# 请确保两台机器的 python 版本保持一致，否则会出现安装失败的情况
tar -xzvf cos-python-sdk-packages.tar.gz
pip install cos-python-sdk-v5 --no-index -f cos-python-sdk-packages
```

版权所有：腾讯云计算（北京）有限责任公司

开始使用

下面为您介绍如何使用 COS Python SDK 完成一个基础操作，例如初始化客户端、创建存储桶、查询存储桶列表、上传对象、查询对象列表、下载对象和删除对象。

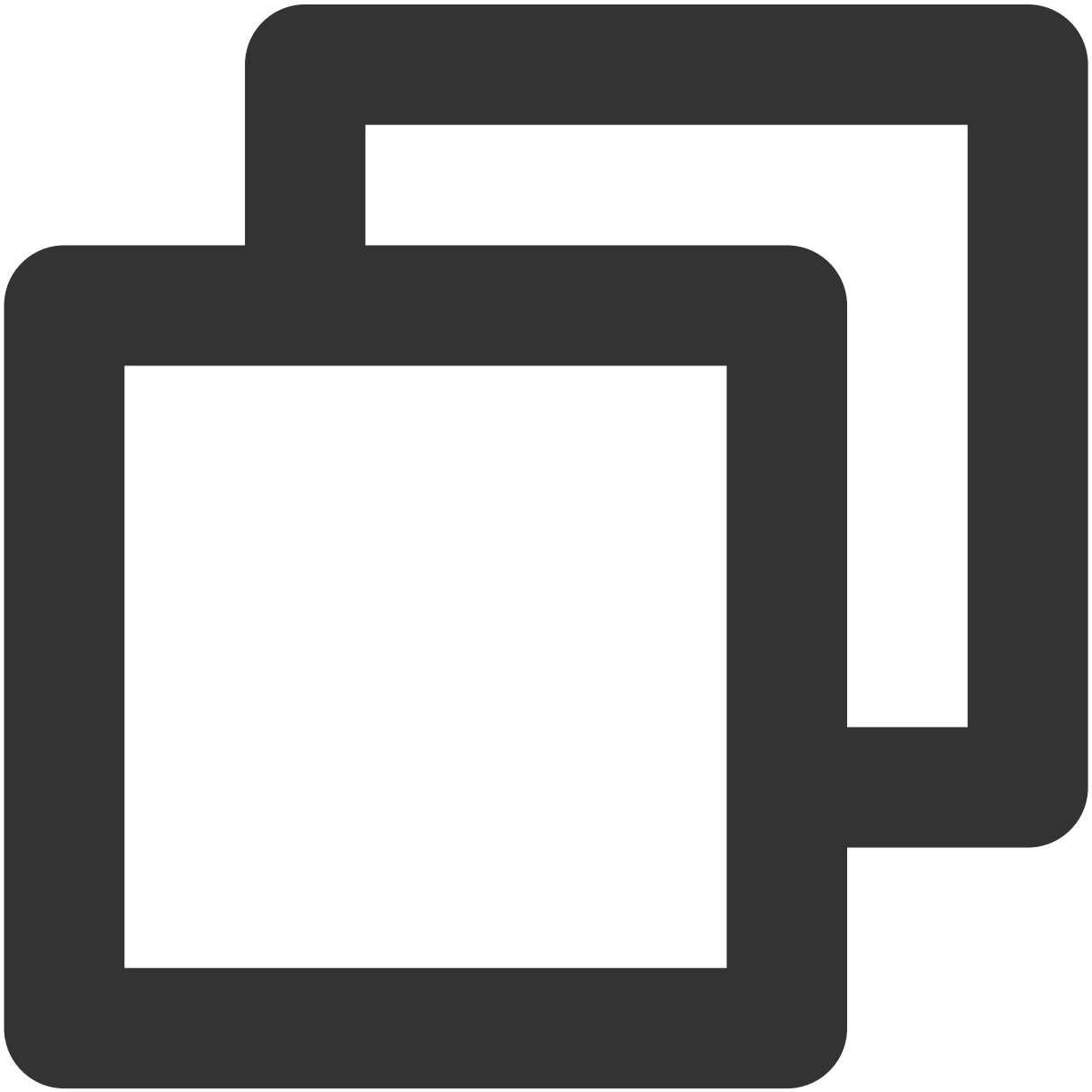
初始化

注意

建议用户 使用临时密钥 调用 SDK，通过临时授权的方式进一步提高 SDK 使用的安全性。申请临时密钥时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。
如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。
下面介绍几种初始化 Python SDK Client 的方式，您可以根据具体场景选择其中一种。

通过 COS 默认域名初始化（默认方式）

通过 COS 默认域名访问时，SDK 会以 **{bucket-appid}.cos.{region}.myqcloud.com** 的域名形式访问 COS。



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS支持的所有 region 列表参见 https://cloud.tencent.com/d
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

注意

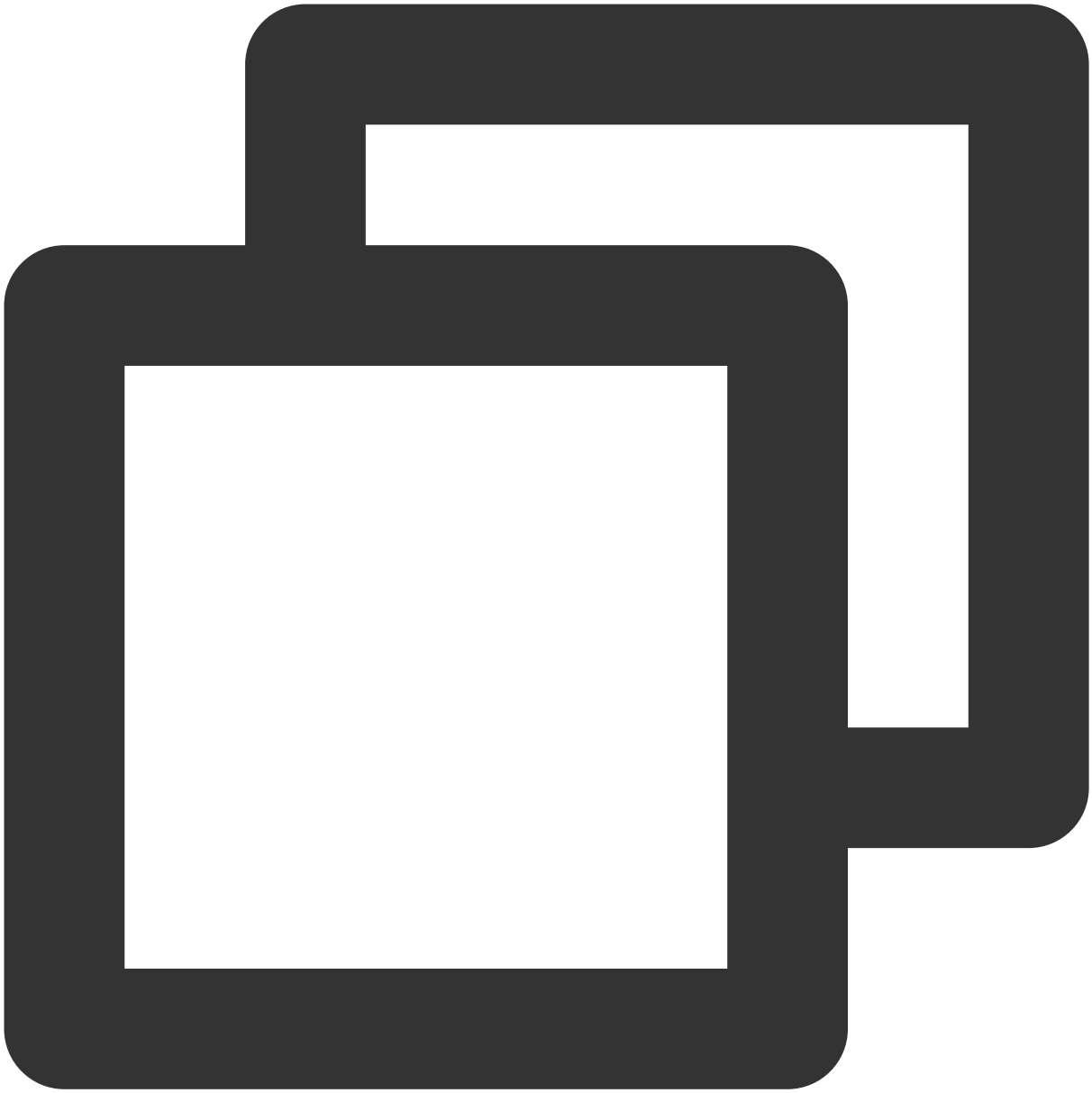
正常情况下一个 region 只需要生成一个 CosS3Client 实例，然后循环上传或下载对象，不能每次访问都生成 CosS3Client 实例，否则 python 进程会占用过多的连接和线程。

说明

关于临时密钥如何生成和使用，请参见 临时密钥生成及使用指引。

通过 COS 默认域名和代理初始化

当需要通过代理访问 COS 时配置，否则跳过。

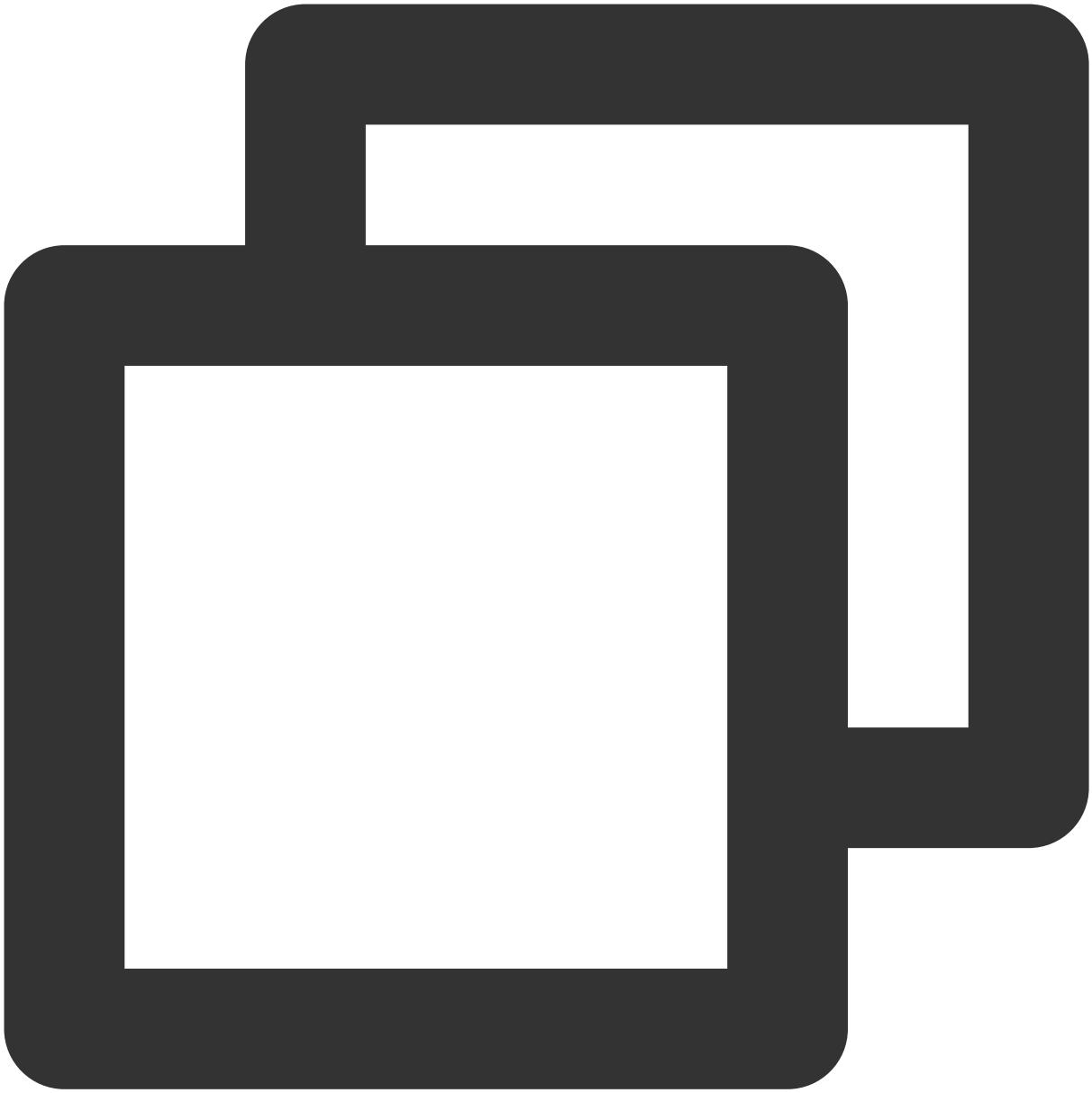


```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS支持的所有 region 列表参见 https://cloud.tencent.com/d
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
proxies = {
    'http': '127.0.0.1:80',  # 替换为用户的 HTTP 代理地址
    'https': '127.0.0.1:443' # 替换为用户的 HTTPS 代理地址
}

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

通过 COS 加速域名初始化

通过 COS 全球加速域名访问时，SDK 会以 **{bucket-appid}.cos.accelerate.myqcloud.com** 的域名形式访问 COS，region 不会出现在访问域名中。

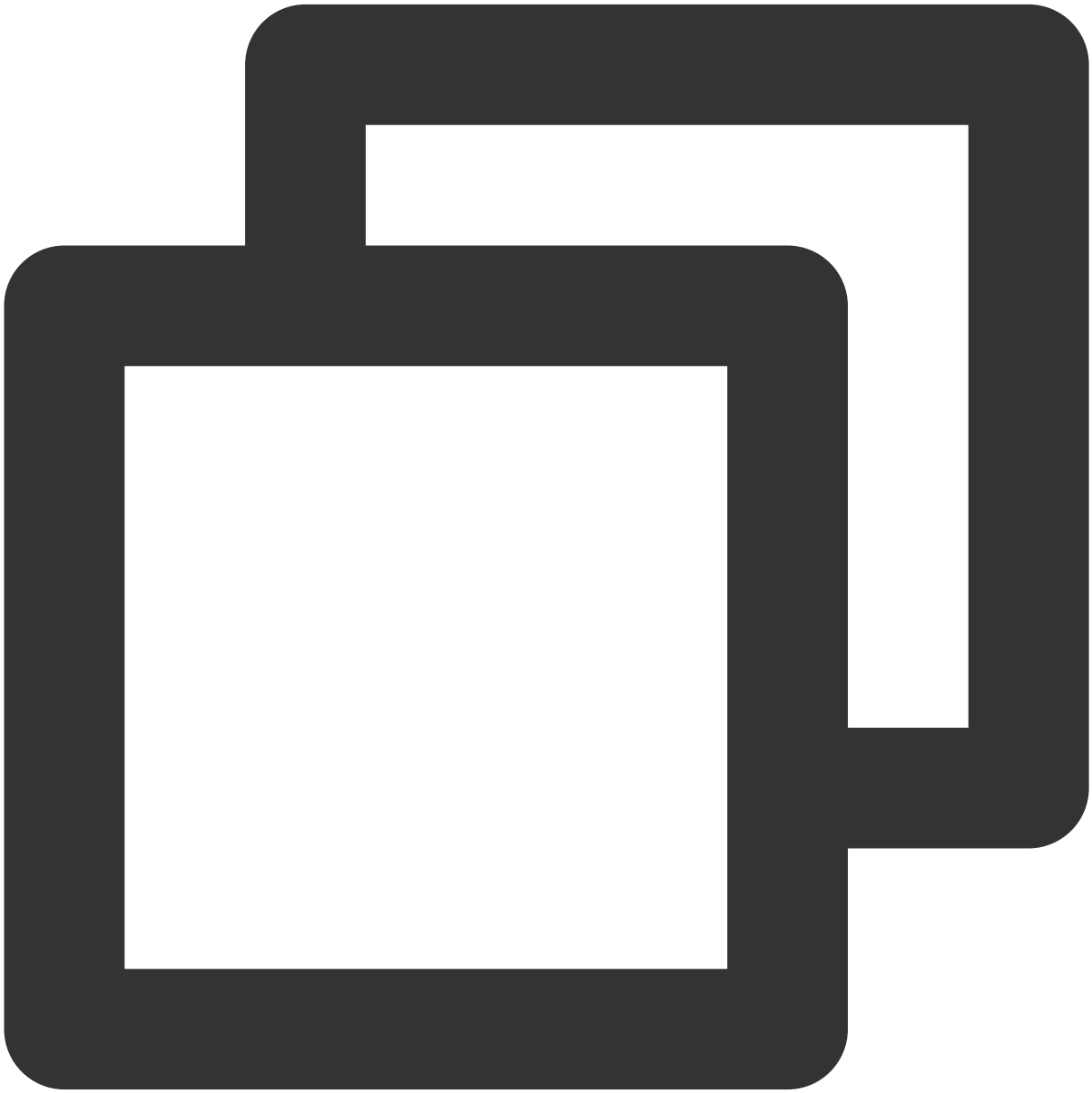


```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
region = None               # 通过 Endpoint 初始化不需要配置 region
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

endpoint = 'cos.accelerate.myqcloud.com' # 替换为用户的 endpoint 或者 cos 全局加速域名，
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

通过自定义域名初始化

通过自定义域名访问时，SDK 会使用配置的**用户域名**直接访问 COS，bucket 和 region 都不会出现在访问域名中。

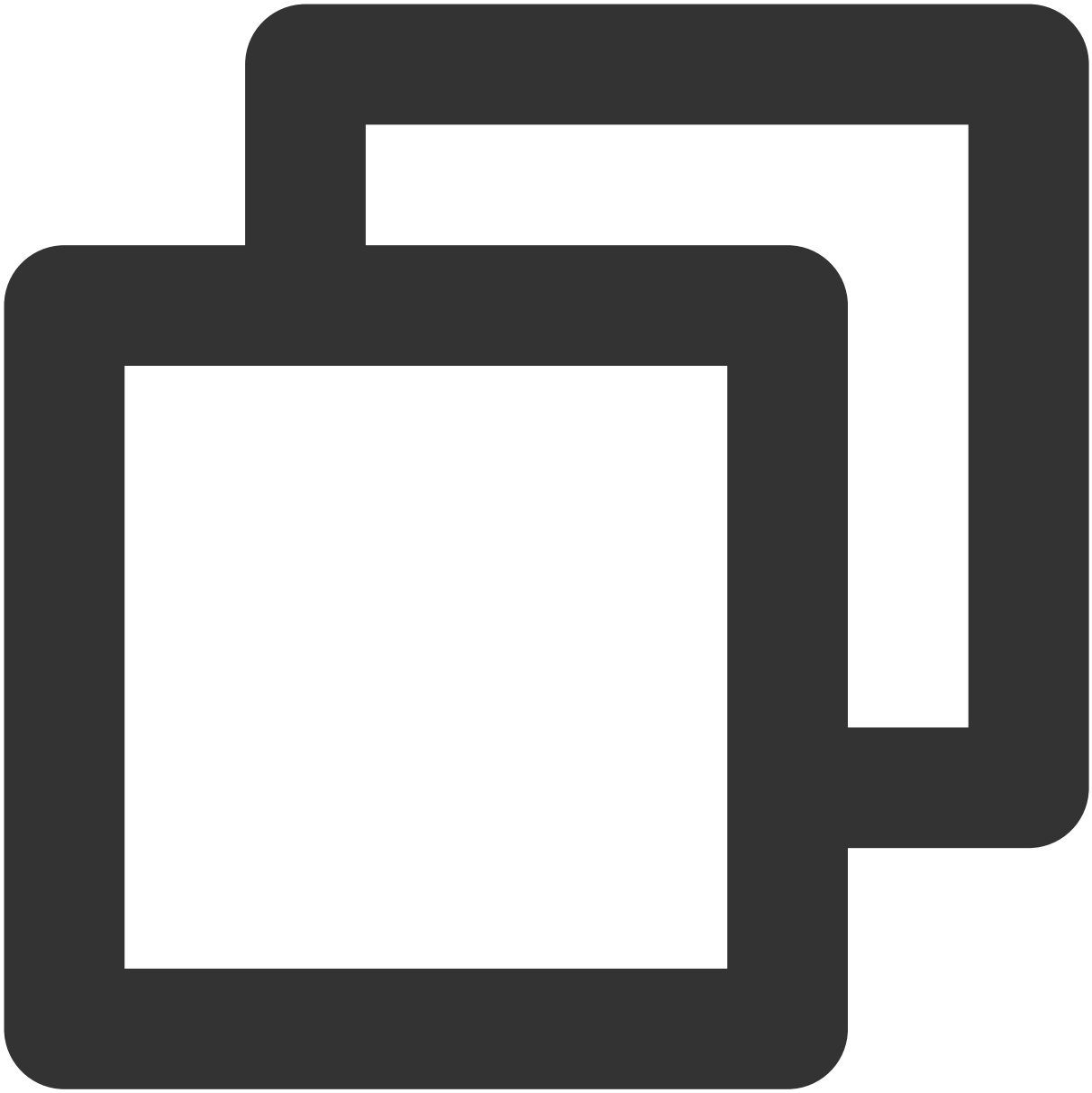


```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
region = None               # 通过自定义域名初始化不需要配置 region
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

domain = 'user-define.example.com' # 用户自定义域名，需要先开启桶的自定义域名，具体请参见 ht
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

通过 CDN 默认域名初始化

通过 CDN 默认域名访问时，SDK 会以 **{bucket-appid}.file.mycloud.com** 的域名形式访问 **CDN**，由 CDN 服务回源访问 COS。

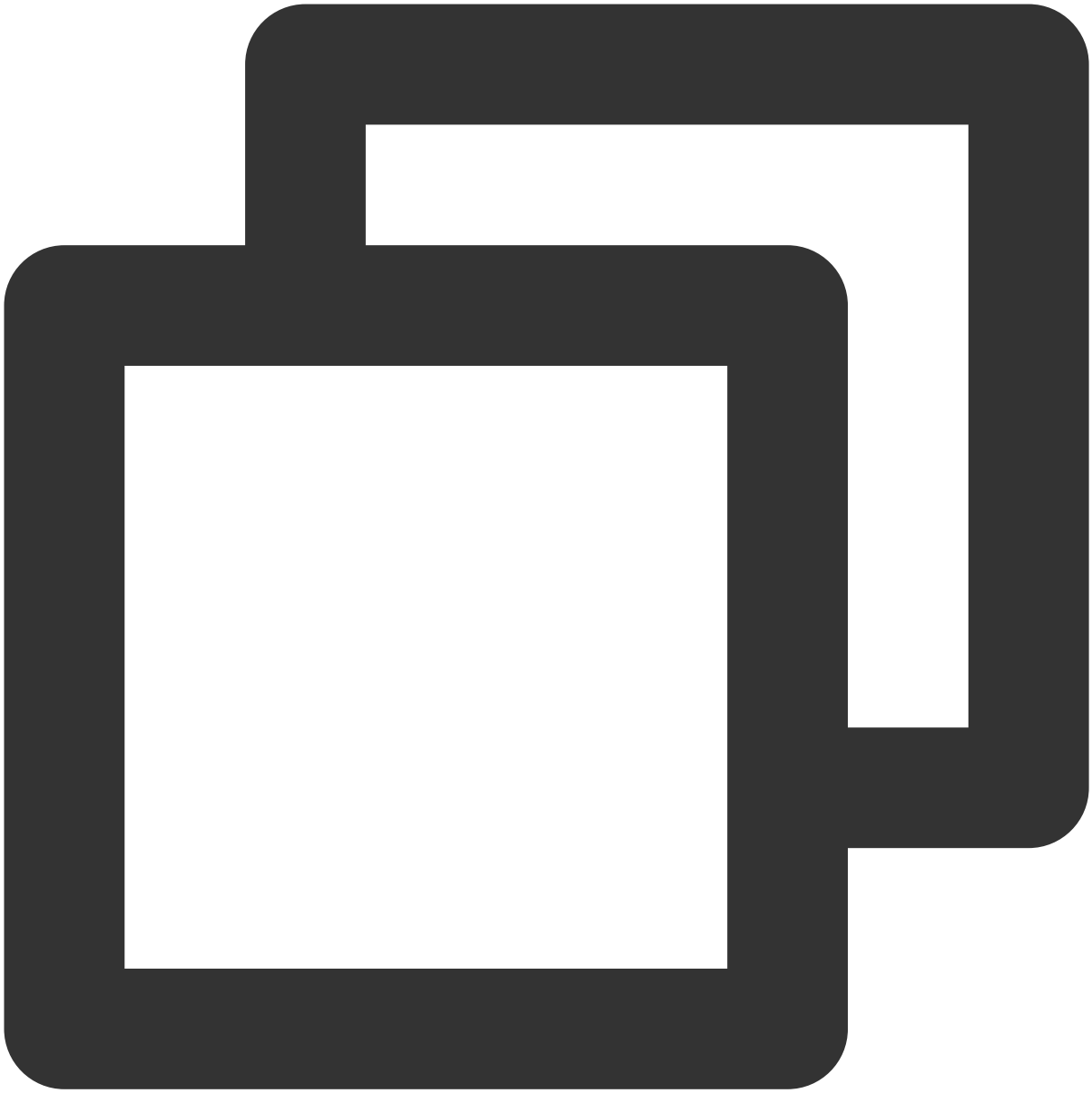


```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
region = None               # 通过Endpoint初始化不需要配置 region
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

endpoint = 'file.mycloud.com' # 替换为用户的 CDN 默认加速域名，需要开通 CDN 加速配置，参见
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

通过 CDN 自定义域名初始化

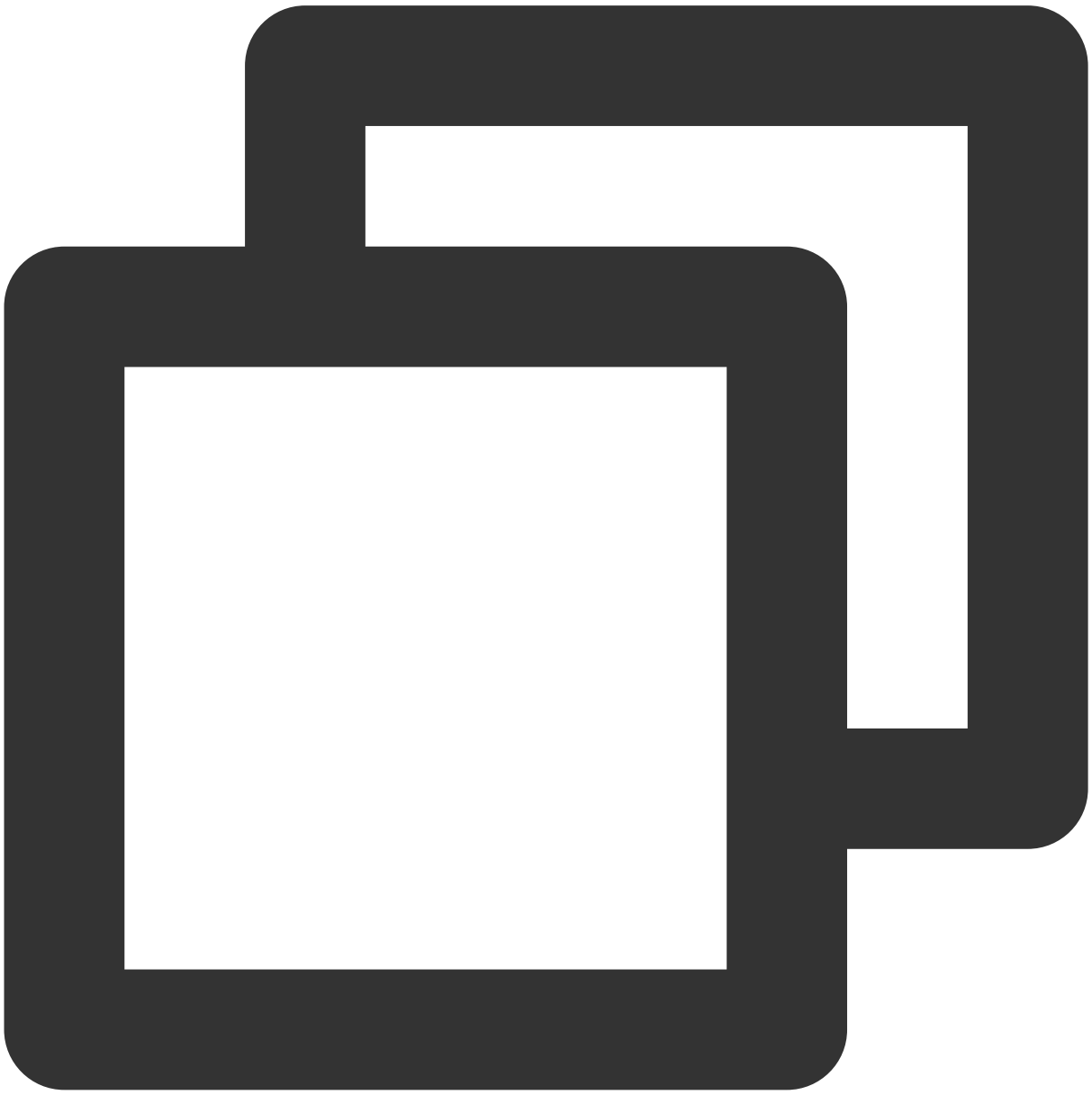
通过 CDN 自定义域名访问时，SDK 会使用配置的**用户域名**直接访问 **CDN**，bucket 和 region 都不会出现在访问域名中，由 CDN 服务回源访问 COS。



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在CosConfig中移除，请在参
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

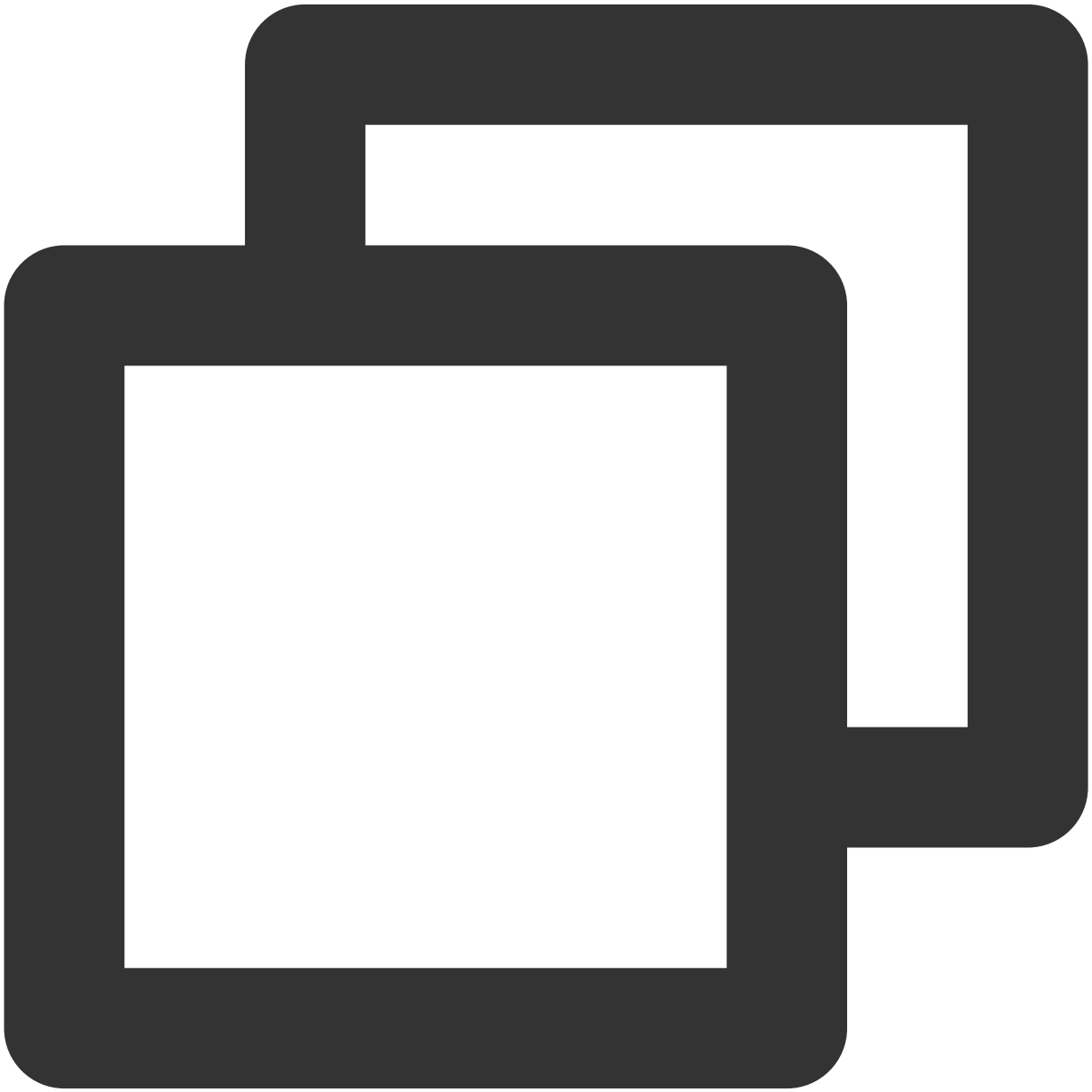
```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
region = None               # 通过自定义域名初始化不需要配置 region
token = None                # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

domain = 'user-define.example-cdn.com' # 用户自定义 CDN 域名，需要开通 CDN 自定义域名加速,
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)
```

创建存储桶



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id, secret_key, region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，ht
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，h
                            # COS 支持的所有 region 列表参见 https://cloud.tencent.com/
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

response = client.create_bucket(
    Bucket='examplebucket-1250000000'
)
```

查询存储桶列表



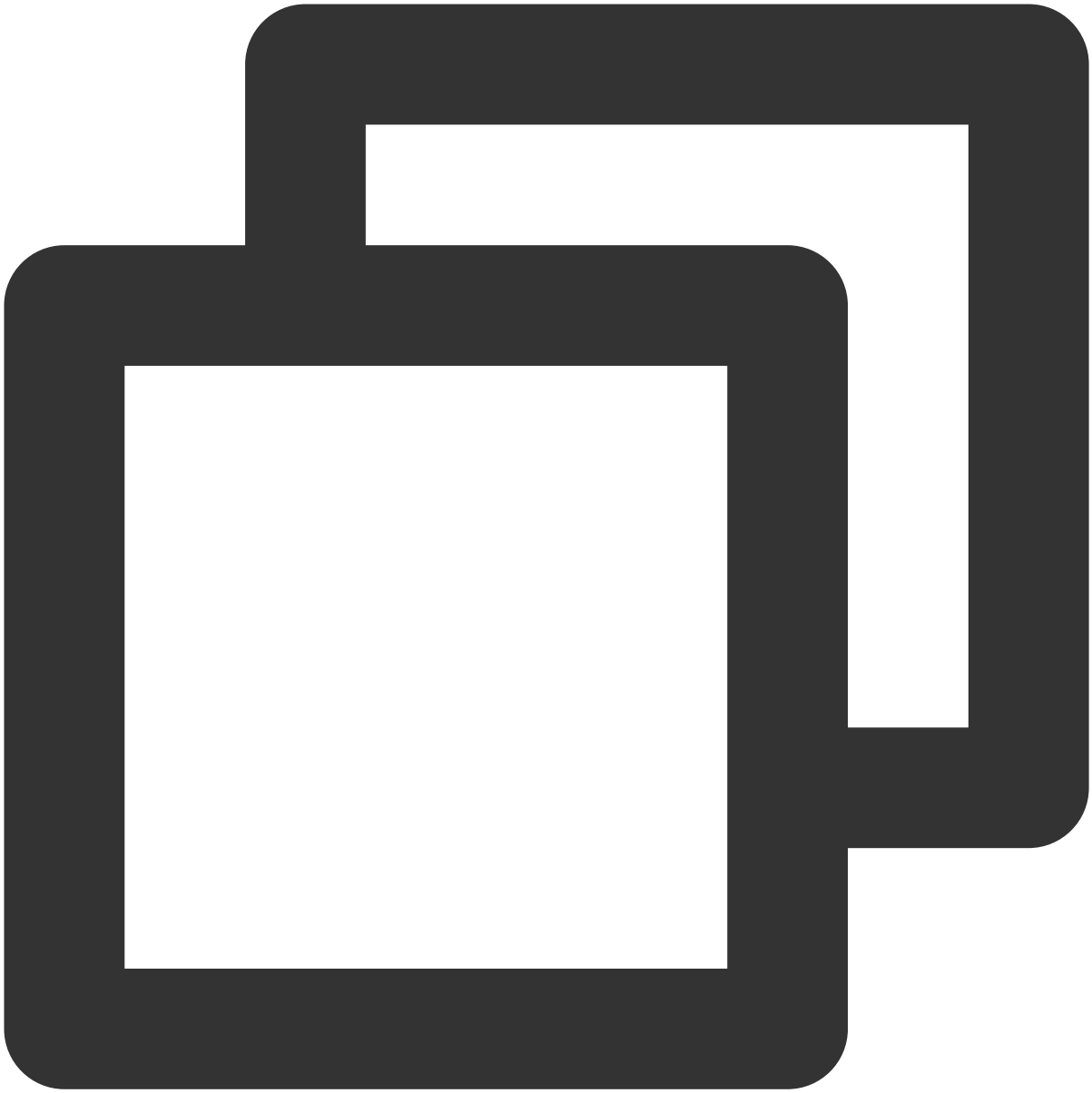
```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS 支持的所有 region 列表参见 https://cloud.tencent.com/
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

response = client.list_buckets(
)
```

上传对象

注意

简单上传不支持超过5G的文件，推荐使用下方高级上传接口。参数说明可参见 对象操作 文档。

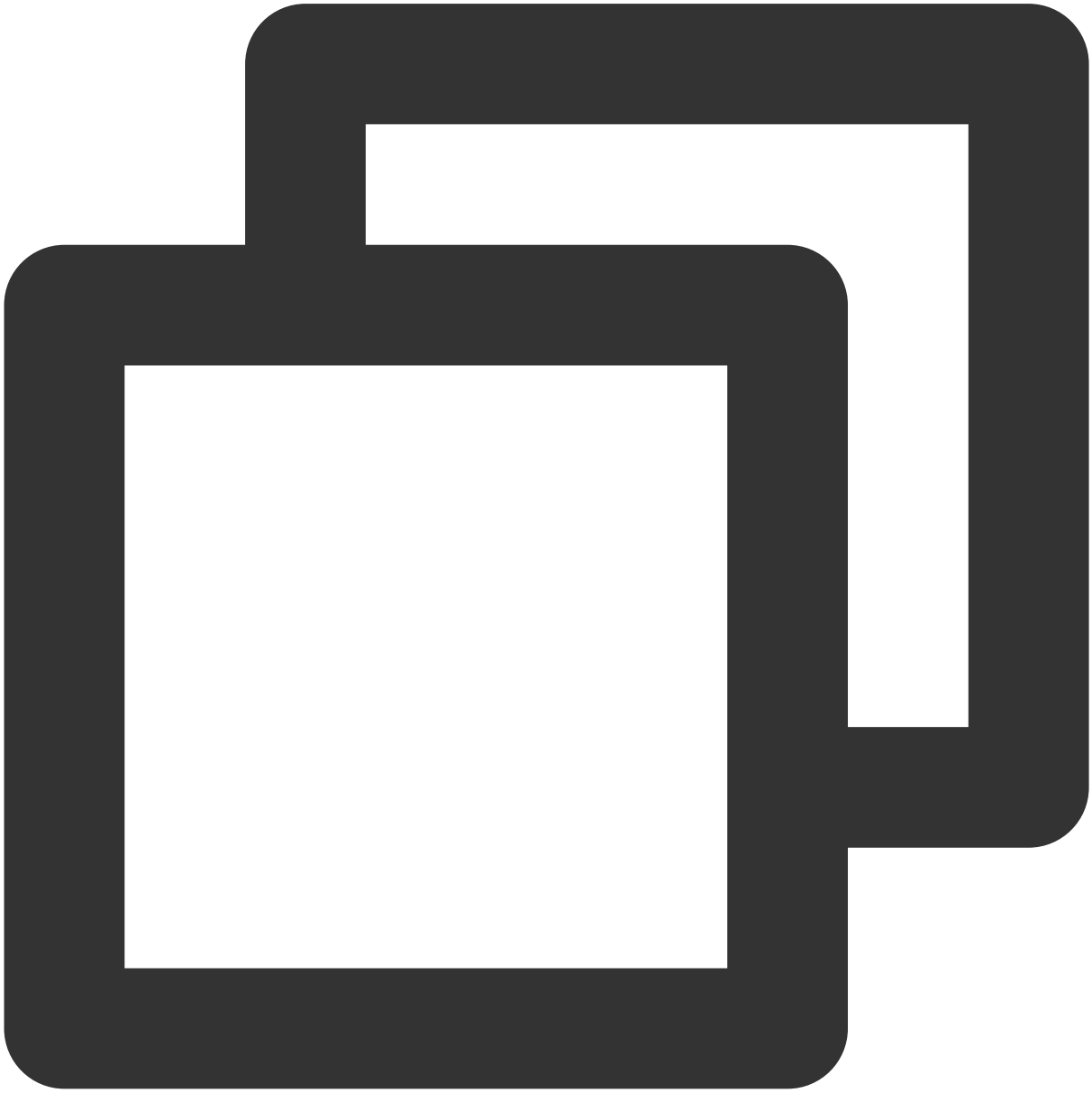


```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS 支持的所有 region 列表参见 https://cloud.tencent.com/
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

#### 文件流简单上传（不支持超过5G的文件，推荐使用下方高级上传接口）
# 强烈建议您以二进制模式(binary mode)打开文件,否则可能会导致错误
with open('picture.jpg', 'rb') as fp:
    response = client.put_object(
        Bucket='examplebucket-1250000000',
        Body=fp,
        Key='picture.jpg',
        StorageClass='STANDARD',
        EnableMD5=False
    )
print(response['ETag'])

#### 字节流简单上传
response = client.put_object(
    Bucket='examplebucket-1250000000',
    Body=b'bytes',
    Key='picture.jpg',
    EnableMD5=False
)
print(response['ETag'])


#### chunk 简单上传
import requests
stream = requests.get('https://cloud.tencent.com/document/product/436/7778')

# 网络流将以 Transfer-Encoding:chunked 的方式传输到 COS
response = client.put_object(
    Bucket='examplebucket-1250000000',
    Body=stream,
    Key='picture.jpg'
)
print(response['ETag'])

#### 高级上传接口（推荐）
# 根据文件大小自动选择简单上传或分块上传，分块上传具备断点续传功能。
response = client.upload_file(
    Bucket='examplebucket-1250000000',
```

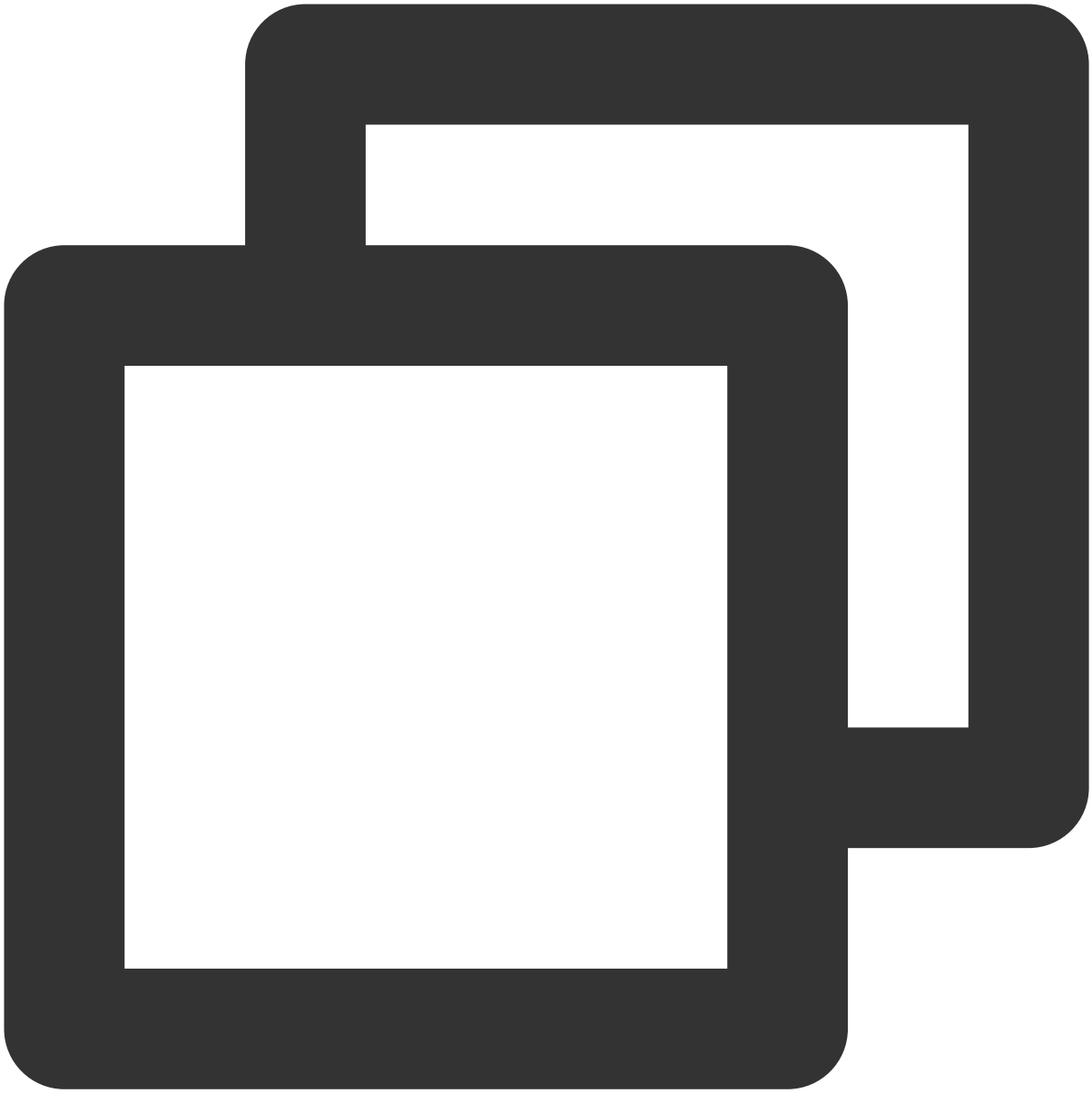
```
    LocalFilePath='local.txt',
    Key='picture.jpg',
    PartSize=1,
    MAXThread=10,
    EnableMD5=False
)
print(response['ETag'])
```

查询对象列表



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，ht
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，htt
                            # COS 支持的所有 region 列表参见https://cloud.tencent.com/d
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

response = client.list_objects(
    Bucket='examplebucket-1250000000',
    Prefix='folder1'
)
```

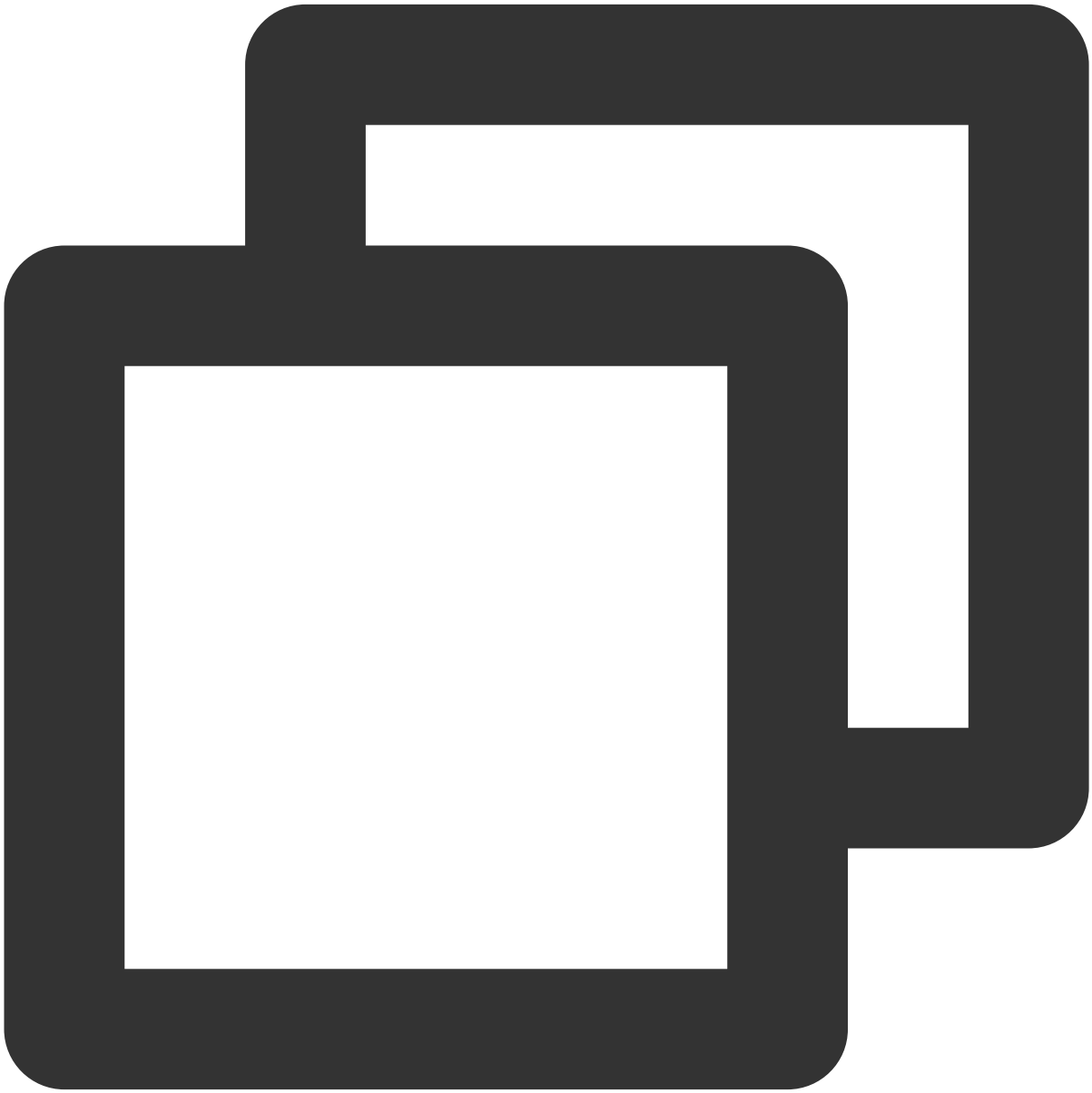
单次调用 `list_objects` 接口一次只能查询1000个对象，如需要查询所有的对象，则需要循环调用。



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region等。Appid 已在 CosConfig 中移除，请在
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS 支持的所有 region 列表参见https://cloud.tencent.com/d
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

marker = ""
while True:
    response = client.list_objects(
        Bucket='examplebucket-1250000000',
        Prefix='folder1',
        Marker=marker
    )
    print(response['Contents'])
    if response['IsTruncated'] == 'false':
        break
    marker = response['NextMarker']
```

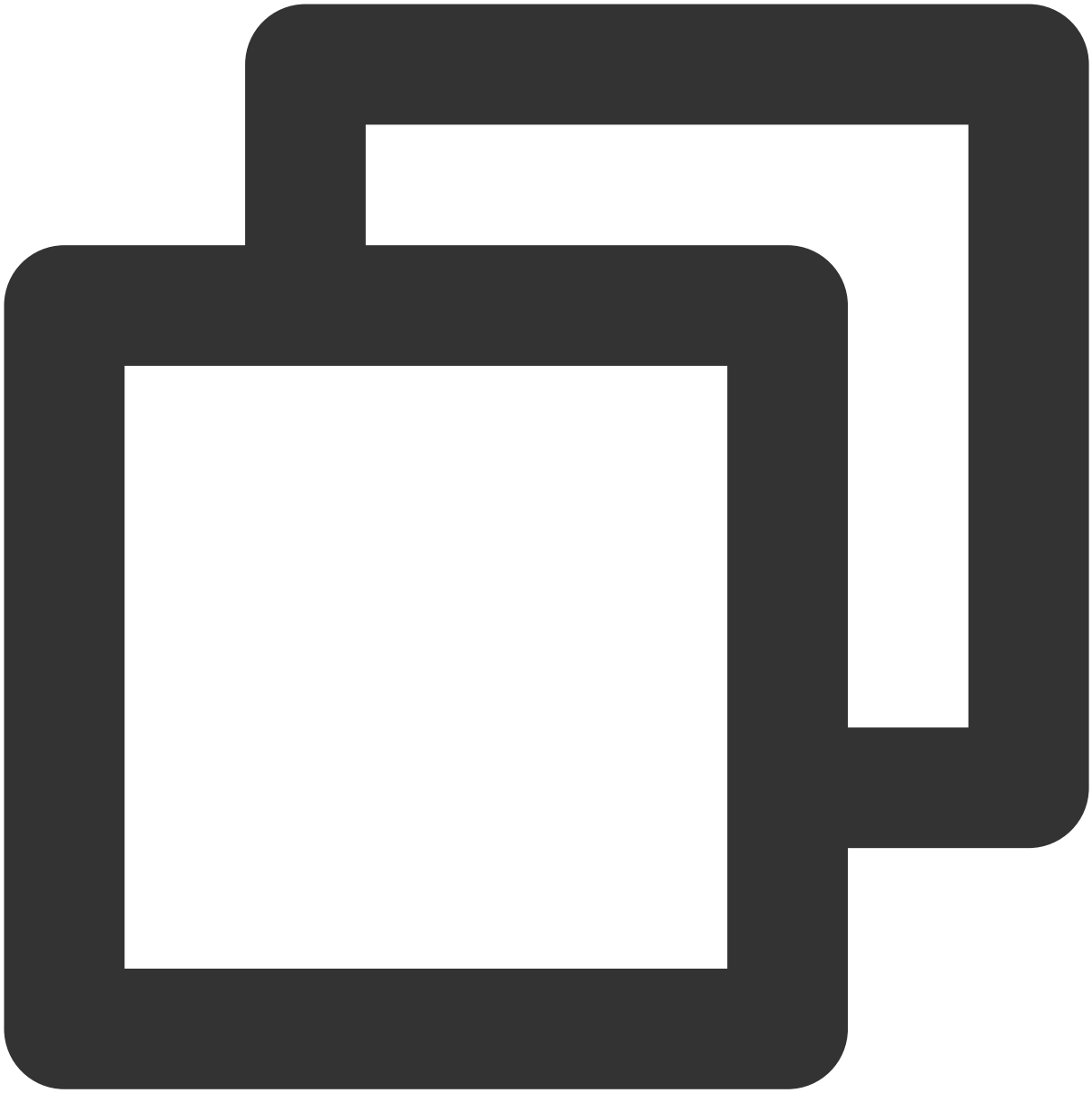
下载对象



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS支持的所有 region 列表参见 https://cloud.tencent.com/d
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

####  获取文件到本地
response = client.get_object(
    Bucket='examplebucket-1250000000',
    Key='picture.jpg',
)
response['Body'].get_stream_to_file('output.txt')

#### 获取文件流
response = client.get_object(
    Bucket='examplebucket-1250000000',
    Key='picture.jpg',
)
fp = response['Body'].get_raw_stream()
print(fp.read(2))

#### 设置 Response HTTP 头部
response = client.get_object(
    Bucket='examplebucket-1250000000',
    Key='picture.jpg',
    ResponseContentType='text/html; charset=utf-8'
)
print(response['Content-Type'])
fp = response['Body'].get_raw_stream()
print(fp.read(2))

#### 指定下载范围
response = client.get_object(
    Bucket='examplebucket-1250000000',
    Key='picture.jpg',
    Range='bytes=0-10'
)
fp = response['Body'].get_raw_stream()
print(fp.read())
```

删除对象



```
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

# 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)

# 1. 设置用户属性，包括 secret_id，secret_key，region 等。Appid 已在 CosConfig 中移除，请
secret_id = 'SecretId'      # 替换为用户的 SecretId，请登录访问管理控制台进行查看和管理，http
```

```
secret_key = 'SecretKey'    # 替换为用户的 SecretKey，请登录访问管理控制台进行查看和管理，htt
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，h
                            # COS 支持的所有 region 列表参见 https://cloud.tencent.com/
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
scheme = 'https'            # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=t
client = CosS3Client(config)

# 删除 object
## deleteObject
response = client.delete_object(
    Bucket='examplebucket-1250000000',
    Key='exampleobject'
)

# 删除多个 object
## deleteObjects
response = client.delete_objects(
    Bucket='examplebucket-1250000000',
    Delete={
        'Object': [
            {
                'Key': 'exampleobject1',
            },
            {
                'Key': 'exampleobject2',
            },
        ],
        'Quiet': 'true'|'false'
    }
)
```

图片处理
图片持久化处理

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于图片持久化处理相关的 API 概览以及 SDK 示例代码。

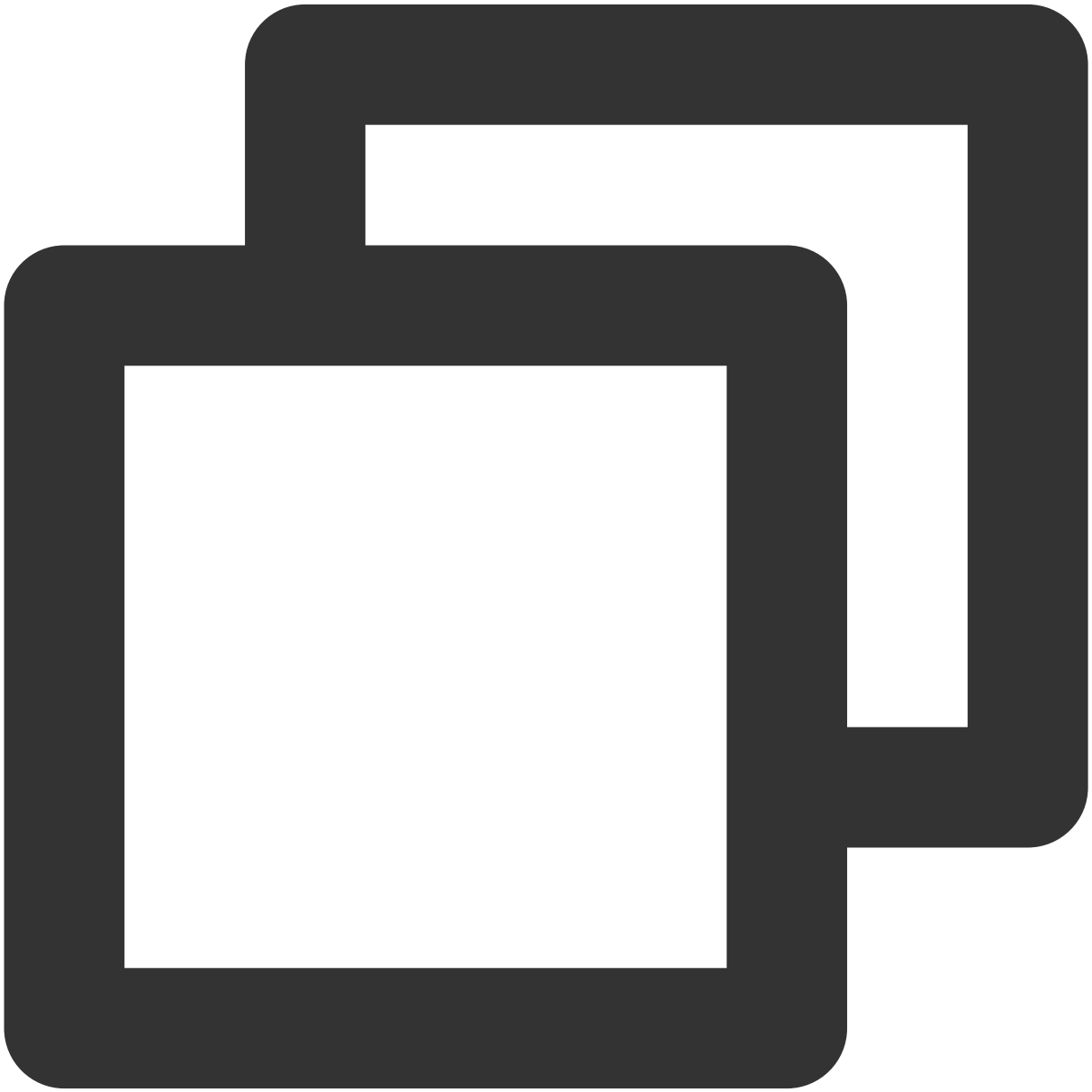
API	操作描述
图片持久化处理	对象存储（Cloud Object Storage，COS）提供的上传时处理功能可以帮助使用者在上传时实现图片处理。此外能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。支持处理32MB以内的图片。

上传时处理

功能说明

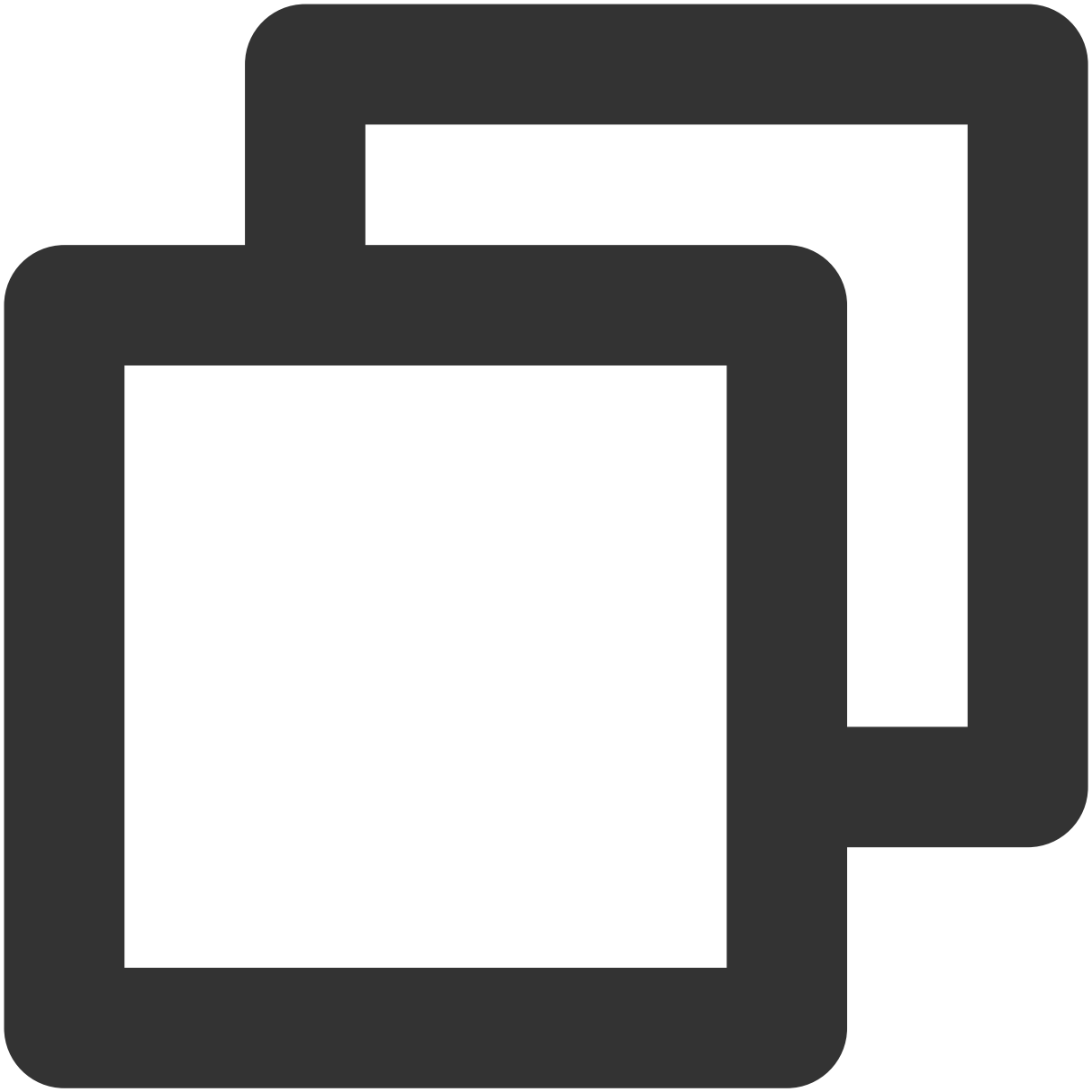
数据万象提供的上传时处理功能可以帮助使用者在上传时实现图片处理。您只需要在请求包头部中加入 Pic-Operations 项并设置好相应参数，就可在图片上传时实现相应的图片处理，并可将原图和处理结果存入到 COS。

方法原型



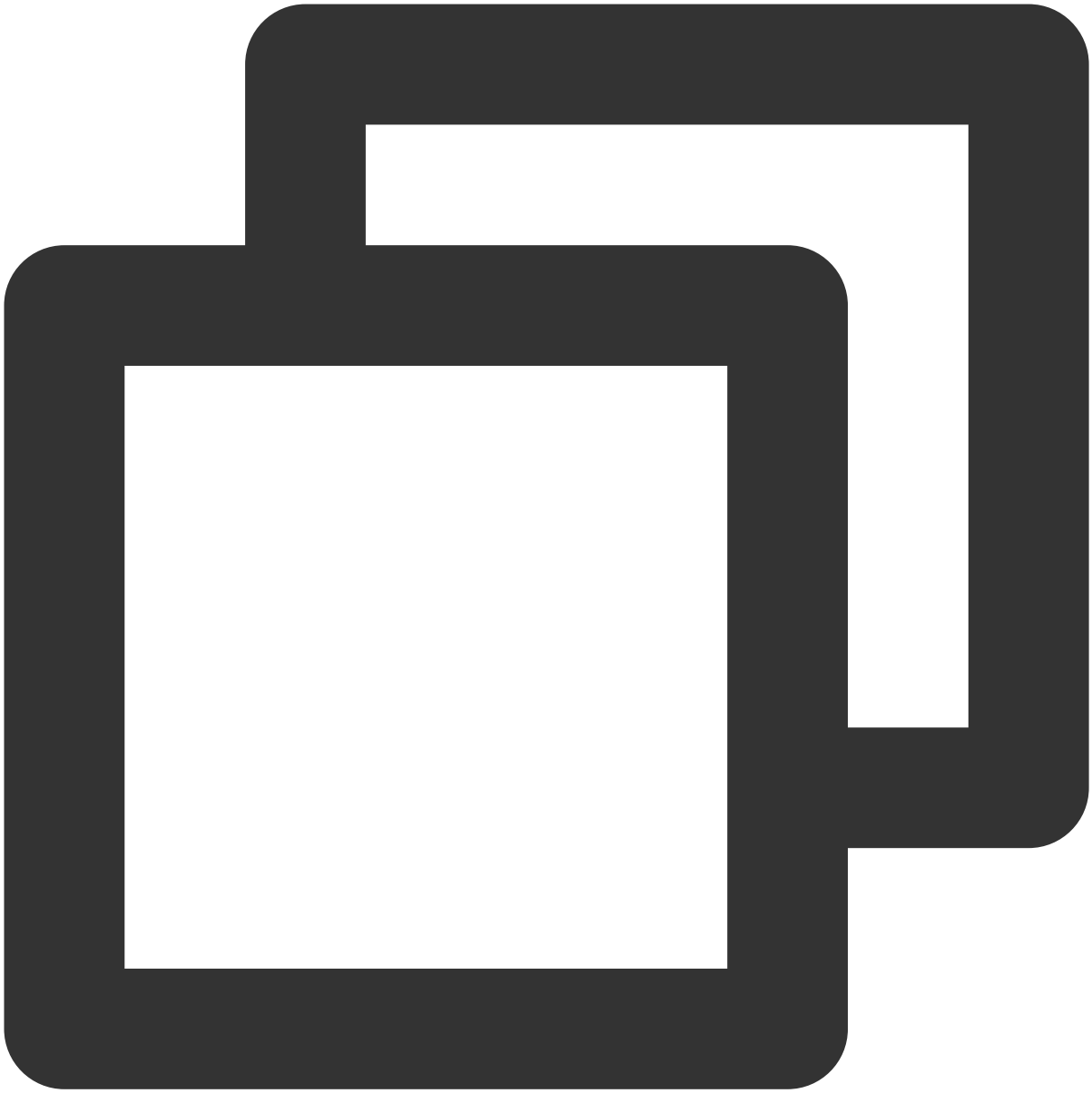
```
ci_put_object(self, Bucket, Body, Key, EnableMD5=False, **kwargs)
```

请求示例



```
with open('local.jpg', 'rb') as fp:
    response, data = client.ci_put_object(
        Bucket='examplebucket-1250000000',
        Body=fp,
        Key=ci_file_name,
        # pic operation json struct
        PicOperations='{"is_pic_info":1,"rules":[{"fileid":"format.png","rule":"ima
```

全部参数请求示例



```
response = client.ci_put_object(
    Bucket='examplebucket-1250000000',
    Body=b'bytes'|file,
    Key='exampleobject',
    EnableMD5=False|True,
    ACL='private'|'public-read',  # 请慎用此参数,否则会达到1000条 ACL 上限
    GrantFullControl='string',
    GrantRead='string',
    StorageClass='STANDARD'|'STANDARD_IA'|'ARCHIVE',
    Expires='string',
    CacheControl='string',
```

```
    ContentType='string',
    ContentDisposition='string',
    ContentEncoding='string',
    ContentLanguage='string',
    ContentLength='123',
    ContentMD5='string',
    Metadata={
        'x-cos-meta-key1': 'value1',
        'x-cos-meta-key2': 'value2'
    },
    TrafficLimit='1048576'
    PicOperations='{"is_pic_info":1,"rules":[{"fileid":"format.png","rule":"imageVi
)
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶名称，由 BucketName-APPID 构成	String	是
Body	上传对象的内容，可以为文件流或字节流	file/bytes	是
Key	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	String	是
PicOperations	万象图片处理参数，请参见 图片持久化处理	String	是
EnableMD5	是否需要 SDK 计算 Content-MD5，默认关闭，打开后将增加上传耗时	Bool	否
ACL	设置对象的 ACL，例如 'private'，'public-read'	String	否
GrantFullControl	赋予被授权者所有的权限，格式为 `id="OwnerUin"`	String	否
GrantRead	赋予被授权者读的权限，格式为 `id="OwnerUin"`	String	否
StorageClass	设置对象的存储类型，STANDARD，STANDARD_IA，ARCHIVE。默认值为 STANDARD。更多存储类型，请参见 存储类型概述	String	否
Expires	设置 Expires	String	否
CacheControl	缓存策略，设置 Cache-Control	String	否

版权所有：腾讯云计算（北京）有限责任公司

ContentType	内容类型，设置 Content-Type	String	否
ContentDisposition	对象名称，设置 Content-Disposition	String	否
ContentEncoding	编码格式，设置 Content-Encoding	String	否
ContentLanguage	语言类型，设置 Content-Language	String	否
ContentLength	设置传输长度	String	否
ContentMD5	设置上传对象的 MD5 值用于校验	String	否
Metadata	用户自定义的对象元数据， 必须以 x-cos-meta 开头，否则会被忽略	Dict	否
TrafficLimit	单链接限速的值，单位为 bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s	String	否

PicOperations 为 json 格式的字符串，具体参数如下：

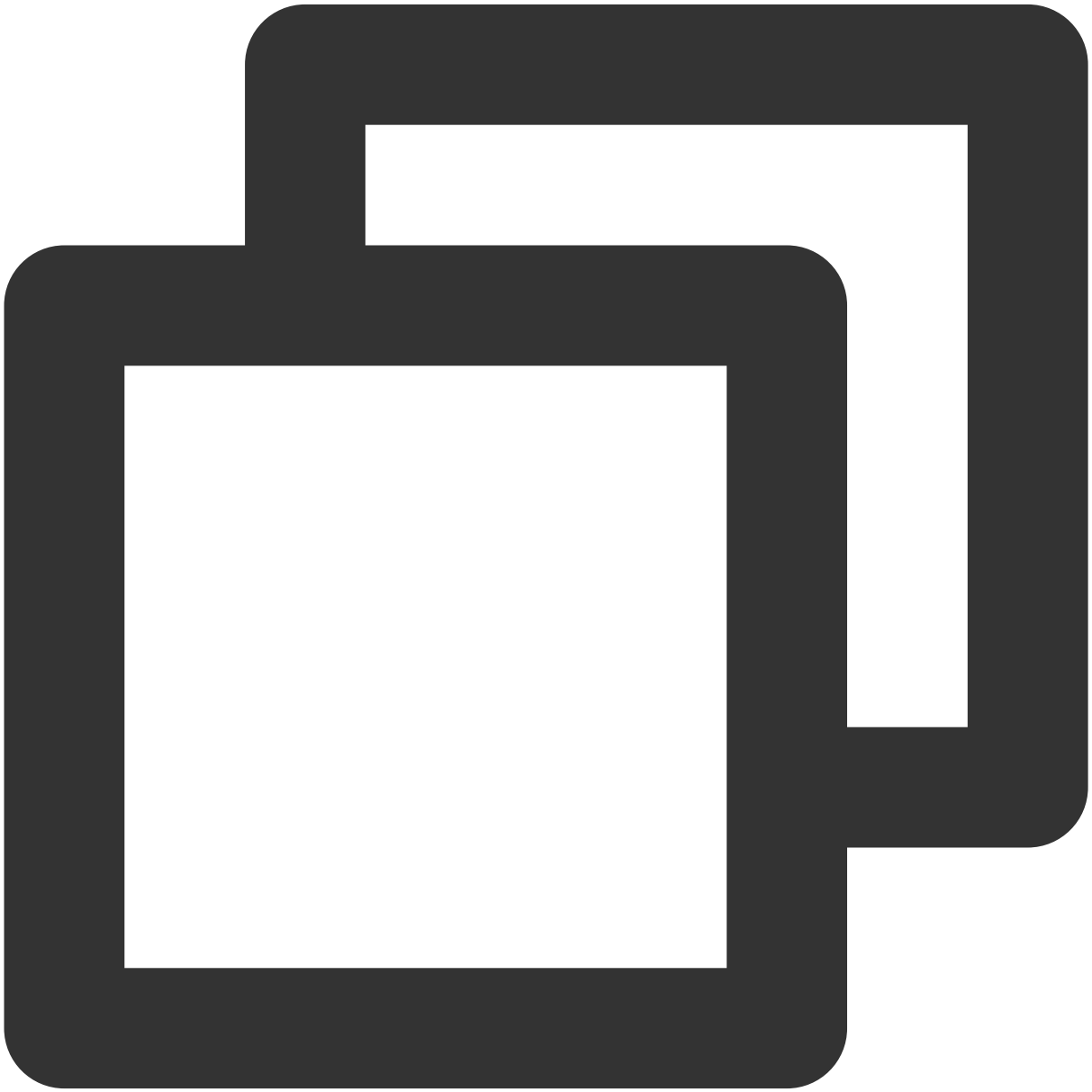
参数名称	描述	类型	是否必选
is_pic_info	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	Int	否
rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	Array	否

rules（json 数组）中每一项具体参数如下：

参数名称	描述	类型	是否必选
bucket	存储结果的目标存储桶名称，格式为 BucketName-APPID，如果不指定的话默认保存到当前存储桶	String	否
fileid	处理结果的文件路径名称，如以 `/` 开头，则存入指定文件夹中，否则，存入原图文件存储的同目录	String	是
rule	处理参数，参见数据万象图片处理 API。若按指定样式处理，则以 `style/` 开头，后加样式名，如样式名为 `test`，则 rule 字段为 `style/test`	String	是

返回结果说明

获取原图片和处理信息，类型为 dict：



```
{
    'OriginalInfo': {
        'Key': 'local.jpg',
        'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/local.j
        'ETag': '"aff1b996bcc63a0f0259df0de2fa989f38c5ce7e"',
        'ImageInfo': {
            'Format': 'JPEG',
            'Width': '300',
            'Height': '168',
            'Quality': '74',
            'Ave': '0x1a3451',
```

```
            'Orientation': '0'
            'FrameCount': '1'
        }
    },
    'ProcessResults': {
        'Object': {
            'Key': 'format.png',
            'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/for
            'Format': 'png',
            'Width': '300',
            'Height': '168',
            'Size': '77063',
            'Quality': '74',
            'ETag': '"a07dc5bcfa238e7d23d2d5884da4ac328aaaa9c6"'
            'FrameCount': '1'
        }
    }
}
```

响应包体具体数据内容如下：

参数名称	类型	描述
UploadResult	Container	原图信息

UploadResult 节点内容：

参数名称	类型	描述
OriginalInfo	Container	原图信息
ProcessResults	Container	图片处理结果

OriginalInfo 节点内容：

节点名称	类型	描述
Key	String	原图文件名
Location	String	图片路径
ImageInfo	Container	原图图片信息
ETag	String	原图 ETag 信息（若处理结果图覆盖原图则为结果图 ETag 信息）

ImageInfo 节点内容：

节点名称	类型	描述
Format	String	格式
Width	Int	图片宽度
Height	Int	图片高度
Quality	Int	图片质量
Ave	String	图片主色调
Orientation	Int	图片旋转角度
FrameCount	Int	图片帧数，静态图默认为1

ProcessResults 节点内容：

节点名称	类型	描述
Object	Container	每一个图片处理结果

Object 节点内容：

节点名称	类型	描述
Key	String	文件名
Location	String	图片路径
Format	String	图片格式
Width	Int	图片宽度
Height	Int	图片高度
Size	Int	图片大小
Quality	Int	图片质量
ETag	String	处理结果图 ETag 信息
FrameCount	Int	图片帧数，静态图默认为1

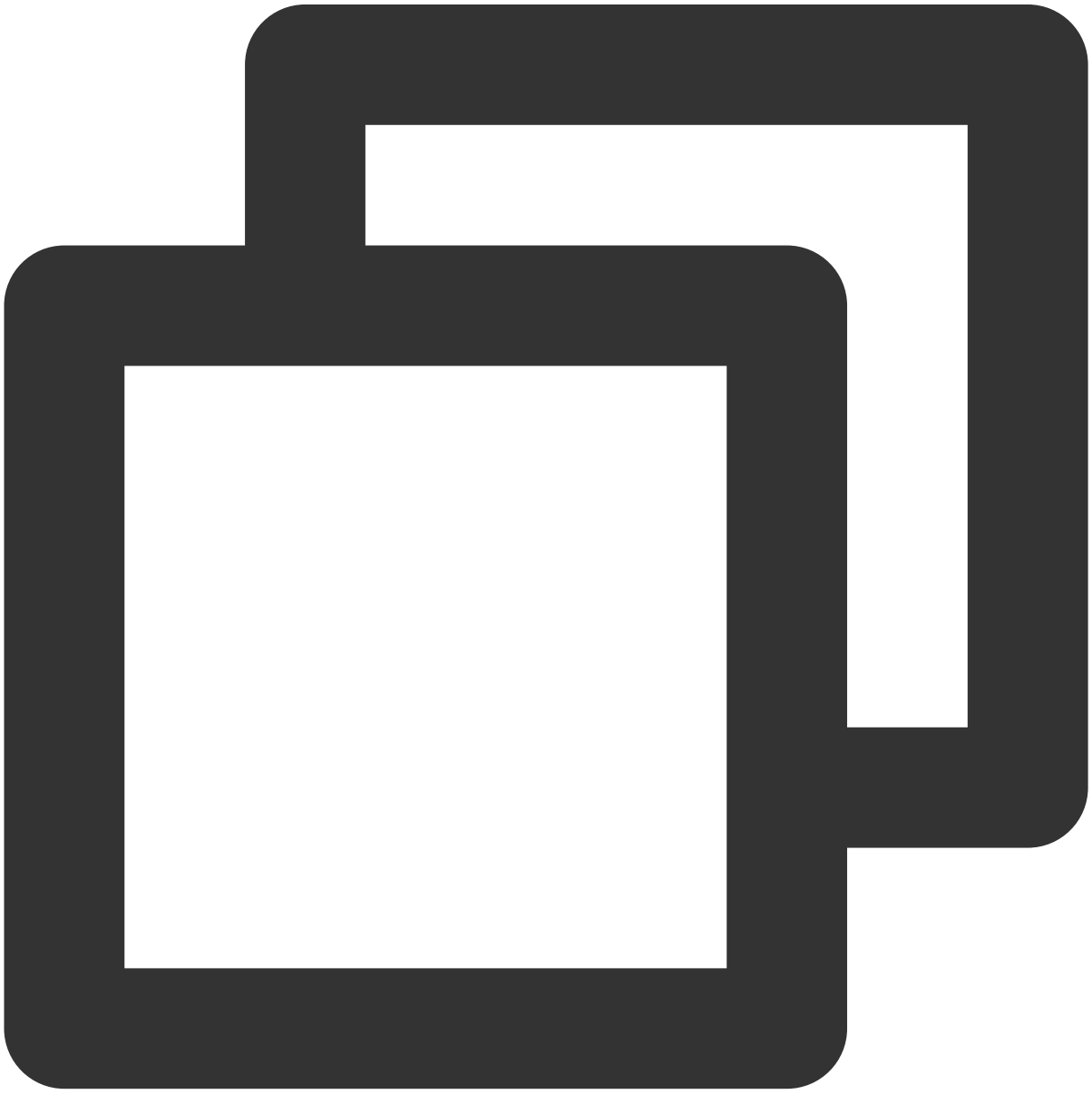
云上数据处理

版权所有：腾讯云计算（北京）有限责任公司

功能说明

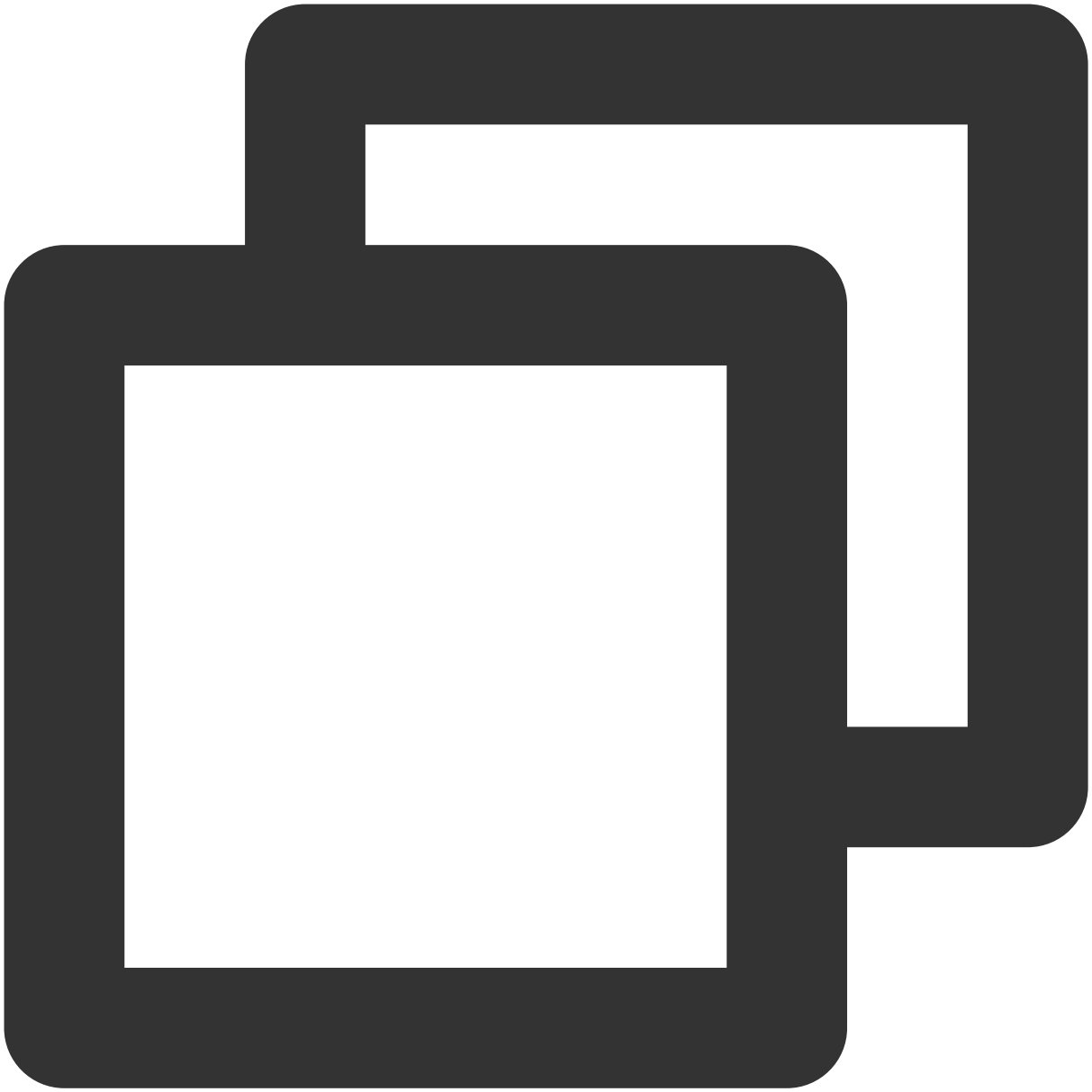
图片持久化处理 API 能够对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

方法原型



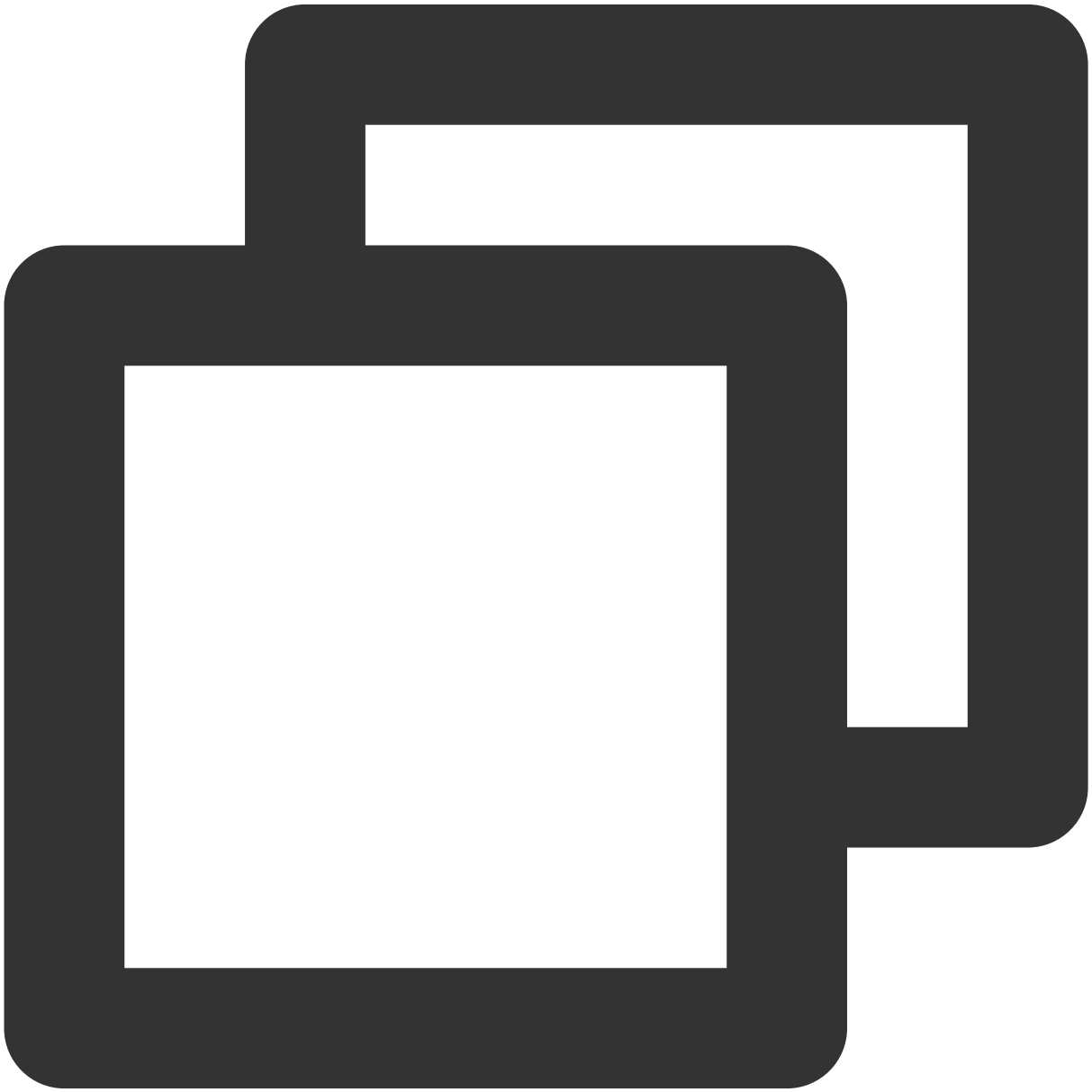
```
ci_image_process(self, Bucket, Key, **kwargs)
```

请求示例



```
response, data = client.ci_image_process(
    Bucket='examplebucket-1250000000',
     Key=ci_file_name,
    # pic operation json struct
    PicOperations='{"is_pic_info":1,"rules":[{"fileid":"format.png","rule":"imageVi
```

全部参数请求示例



```
response, data = client.ci_image_process(
    Bucket='examplebucket-1250000000',
    Key=ci_file_name,
    # pic operation json struct
    PicOperations='{"is_pic_info":1,"rules":[{"fileid":"format.png","rule":"imageVi
```

参数说明

参数名称	参数描述	类型	是否必填

Bucket	存储桶名称，由 BucketName-APPID 构成	String	是
Key	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	String	是
PicOperations	万象图片处理参数，请参见 图片持久化处理	String	是

PicOperations 为 json 格式的字符串，具体参数如下：

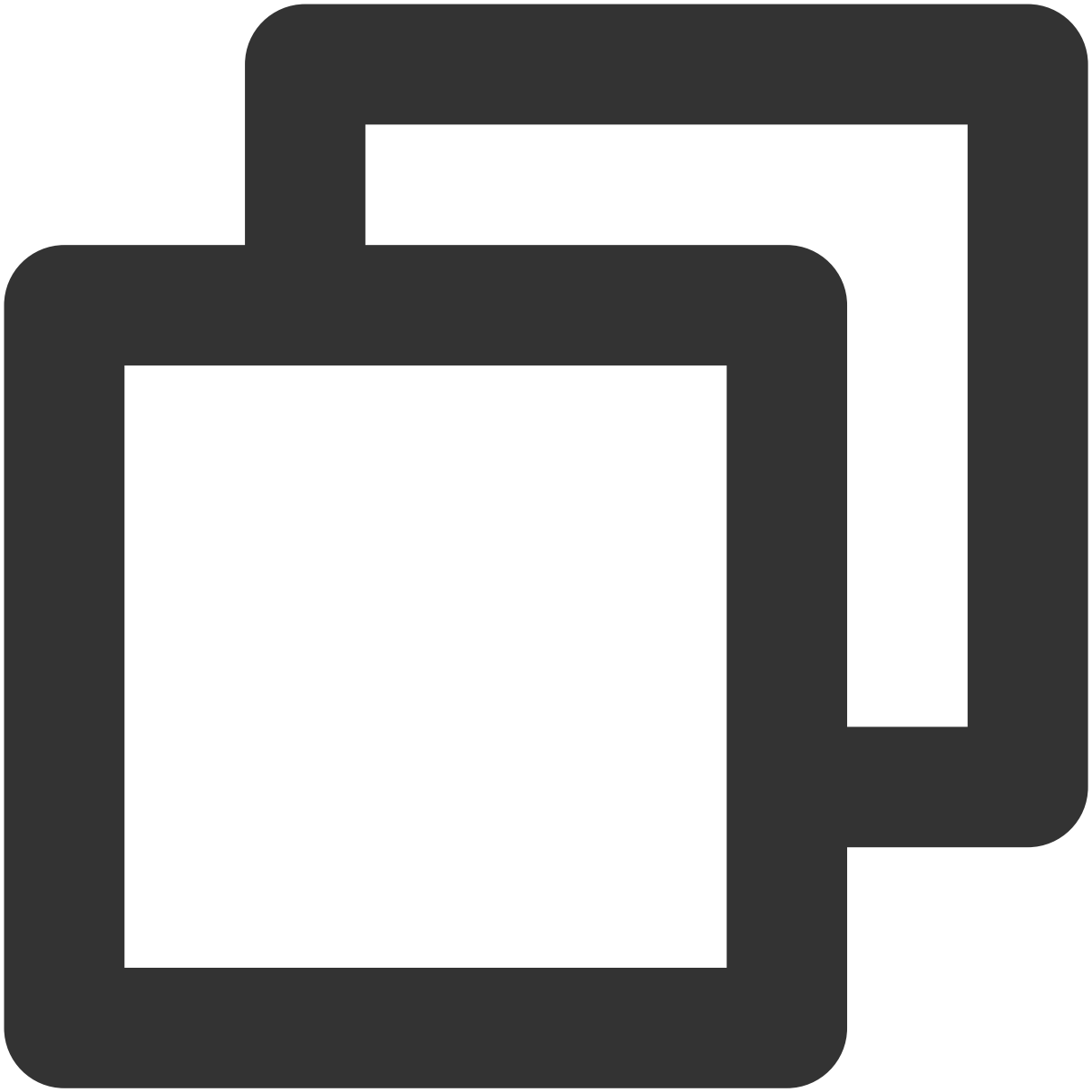
参数名称	描述	类型	是否必选
is_pic_info	是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0	Int	否
rules	处理规则，一条规则对应一个处理结果（目前支持五条规则），不填则不进行图片处理	Array	否

rules（json 数组）中每一项具体参数如下：

参数名称	描述	类型	是否必选
bucket	存储结果的目标存储桶名称，格式为 BucketName-APPID，如果不指定的话默认保存到当前存储桶	String	否
fileid	处理结果的文件路径名称，如以 `/` 开头，则存入指定文件夹中，否则，存入原图文件存储的同目录	String	是
rule	处理参数，可参见图片处理 API。若按指定样式处理，则以 `style/` 开头，后加样式名，如样式名为 `test`，则 rule 字段为 `style/test`	String	是

返回结果说明

获取对象的元信息，类型为 dict：



```
{
    'OriginalInfo': {
        'Key': 'local.jpg',
        'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/local.j
        'ETag': '"aff1b996bcc63a0f0259df0de2fa989f38c5ce7e"',
        'ImageInfo': {
            'Format': 'JPEG',
            'Width': '300',
            'Height': '168',
            'Quality': '74',
            'Ave': '0x1a3451',
```

```
            'Orientation': '0'
            'FrameCount': '1'
        }
    },
    'ProcessResults': {
        'Object': {
            'Key': 'format.png',
            'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/for
            'Format': 'png',
            'Width': '300',
            'Height': '168',
            'Size': '77063',
            'Quality': '74',
            'ETag': '"a07dc5bcfa238e7d23d2d5884da4ac328aaaa9c6"'
            'FrameCount': '1'
        }
    }
}
```

响应包体具体数据内容如下：

参数名称	类型	描述
UploadResult	Container	原图信息

UploadResult 节点内容：

参数名称	类型	描述
OriginalInfo	Container	原图信息
ProcessResults	Container	图片处理结果

OriginalInfo 节点内容：

节点名称	类型	描述
Key	String	原图文件名
Location	String	图片路径
ImageInfo	Container	原图图片信息
ETag	String	原图 ETag 信息（若处理结果图覆盖原图则为结果图 ETag 信息）

ImageInfo 节点内容：

节点名称	类型	描述
Format	String	格式
Width	Int	图片宽度
Height	Int	图片高度
Quality	Int	图片质量
Ave	String	图片主色调
Orientation	Int	图片旋转角度
FrameCount	Int	图片帧数，静态图默认为1

ProcessResults 节点内容：

节点名称	类型	描述
Object	Container	每一个图片处理结果

Object 节点内容：

节点名称	类型	描述
Key	String	文件名
Location	String	图片路径
Format	String	图片格式
Width	Int	图片宽度
Height	Int	图片高度
Size	Int	图片大小
Quality	Int	图片质量
ETag	String	处理结果图 ETag 信息
FrameCount	Int	图片帧数，静态图默认为1

版权所有：腾讯云计算（北京）有限责任公司

图片高级压缩

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于图片高级压缩的 API 概览以及 SDK 示例代码。

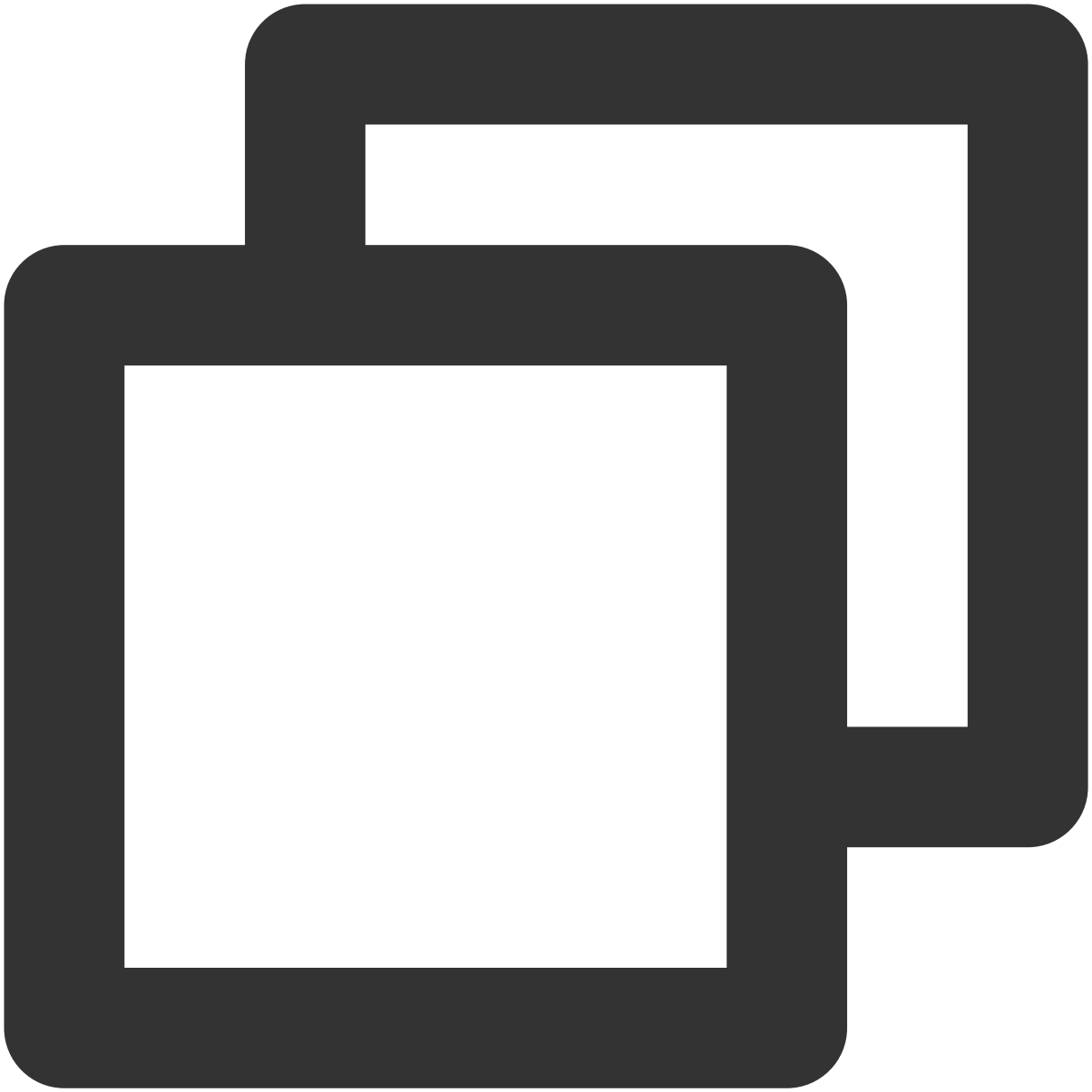
API	操作描述
图片高级压缩	对指定存储桶下的图片进行压缩

图片高级压缩

功能说明

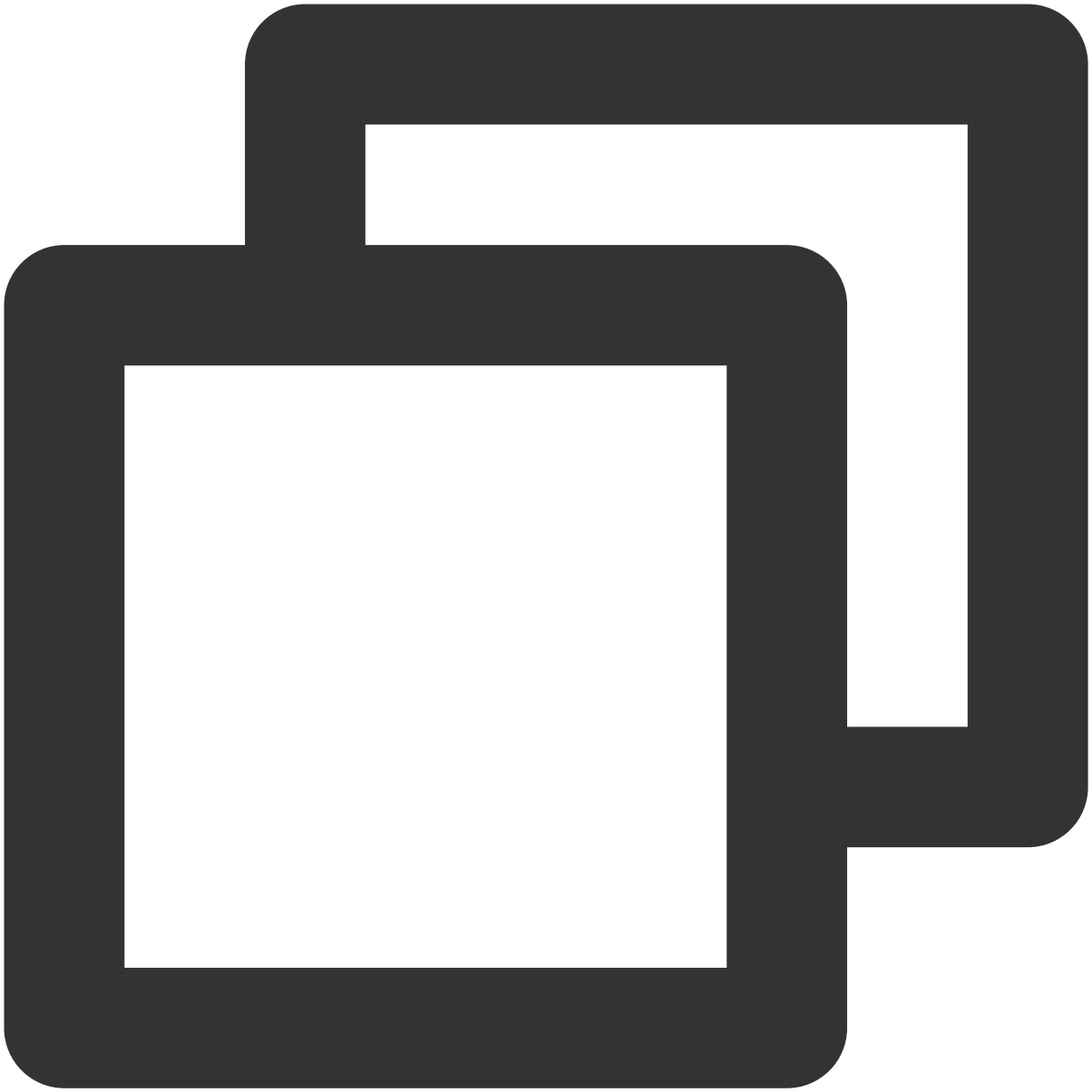
通过数据万象 imageMogr2 接口提供图片高级压缩功能。

方法原型



```
ci_download_compress_image(self, Bucket, Key, DestImagePath, CompressType, **kwargs
```

请求示例



```
# TPG 压缩
response = client.ci_download_compress_image(
    Bucket='examplebucket-1250000000',
    Key='sample.png',
    DestImagePath='sample.tpg',
    CompressType='tpg'
)

# HEIF 压缩
response = client.ci_download_compress_image(
    Bucket='examplebucket-1250000000',
```

```
    Key='sample.png',
    DestImagePath='sample.heif',
    CompressType='heif'
)
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶名称，格式为 BucketName-APPID ，详情请参见 命名规范	String	是
Key	对象Key，长度不超过128字节, 支持英文字母、数字、空格、加号、减号、下划线、等号、点号、冒号、斜线	String	是
DestImagePath	压缩图片保存至本地的目的路径	String	是
CompressType	压缩格式，支持 TPG 或 HEIF	String	是

返回结果说明

该方法返回值为请求响应头部。

盲水印

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于盲水印的 API 概览以及 SDK 示例代码。

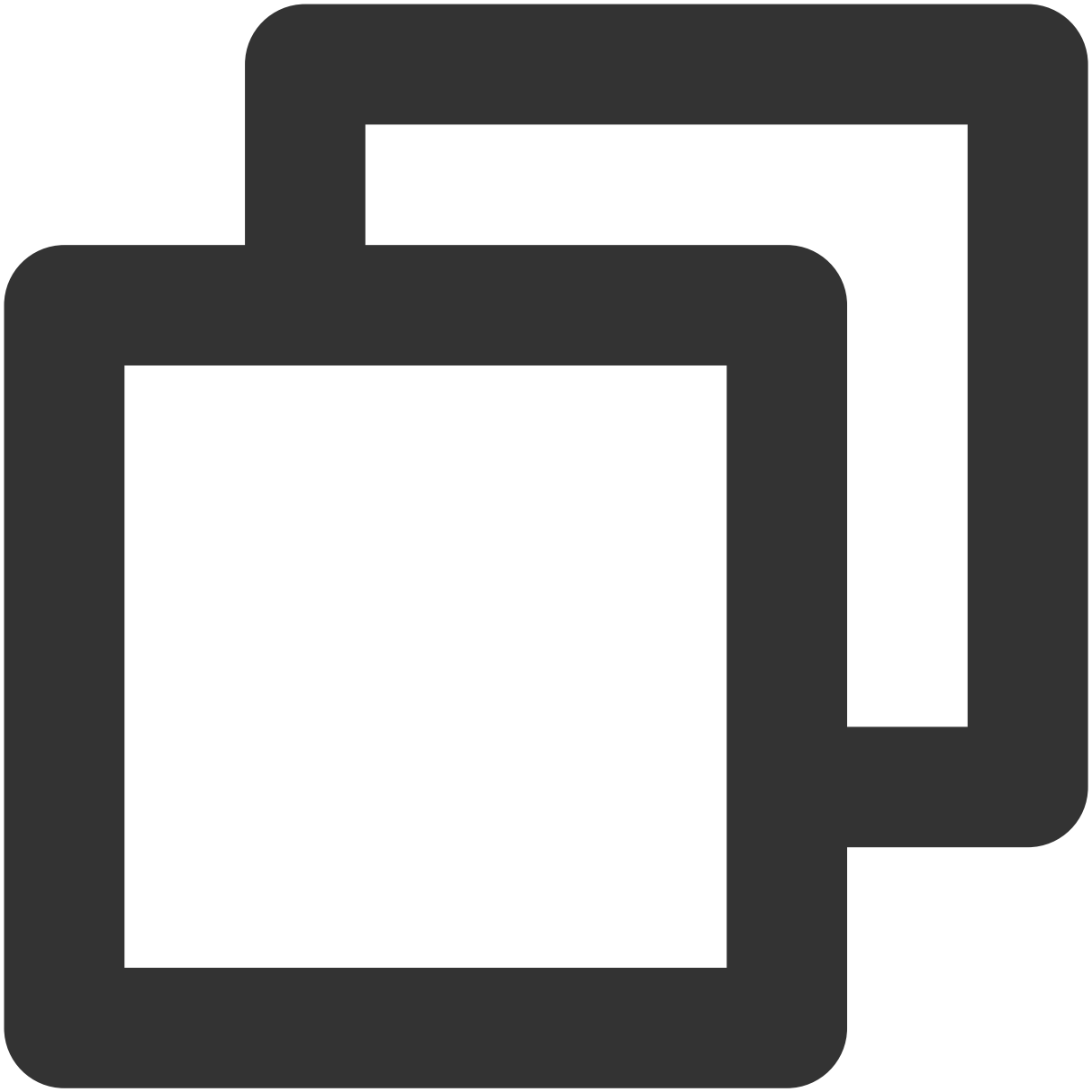
API	操作描述
盲水印	对本地图片添加或提取盲水印并上传至存储桶

盲水印

功能说明

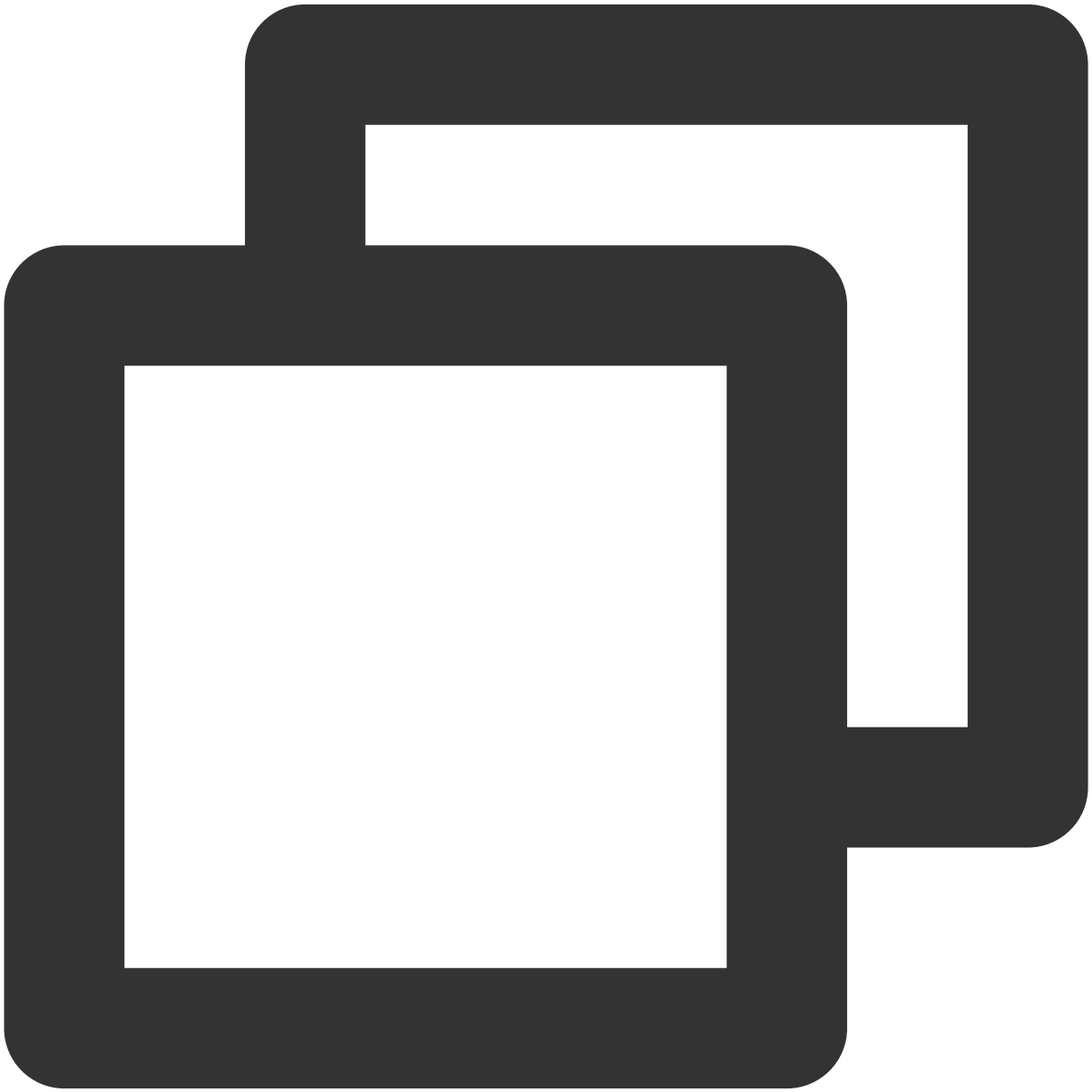
盲水印功能基于腾讯云数据万象，是一种全新的水印模式。

方法原型



```
def ci_put_object_from_local_file(self, Bucket, LocalFilePath, Key, EnableMD5=False
```

请求示例



```
# 添加盲水印
watermark_url = 'http://{bucket}.cos.{region}.myqcloud.com/watermark.png'.format(bu
watermark_url_base64 = bytes.decode(base64.b64encode(str.encode(watermark_url)))
print(watermark_url_base64)
response, data = client.ci_put_object_from_local_file(
    Bucket='examplebucket-1250000000',
    LocalFilePath='sample.png',
    Key="sample.png",
    # pic operation json struct
    PicOperations='{"is_pic_info":1,"rules":[{"fileid": "format.png","rule": "water
                watermark_url_base64 + '" }]}'
```

```
)

# 提取盲水印
sample_url = 'http://{bucket}.cos.{region}.myqcloud.com/sample.png'.format(bucket='
sample_url_base64 = bytes.decode(base64.b64encode(str.encode(sample_url)))
response, data = client.ci_put_object_from_local_file(
    Bucket='examplebucket-1250000000',
    LocalFilePath='format.png',
    Key="format.png",
    # pic operation json struct
    PicOperations='{"is_pic_info":1,"rules":[{"fileid": "watermark.png","rule": "wa
                  sample_url_base64 + '" }]}'
)
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶名称，格式为 BucketName-APPID ，详情请参见 命名规范	String	是
LocalFilePath	需要处理的本地图片路径	String	是
Key	对象键，长度不超过128字节, 支持英文字母、数字、空格、加号、减号、下划线、等号、点号、冒号、斜线	String	是
EnableMD5	开启对象上传的 MD5 校验	Bool	否

返回结果说明

该方法返回值为请求响应头部和内容。

媒体处理
媒体信息接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

注意：

需要 COS PYTHON SDK v5.1.9.11 及以上版本。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

注意：

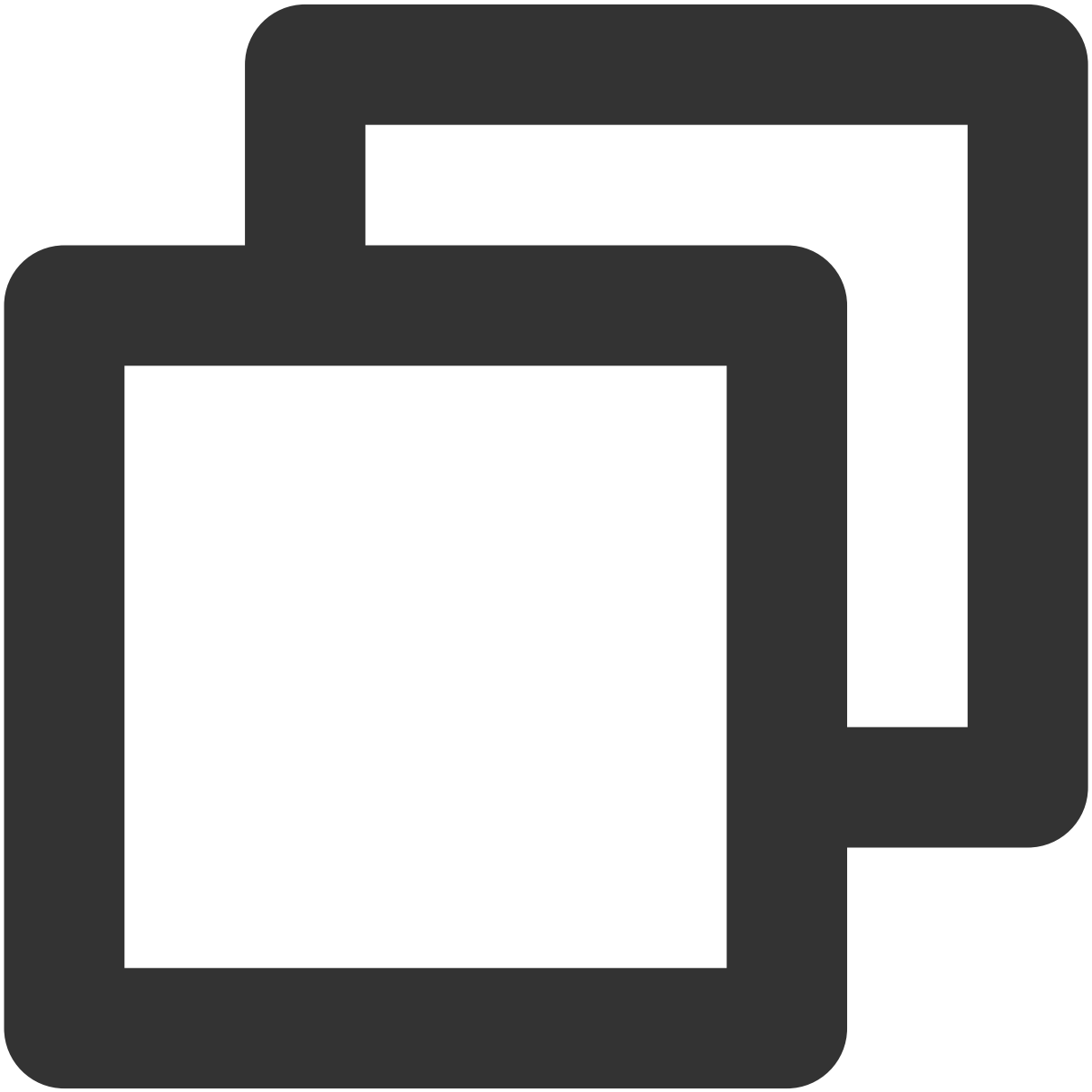
使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。详情请参见 开通媒体处理。

查询文件信息

功能说明

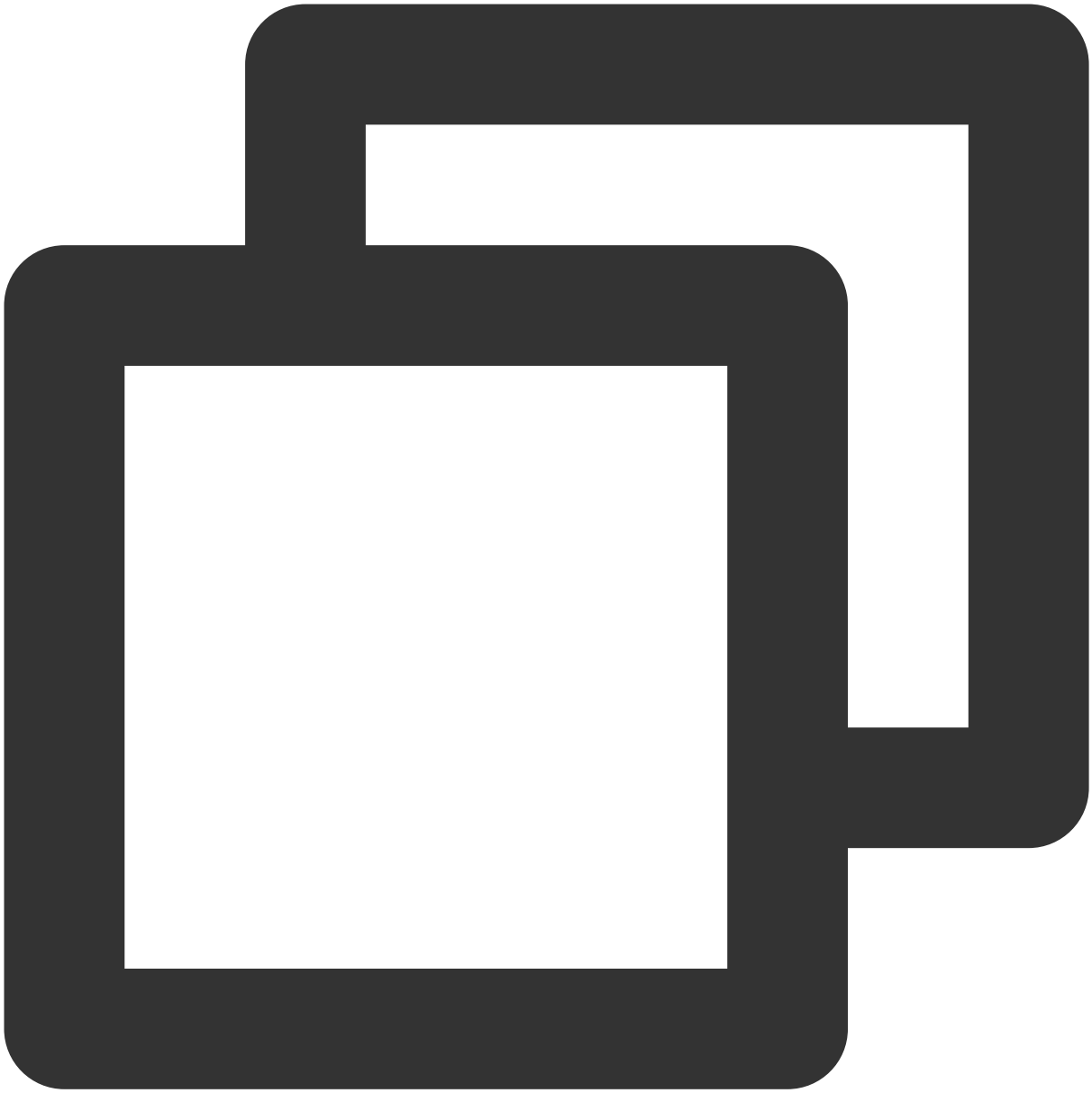
用于查询媒体文件的信息。

方法原型



```
def get_media_info(Bucket, Key, **kwargs)
```

请求示例



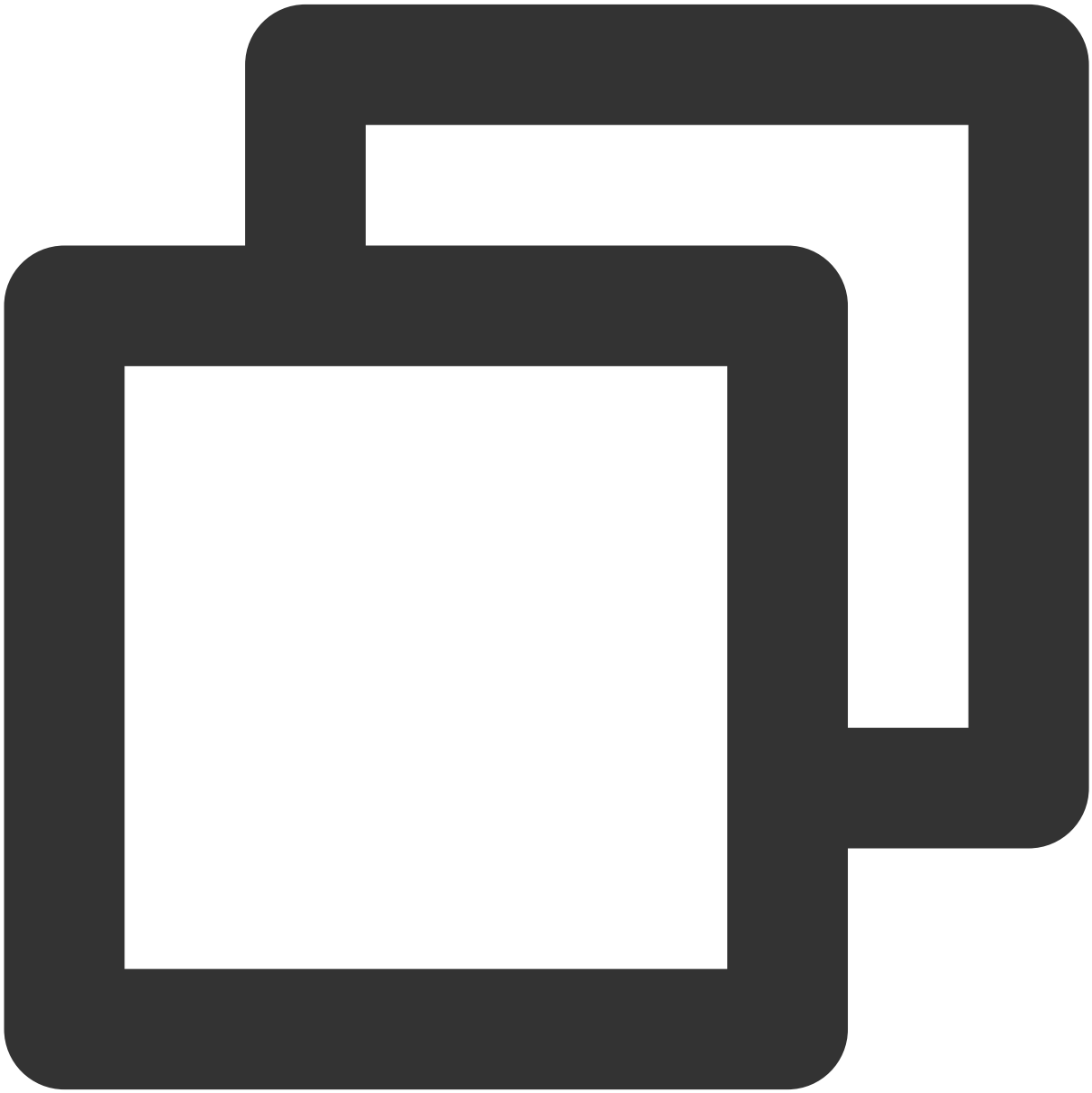
```
response = client.get_media_info(
    Bucket=bucket_name,
    Key='demo.mp4'
)
print(response)
```

参数说明

参数名称	参数描述	是否必填	类型

Bucket	存储桶名称，格式为 BucketName-APPID ， 详情请参见 命名规范	是	String
Key	对象键（Key）是对象在存储桶中的唯一标识。	是	String

返回结果说明



```
{
    'MediaInfo': [{
        'Format': {
            'Bitrate': '16869.432000',
            'Duration': '129.200000',
```

```
            'FormatLongName': 'QuickTime / MOV',
            'FormatName': 'mov,mp4,m4a,3gp,3g2,mj2',
            'NumProgram': '0',
            'NumStream': '2',
            'Size': '272441346',
            'StartTime': '0.000000'
        },
        'Stream': {
            'Audio': {
                'Bitrate': '125.712000',
                'Channel': '2',
                'ChannelLayout': 'stereo',
                'CodecLongName': 'AAC (Advanced Audio Coding)',
                'CodecName': 'aac',
                'CodecTag': '0x6134706d',
                'CodecTagString': 'mp4a',
                'CodecTimeBase': '1/44100',
                'Duration': '129.160998',
                'Index': '0',
                'Language': 'und',
                'SampleFmt': 'fltp',
                'SampleRate': '44100',
                'StartTime': '0.000000',
                'Timebase': '1/44100'
            },
            'Subtitle': None,
            'Video': {
                'AvgFps': '30/1',
                'Bitrate': '16738.543000',
                'CodecLongName': 'H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10',
                'CodecName': 'h264',
                'CodecTag': '0x31637661',
                'CodecTagString': 'avc1',
                'CodecTimeBase': '1/600',
                'Duration': '129.200000',
                'Fps': '30.500000',
                'HasBFrame': '1',
                'Height': '1920',
                'Index': '1',
                'Language': 'und',
                'Level': '40',
                'NumFrames': '3876',
                'PixFormat': 'yuvj420p',
                'Profile': 'High',
                'RefFrames': '1',
                'Rotation': '0.000000',
                'StartTime': '0.000000',
```


版权所有：腾讯云计算（北京）有限责任公司

```
                'Timebase': '1/600',
                'Width': '1080'
            }
        }
    }]
}
```

具体的数据内容如下：

MediaInfo 的内容：

节点名称（关键字）	父节点	描述	类型
Stream	MediaInfo	流信息	dict
Format	MediaInfo	格式信息	dict

Stream 的内容：

节点名称（关键字）	父节点	描述	类型
Video	MediaInfo.Stream	视频信息	dict
Audio	MediaInfo.Stream	音频信息	dict
Subtitle	MediaInfo.Stream	字幕信息	dict

Format 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
NumStream	MediaInfo.Format	Stream（包含 Video、Audio、Subtitle）的数量	String
NumProgram	MediaInfo.Format	节目的数量	String
FormatName	MediaInfo.Format	容器格式名字	String
FormatLongName	MediaInfo.Format	容器格式的详细名字	String
StartTime	MediaInfo.Format	起始时间，单位为秒	String
Duration	MediaInfo.Format	时长，单位为秒	String
Bitrate	MediaInfo.Format	比特率，单位为 kbps	String
Size	MediaInfo.Format	大小，单位为 Byte	String

版权所有：腾讯云计算（北京）有限责任公司

Video 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
Index	MediaInfo.Stream.Video	该流的编号	String
CodecName	MediaInfo.Stream.Video	编解码格式名字	String
CodecLongName	MediaInfo.Stream.Video	编解码格式的详细名字	String
CodecTimeBase	MediaInfo.Stream.Video	编码时基	String
CodecTagString	MediaInfo.Stream.Video	编码标签名	String
CodecTag	MediaInfo.Stream.Video	编码标签	String
Profile	MediaInfo.Stream.Video	视频编码档位	String
Height	MediaInfo.Stream.Video	视频高，单位 px	String
Width	MediaInfo.Stream.Video	视频宽，单位 px	String
HasBFrame	MediaInfo.Stream.Video	是否有B帧。1表示有，0表示无	String
RefFrames	MediaInfo.Stream.Video	视频编码的参考帧个数	String
Sar	MediaInfo.Stream.Video	采样宽高比	String
Dar	MediaInfo.Stream.Video	显示宽高比	String
PixFormat	MediaInfo.Stream.Video	像素格式	String
FieldOrder	MediaInfo.Stream.Video	场的顺序	String
Level	MediaInfo.Stream.Video	视频编码等级	String
Fps	MediaInfo.Stream.Video	视频帧率	String
AvgFps	MediaInfo.Stream.Video	平均帧率	String
Timebase	MediaInfo.Stream.Video	时基	String
StartTime	MediaInfo.Stream.Video	视频开始时间，单位为秒	String
Duration	MediaInfo.Stream.Video	视频时长，单位为秒	String
Bitrate	MediaInfo.Stream.Video	比特率，单位为 kbps	String
NumFrames	MediaInfo.Stream.Video	总帧数	String

版权所有：腾讯云计算（北京）有限责任公司

Language	MediaInfo.Stream.Video	语言	String

Audio 的内容（查询视频信息时，可能部分字段未返回）：

节点名称（关键字）	父节点	描述	类型
Index	MediaInfo.Stream.Audio	该流的编号	String
CodecName	MediaInfo.Stream.Audio	编解码格式名字	String
CodecLongName	MediaInfo.Stream.Audio	编解码格式的详细名字	String
CodecTimeBase	MediaInfo.Stream.Audio	编码时基	String
CodecTagString	MediaInfo.Stream.Audio	编码标签名	String
CodecTag	MediaInfo.Stream.Audio	编码标签	String
SampleFmt	MediaInfo.Stream.Audio	采样格式	String
SampleRate	MediaInfo.Stream.Audio	采样率	String
Channel	MediaInfo.Stream.Audio	通道数量	String
ChannelLayout	MediaInfo.Stream.Audio	通道格式	String
Timebase	MediaInfo.Stream.Audio	时基	String
StartTime	MediaInfo.Stream.Audio	音频开始时间，单位秒	String
Duration	MediaInfo.Stream.Audio	音频时长，单位秒	String
Bitrate	MediaInfo.Stream.Audio	比特率，单位 kbps	String
Language	MediaInfo.Stream.Audio	语言	String

Subtitle 的内容：

节点名称（关键字）	父节点	描述	类型
Index	MediaInfo.Stream.Subtitle	该流的编号	String
Language	MediaInfo.Stream.Subtitle	语言，und 表示无查询结果	String

版权所有：腾讯云计算（北京）有限责任公司

媒体截图接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

注意：

需要 COS PYTHON SDK v5.1.9.11 及以上版本。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

注意：

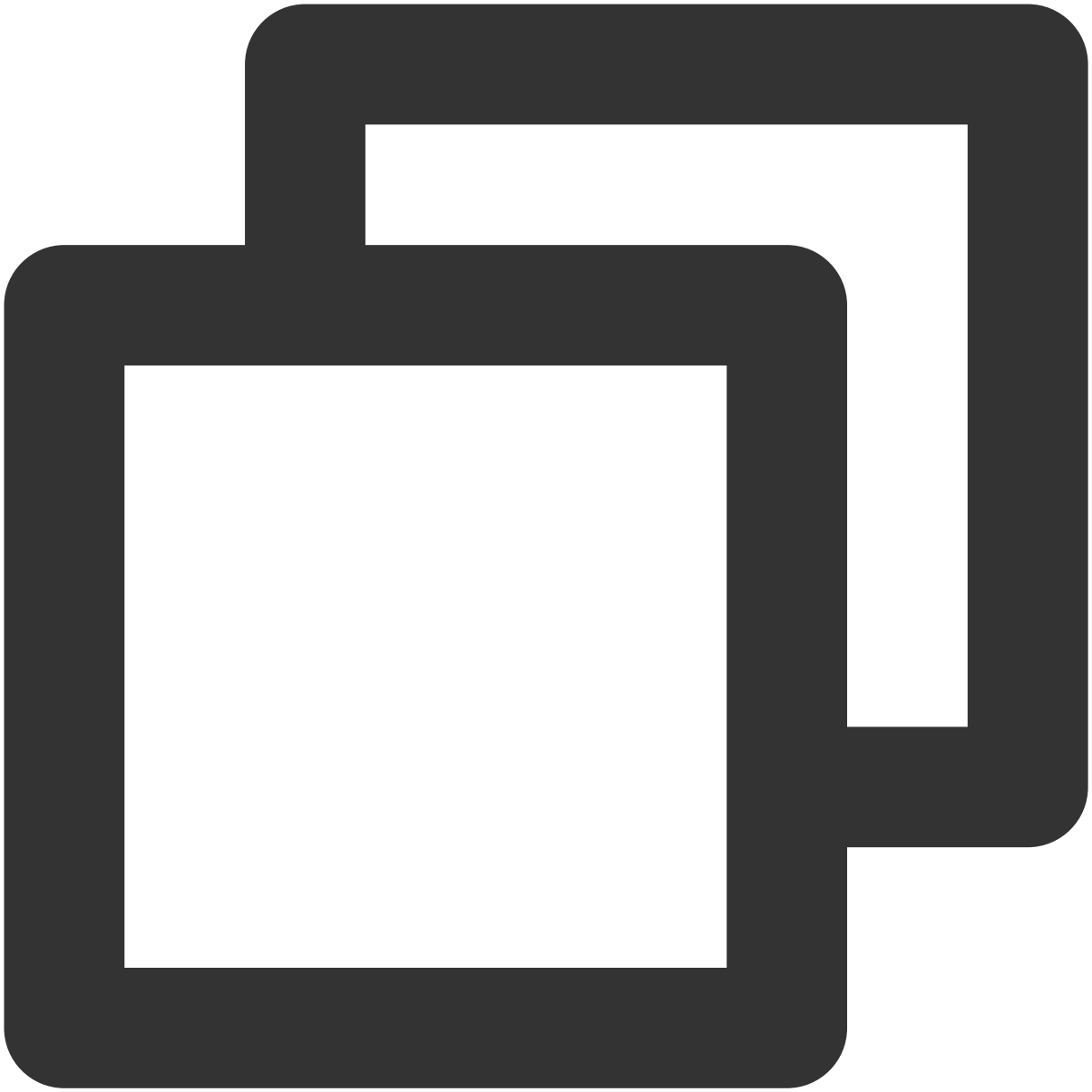
使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。详情请参见 开通媒体处理。

查询截图

功能说明

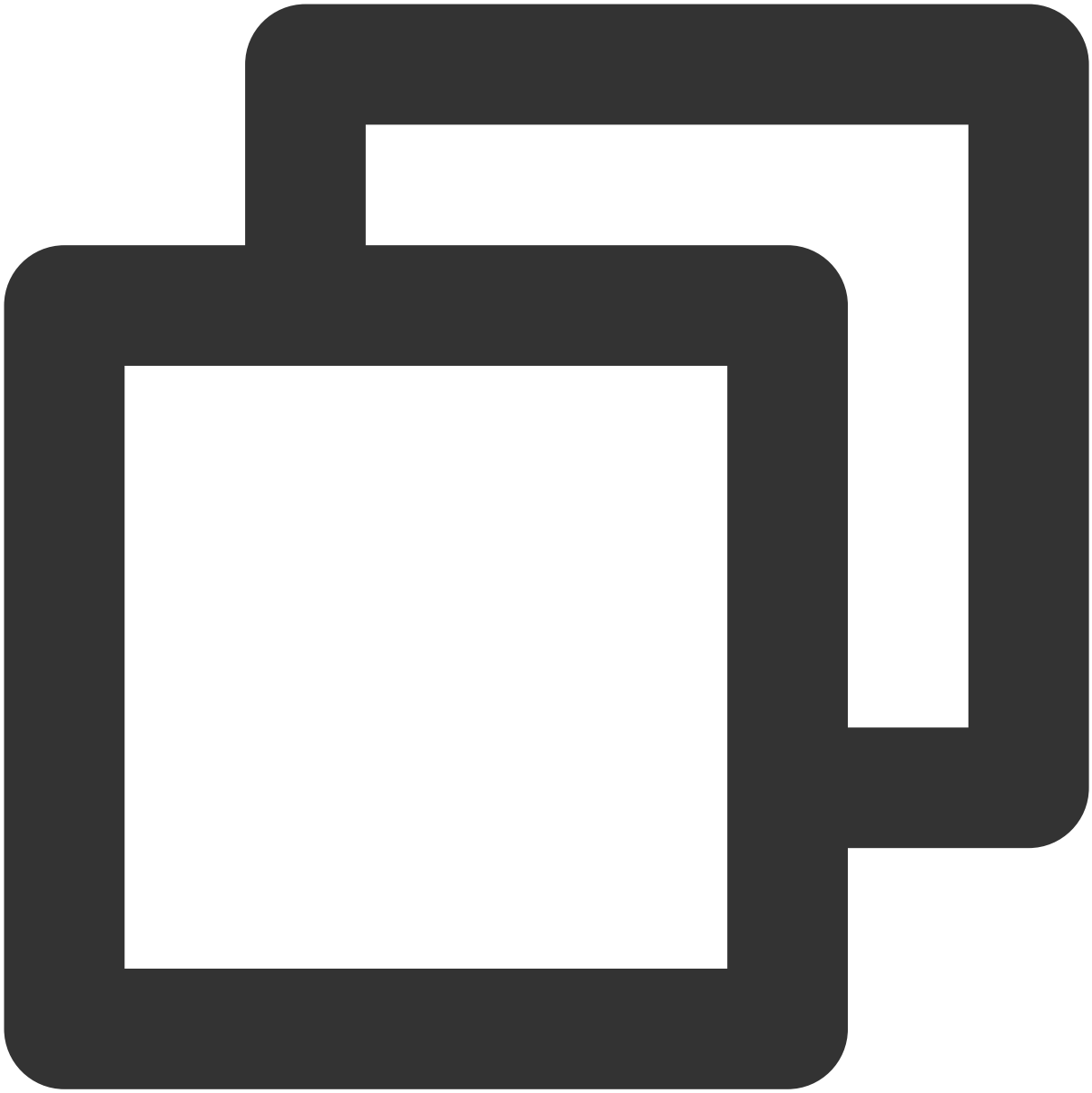
用于查询媒体文件在某个时间的截图。

方法原型



```
def get_snapshot(Bucket, Key, Time, Width=None, Height=None, Format='jpg', Rotate='
```

请求示例



```
response = client.get_snapshot(
    Bucket='examplebucket-1250000000',
    Key='demo.mp4',
    Time='1.5',
    Width='480',
    Format='png'
)
print(response)
response['Body'].get_stream_to_file('snapshot.jpg')
```

参数说明

参数名称	参数描述	是否必填	类型
Bucket	存储桶名称，格式为 BucketName-APPID ，详情请参见 命名规范	是	String
Key	对象键（Key）是对象在存储桶中的唯一标识	是	String
Time	截图的时间点，单位为秒，可以支持小数点	是	String
Width	截图的宽。默认为0	否	String
Height	截图的高。默认为0 当 width 和 height 都为0时，表示使用视频的宽高；如果单个为0，则以另外一个值按视频宽高比例自动适应	否	String
Format	截图的格式，支持 jpg 和 png，默认 jpg	否	String
Rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转off：不旋转，默认值为 auto	否	String
Mode	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，默认值为 exactframe	否	String

版权所有：腾讯云计算（北京）有限责任公司

内容审核
图片审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供图片审核相关的 API 概览以及 SDK 示例代码。

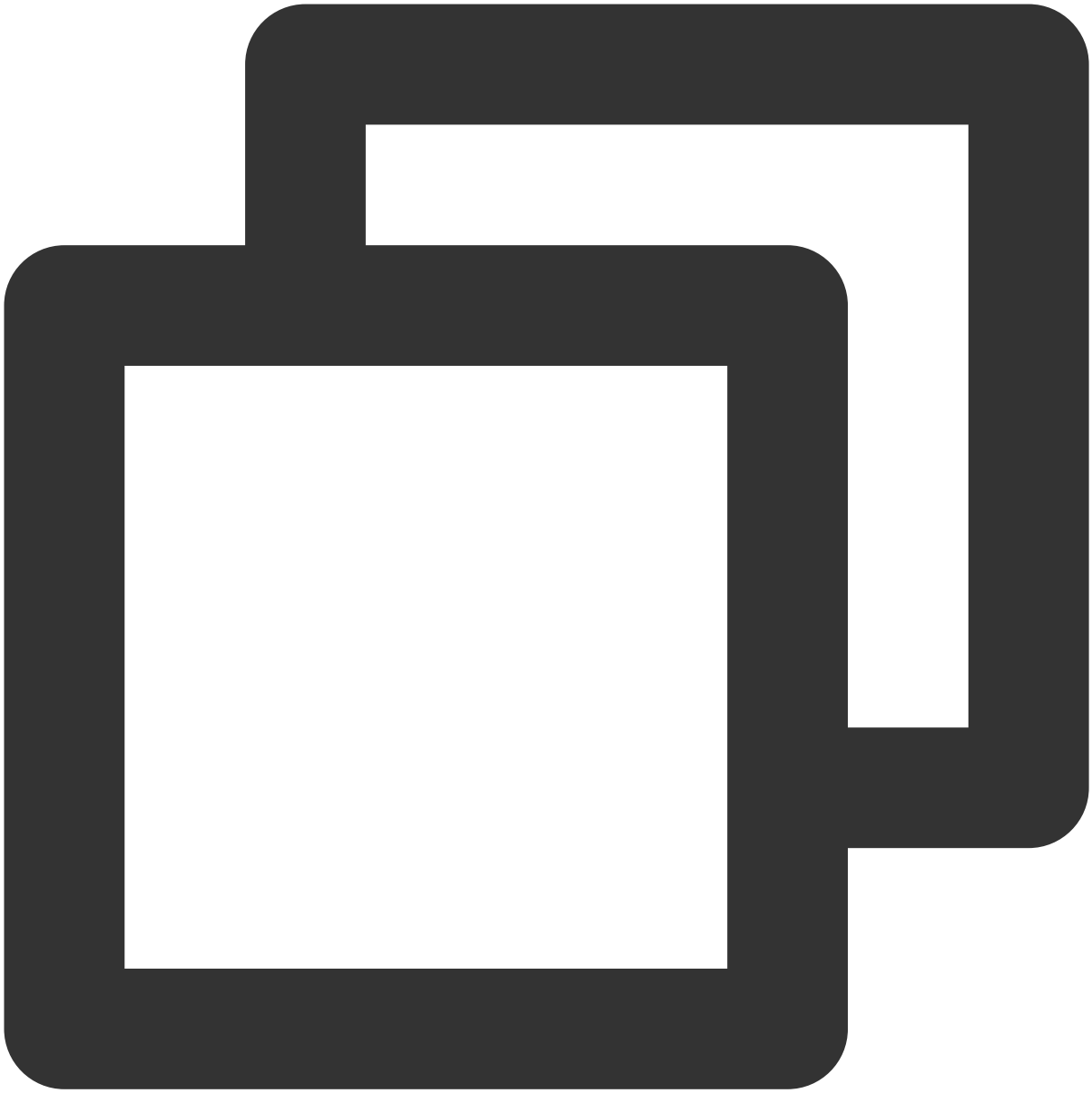
API	操作描述
图片单次审核	图片审核的存量扫描功能借助数据万象的持久化处理接口，实现对对象存储（Cloud Object Storage，COS）存量数据的涉黄、违法违规以及广告引导类图片的扫描。内容审核请求包属于 GET 请求。
图片批量审核	图片批量审核接口为同步请求方式，您可以通过本接口对多个图片文件进行内容审核。该接口属于 POST 请求。

图片单次审核

功能说明

对 COS 上的图片进行审核，审核类型包括涉黄、违法违规以及广告引导类。

示例代码



```
"""测试ci文件内容识别的接口"""
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
from qcloud_cos.cos_comm import CiDetectType
# 先创建 cos client
response = client.get_object_sensitive_content_recognition(
    Bucket=test_bucket,
    Key=test_object,
    DetectUrl='https://examplebucket-1250000000.cos.ap-chongqing.myqcloud.com/t
    DetectType=(CiDetectType.PORN | CiDetectType.ADS)
)
```

```
    print(response)
```

参数说明

调用 get_object_sensitive_content_recognition 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Key	对象文件名，例如 picture.jpg。	Sting	是
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	Sting	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对图片进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Interval	截帧频率，GIF 图/长图检测专用，默认值为0，表示只会检测 GIF 图/长图的第一帧。	Int	否
MaxFrames	最大截帧数量，GIF 图/长图检测专用，默认值为1，表示只取 GIF 的第1帧图片进行审核，或长图不做切分识别。	Int	否
DetectUrl	您可以通过填写 detect-url 审核任意公网可访问的图片链接。不填写 detect-url 时，后台会默认审核 ObjectKey。填写了 detect-url 时，后台会审核 detect-url 链接，无需再填写 ObjectKey。detect-url 示例：http://www.example.com/abc.jpg。	Sting	否
LargeImageDetect	对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。注：图片压缩功能最大支持32MB的图片，且会收取图片压缩费用。	Int	否
dataid	图片标识，该字段在结果中返回原始内容，长度限制为512字节。	String	否

返回参数说明

调用 get_object_sensitive_content_recognition 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 图片同步审核接口 文档。

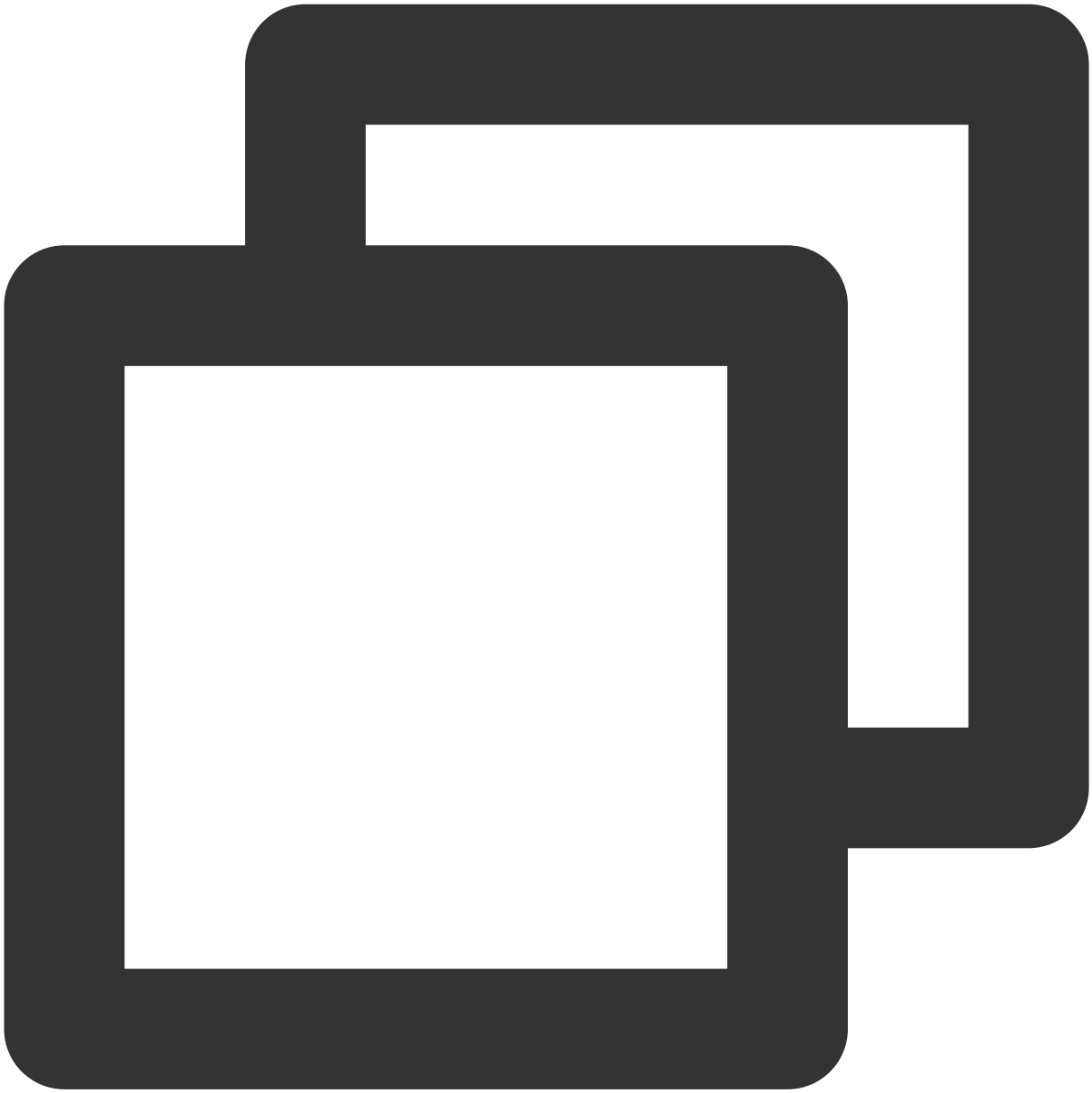
版权所有：腾讯云计算（北京）有限责任公司

图片批量审核

功能说明

对 COS 上的多张图片进行审核，审核类型包括涉黄、违法违规以及广告引导类。

示例代码



```
    """测试ci批量图片内容审核的接口"""
    from qcloud_cos import CosConfig
    from qcloud_cos import CosS3Client
```

```
from qcloud_cos.cos_comm import CiDetectType
# 先创建 cos client
response = client.ci_auditing_image_batch(
    Bucket=bucket_name,
    DetectType=CiDetectType.PORN,
    Input=[{
        'Url':'https://examplebucket-1250000000.cos.ap-chongqing.myqcloud.com/p
        'DataId': 'testdataid-111111',
        'UserInfo':{
        'TokenId': 'token',
        'Nickname': 'test',
        'DeviceId': 'DeviceId-test',
        'AppId': 'AppId-test',
        'Room': 'Room-test',
        'IP': 'IP-test',
        'Type': 'Type-test',
    },
    },{
        'Object':'test-0.jpg',
        'LargeImageDetect' :1,
    },{
        'Url':'https://examplebucket-1250000000.cos.ap-chongqing.myqcloud.com/p
    }]
)
print(response)
```

参数说明

调用 ci_auditing_image_batch 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	Sting	是
BizType	审核策略的唯一标识，由后台自动生成，在控制台中对应为 Biztype 值。	Sting	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对图片进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Input	需要审核的图片信息，每个 array 元素为 dict 类型，支持的参数如下：Object：存储在 COS 存储桶中的图片文件名称，例如在目录 test 中的文件 image.jpg，则文件名称为 test/image.jpg。Object 和 Url 只能选择其中一种。	Array	否

	Url：图片文件的链接地址，例如 http://a-1250000.cos.ap-shanghai.tencentcos.cn/image.jpg.Object 和 Url 只能选择其中一种。 Interval：截帧频率，GIF 图检测专用，默认值为5，表示从第一帧（包含）开始每隔5帧截取一帧 MaxFrames：最大截帧数量，GIF 图检测专用，默认值为5，表示只截取 GIF 的5帧图片进行审核，必须大于0。 DataId：图片标识，该字段在结果中返回原始内容，长度限制为512字节。 LargeImageDetect：对于超过大小限制的图片是否进行压缩后再审核，取值为：0（不压缩），1（压缩）。默认为0。注：图片压缩功能最大支持32MB的图片，且会收取图片压缩费用。 DataId：图片标识，该字段在结果中返回原始内容，长度限制为512字节。 UserInfo：用户业务字段。		

返回参数说明

调用 ci_auditing_image_batch 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 图片批量审核审核接口 文档。

版权所有：腾讯云计算（北京）有限责任公司

视频审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供视频审核相关的 API 概览以及 SDK 示例代码。

注意：

该功能需要 COS Python SDK v5.1.9.10 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

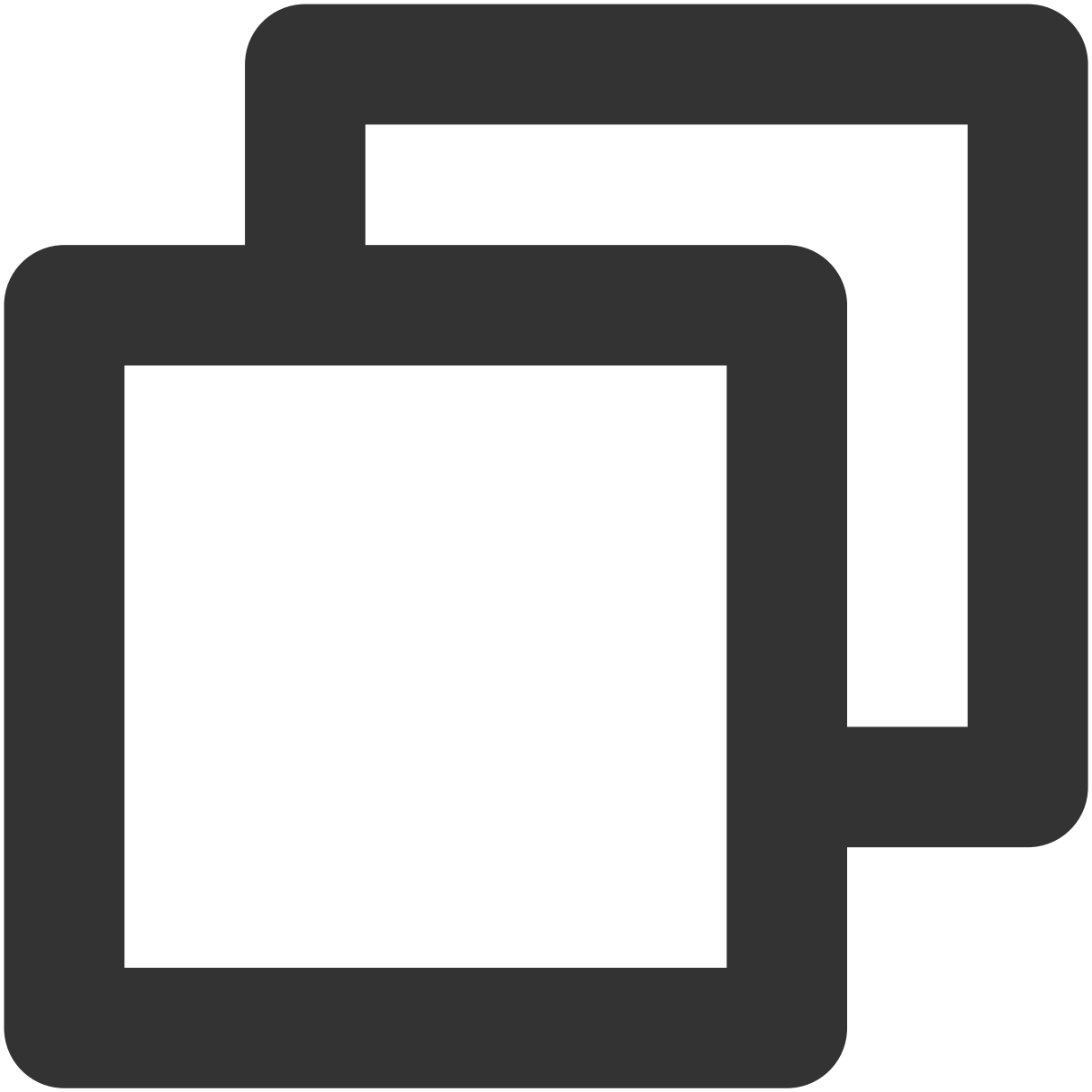
API	操作描述
提交视频审核任务	用于提交一个视频审核任务。
查询视频审核任务结果	用来查询指定的视频审核任务。

提交视频审核任务

功能说明

本接口用于提交一个视频审核任务。视频审核功能为异步任务方式，您可以通过提交视频审核任务来审核您的视频文件，然后通过查询视频审核任务接口查询审核结果。

示例代码



```
"""测试ci视频文件内容审核的接口"""
# 先创建 cos client
# 识别cos上的视频
response = client.ci_auditing_video_submit(
    Bucket='bucket',
    DetectType=CiDetectType.PORN | CiDetectType.ADS,
    Key='test.mp4'
)
print response
```

参数说明

调用 ci_auditing_video_submit 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
Key	对象文件名，例如 picture.jpg。	String	是
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	String	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对视频进行涉黄和广告识别审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Url	支持直接使用非 COS 上的视频 url 进行审核。	String	否
Callback	回调地址，以 http:// 或者 https:// 开头的地址。	String	否
CallbackVersion	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String	否
Mode	截帧模式。Interval 表示间隔模式；Average 表示平均模式；Fps 表示固定帧率模式。 Interval 模式：TimeInterval，Count 参数生效。当设置 Count，未设置 TimeInterval 时，表示截取所有帧，共 Count 张图片。 Average 模式：Count 参数生效。表示整个视频，按平均间隔截取共 Count 张图片。 Fps 模式：TimeInterval 表示每秒截取多少帧，Count 表示共截取多少帧	String	否
Count	视频截帧数量，范围为(0, 10000]。	String	否
TimeInterval	视频截帧频率，范围为(0, 60]，单位为秒，支持 float 格式，执行精度精确到毫秒。	String	否
UserInfo	用户业务字段。可传入的参数请参考 提交视频审核任务 的 UserInfo 字段。	Dict	否
DataId	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否

版权所有：腾讯云计算（北京）有限责任公司

返回参数说明

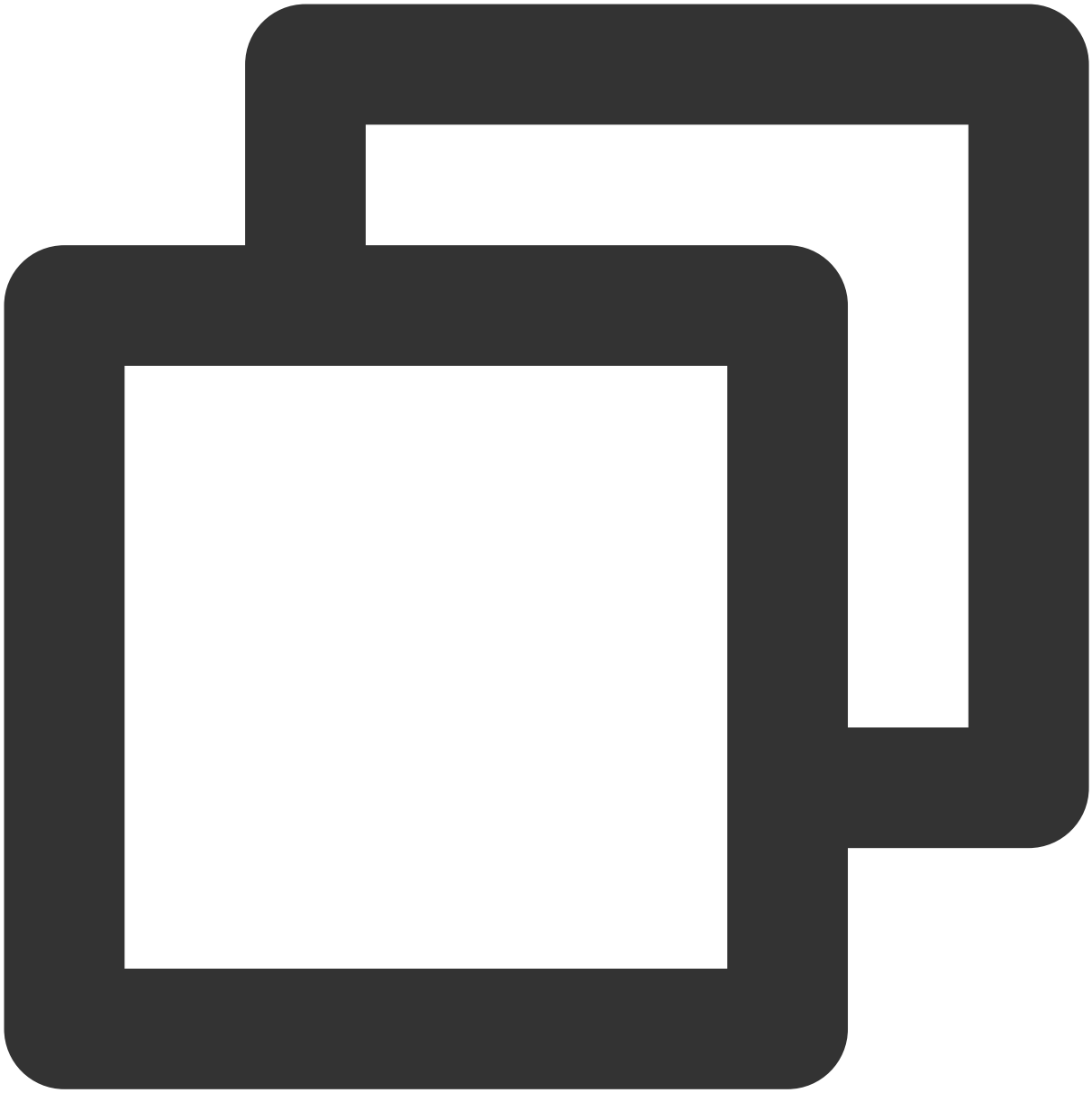
调用 ci_auditing_video_submit 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交视频审核任务 文档。

查询视频审核任务的结果

功能说明

本接口用于主动查询指定的视频审核任务结果。视频审核功能为异步任务方式，您可以通过提交视频审核任务来审核您的视频文件，然后通过查询视频审核任务接口查询审核结果。

示例代码



```
"""测试ci查询审核任务结果的接口"""
# 先创建 cos client
# 查询审核的结果
response = client.ci_auditing_video_query(
    Bucket='bucket',
    JobID='v11122zxxxazzz'
)
print response
```

参数说明

调用 ci_auditing_video_query 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

调用 ci_auditing_video_query 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询视频审核任务结果 文档。

音频审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供音频审核相关的 API 概览以及 SDK 示例代码。

注意：

该功能需要 COS Python SDK v5.1.9.10 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

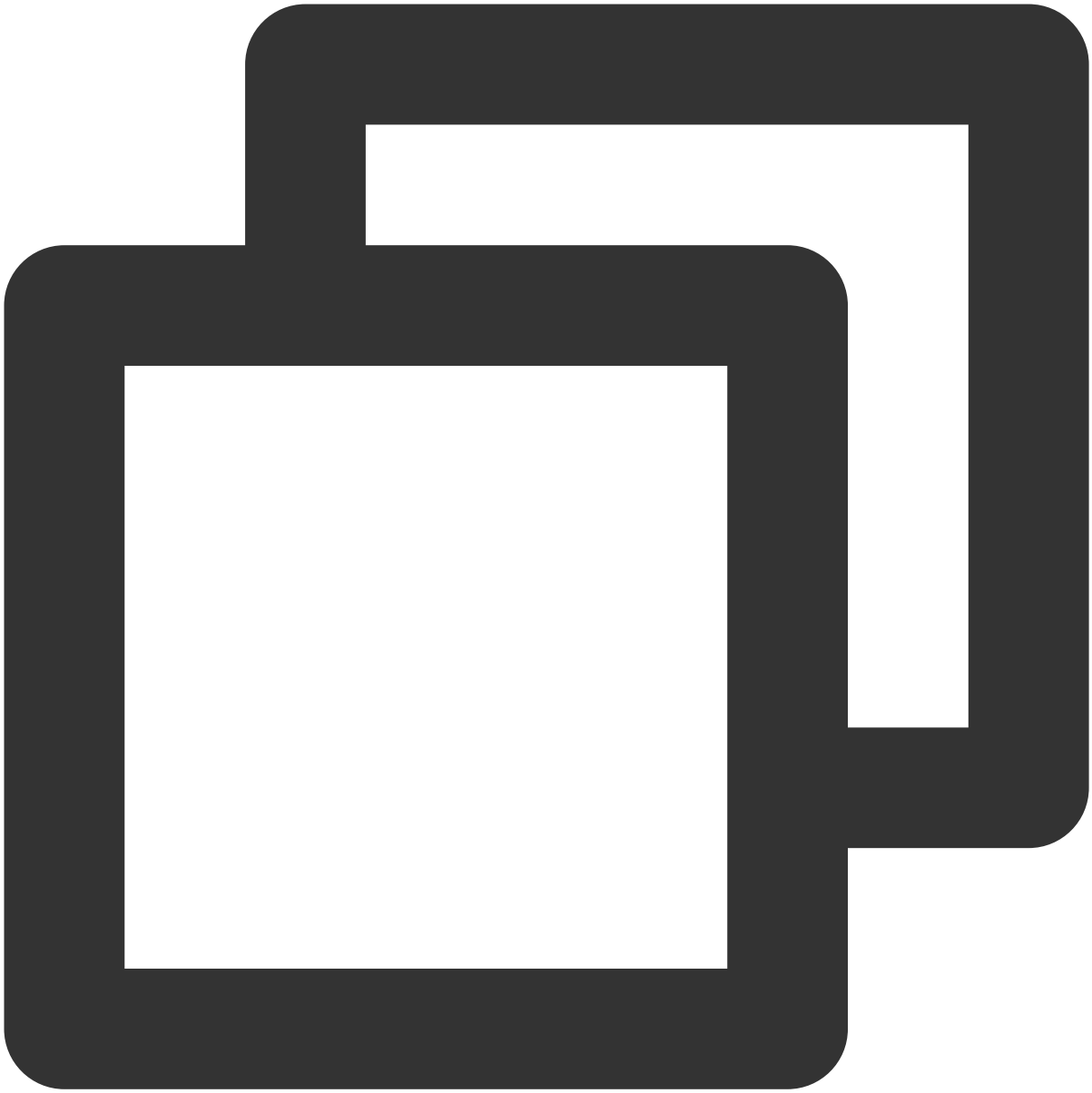
API	操作描述
提交音频审核任务	用于提交一个音频审核任务。
查询音频审核任务结果	用来查询指定的音频审核任务。

提交音频审核任务

功能说明

本接口用于提交一个音频审核任务。音频审核功能为异步任务方式，您可以通过提交音频审核任务来审核您的音频文件，然后通过查询音频审核任务接口查询审核结果。

示例代码



```
"""测试ci音频文件内容审核的接口"""
# 先创建 cos client
# 识别cos上的音频
response= client.ci_auditing_audio_submit(
    Bucket='bucket',
    DetectType=CiDetectType.PORN | CiDetectType.ADS,
    Key='test.mp3'
)
print response
```

参数说明

调用 ci_auditing_audio_submit 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
Key	对象文件名，例如 picture.jpg。	String	是
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	String	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对音频进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Url	支持直接使用非 COS 上的音频 url 进行审核。	String	否
Callback	回调地址，以 http:// 或者 https:// 开头的地址。	String	否
CallbackVersion	回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。	String	否
UserInfo	用户业务字段。可传入的参数请参考 提交音频审核任务 的 UserInfo 字段。	Dict	否
DataId	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否

返回参数说明

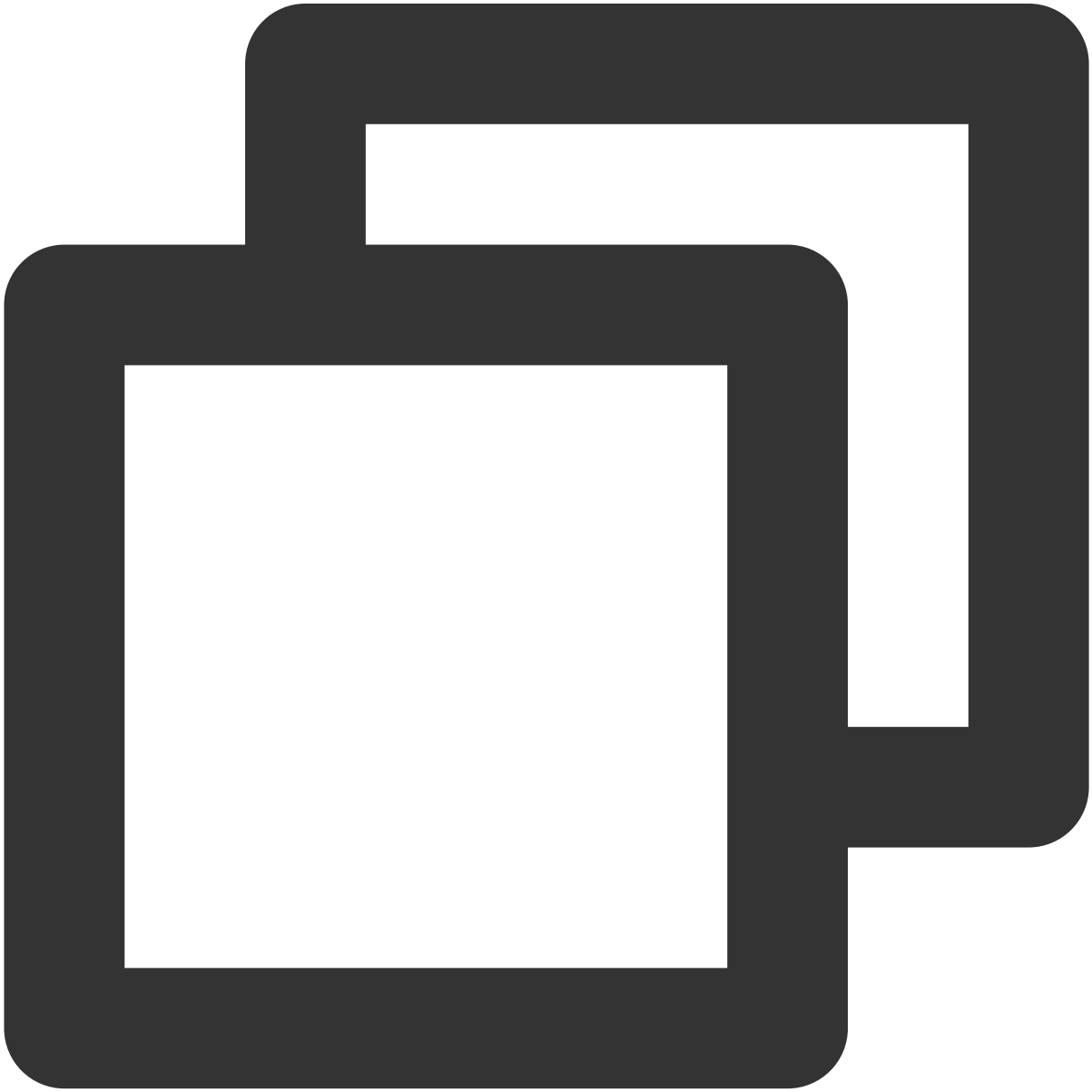
调用 ci_auditing_audio_submit 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交音频审核任务 文档。

查询音频审核任务结果

功能说明

本接口用于主动查询指定的音频审核任务结果。音频审核功能为异步任务方式，您可以通过提交音频审核任务来审核您的音频文件，然后通过查询音频审核任务接口查询审核结果。

示例代码



```
    """测试ci查询审核任务结果的接口"""
    # 先创建 cos client
    # 查询审核的结果
    response = client.ci_auditing_audio_query(
        Bucket='bucket',
        JobID='v11122zxxxazzz'
```

```
    )
    print response
```

参数说明

调用 ci_auditing_audio_query 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

调用 ci_auditing_audio_query 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询音频审核任务结果 文档。

文本审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供文本审核相关的 API 概览以及 SDK 示例代码。

注意：

该功能需要 COS Python SDK v5.1.9.10 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

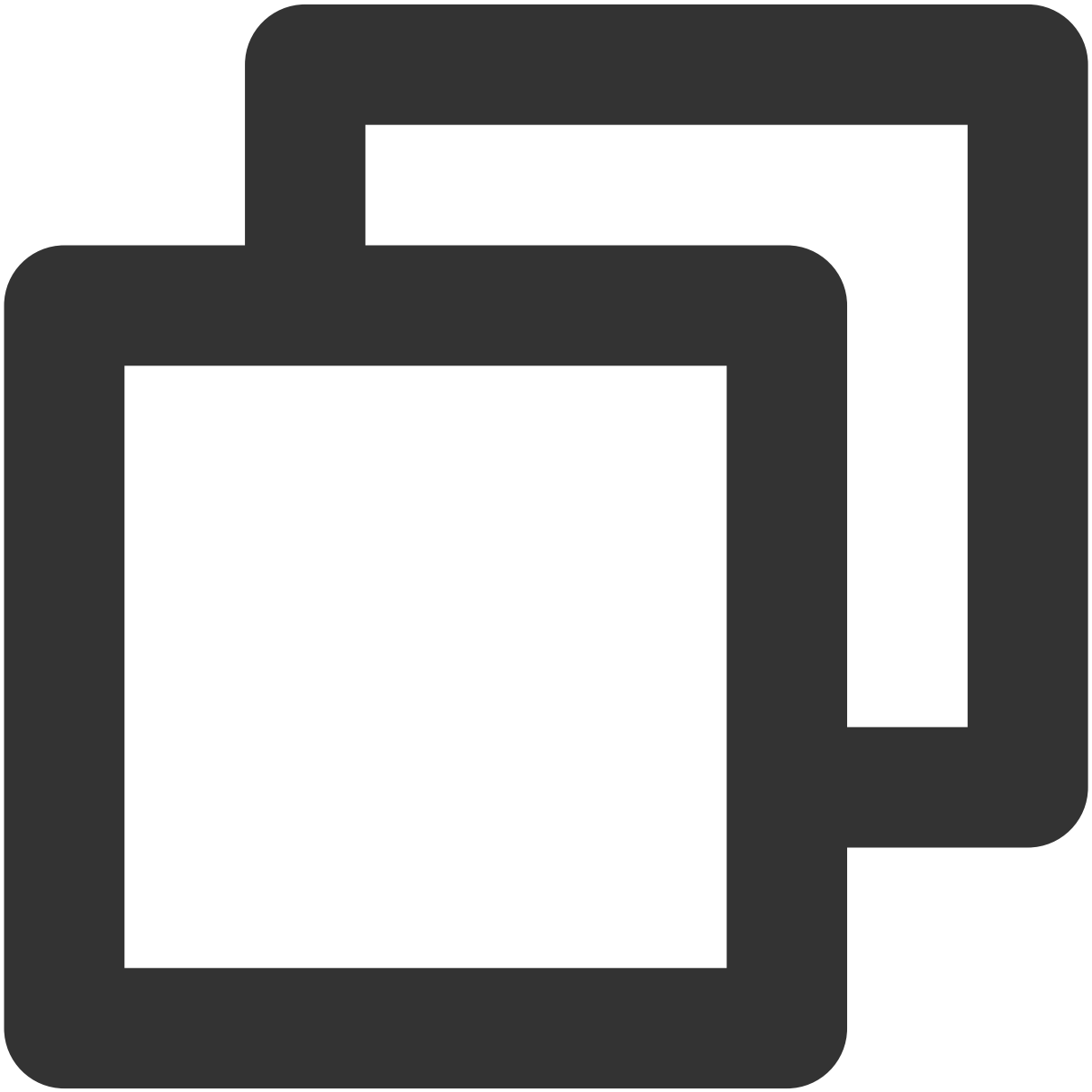
API	操作描述
提交文本审核任务	用于提交一个文本审核任务。
查询文本审核任务结果	用来查询指定的文本审核任务。

提交文本审核任务

功能说明

本接口用于提交一个文本审核任务。文本审核功能为异步任务方式，您可以通过提交文本审核任务来审核您的文本文件，然后通过查询文本审核任务接口查询审核结果。

示例代码



```
"""测试ci文本文件内容审核的接口"""
# 先创建 cos client
# 识别cos上的文本
response= client.ci_auditing_text_submit(
    Bucket='bucket',
    DetectType=CiDetectType.PORN | CiDetectType.ADS,
    Key='test.txt'
)
print response
```

参数说明

调用 ci_auditing_text_submit 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
Key	对象文件名，例如 picture.jpg。	String	否
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	String	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对文本进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Content	当传入的内容为纯文本信息，原文长度不能超过10000个 utf8 编码字符。若超出长度限制，接口将会报错。	String	否
Callback	回调地址，以 http:// 或者 https:// 开头的地址。	String	否
UserInfo	用户业务字段。可传入的参数请参考 提交文本审核任务 的 UserInfo 字段	Dict	否
DataId	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否

返回参数说明

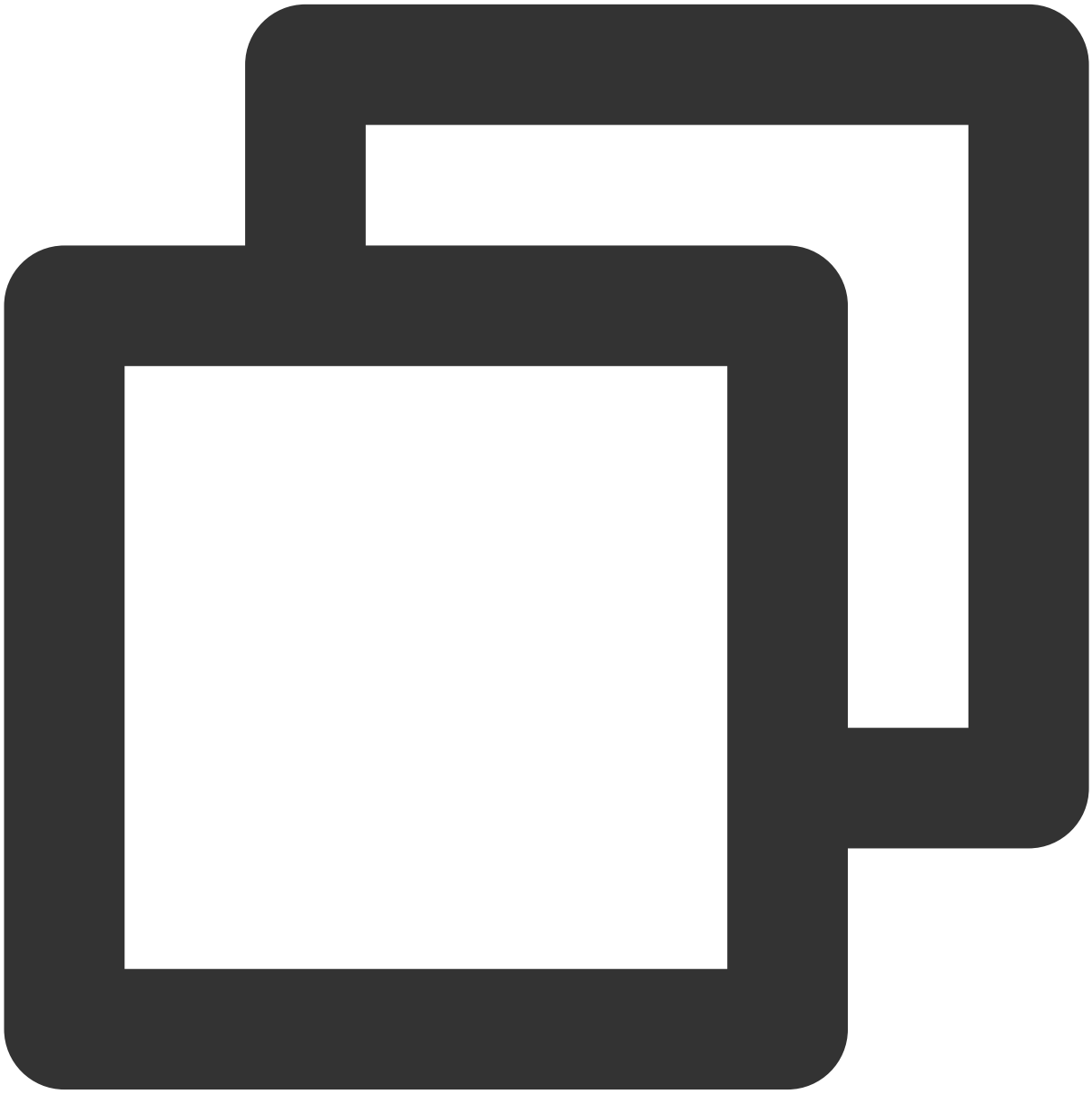
调用 ci_auditing_text_submit 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交文本审核任务 文档。

查询文本审核任务结果

功能说明

本接口用于主动查询指定的文本审核任务结果。文本审核功能为异步任务方式，您可以通过提交文本审核任务来审核您的文本文件，然后通过查询文本审核任务接口查询审核结果。

示例代码



```
"""测试ci查询审核任务结果的接口"""
# 先创建 cos client
# 查询审核的结果
response = client.ci_auditing_text_query(
    Bucket='bucket',
    JobID='v11122zxxxazzz'
)
print response
```

参数说明

调用 ci_auditing_text_query 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

调用 ci_auditing_text_query 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询文本审核任务结果 文档。

文档审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供文档审核相关的 API 概览以及 SDK 示例代码。

注意：

该功能需要 COS Python SDK v5.1.9.10 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

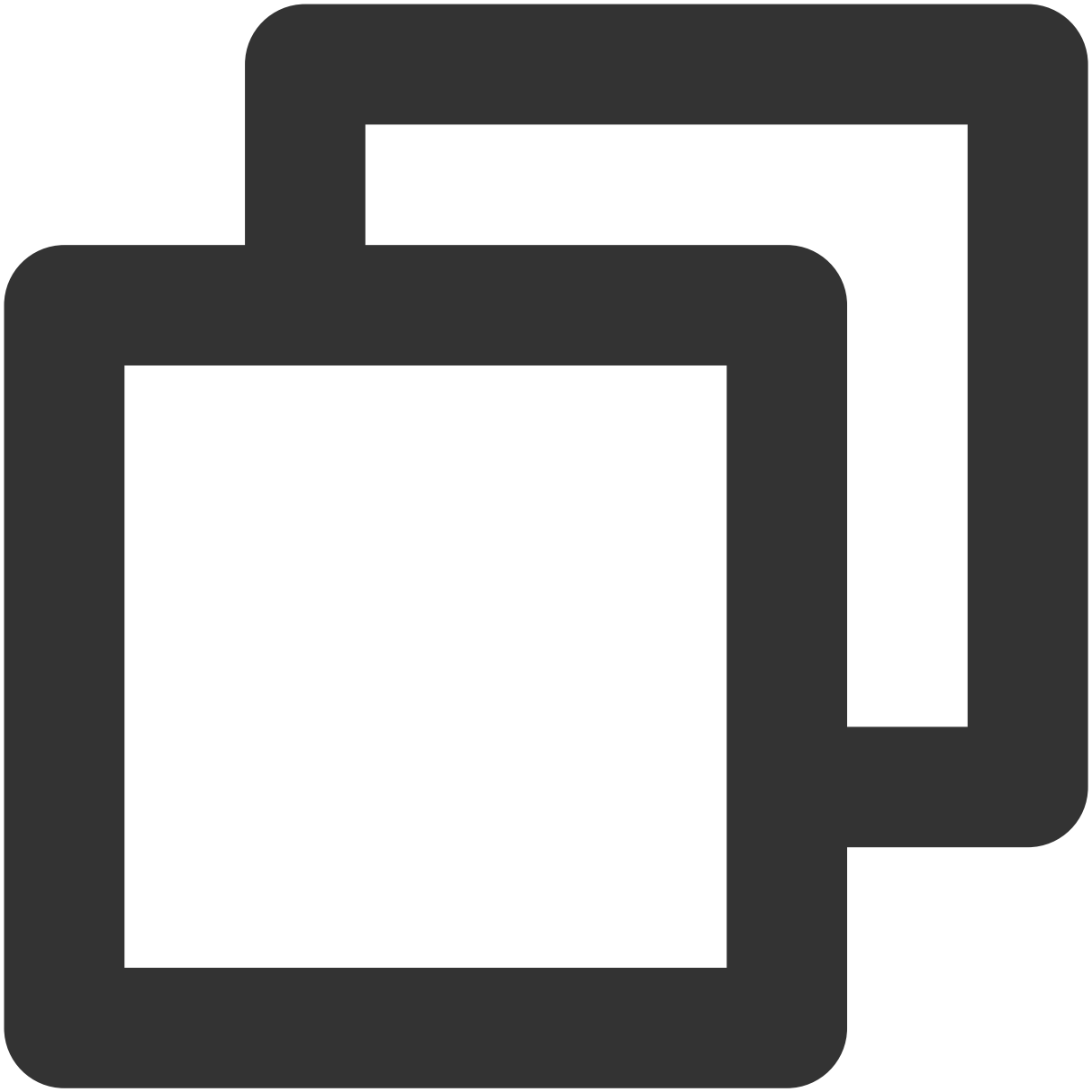
API	操作描述
提交文档审核任务	用于提交一个文档审核任务。
查询文档审核任务结果	用来查询指定的文档审核任务。

提交文档审核任务

功能说明

本接口用于提交一个文档审核任务。文档审核功能为异步任务方式，您可以通过提交文档审核任务来审核您的文档文件，然后通过查询文档审核任务接口查询审核结果。

示例代码



```
"""测试ci文档文件内容审核的接口"""
# 先创建 cos client
# 识别cos上的文档
response = client.ci_auditing_document_submit(
    Bucket=bucket_name,
    DetectType=CiDetectType.PORN | CiDetectType.ADS,
    Url='https://www.test.com/sss.docx',
    Type='docx',
)
print response
```

参数说明

调用 ci_auditing_document_submit 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
Key	对象文件名，例如 picture.jpg。	String	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对文档进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	是
Url	支持直接使用非 COS 上的文档 url 进行审核。	String	否
Type	指定文档文件的类型，如未指定则默认以文件的后缀为类型。如果文件没有后缀，该字段必须指定，否则会审核失败。例如：doc、docx、ppt、pptx 等。	String	否
Callback	回调地址，以 http:// 或者 https:// 开头的地址。	String	否
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	String	否
UserInfo	用户业务字段。可传入的参数请参考 提交文档审核任务 的 UserInfo 字段。	Dict	否
DataId	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否

返回参数说明

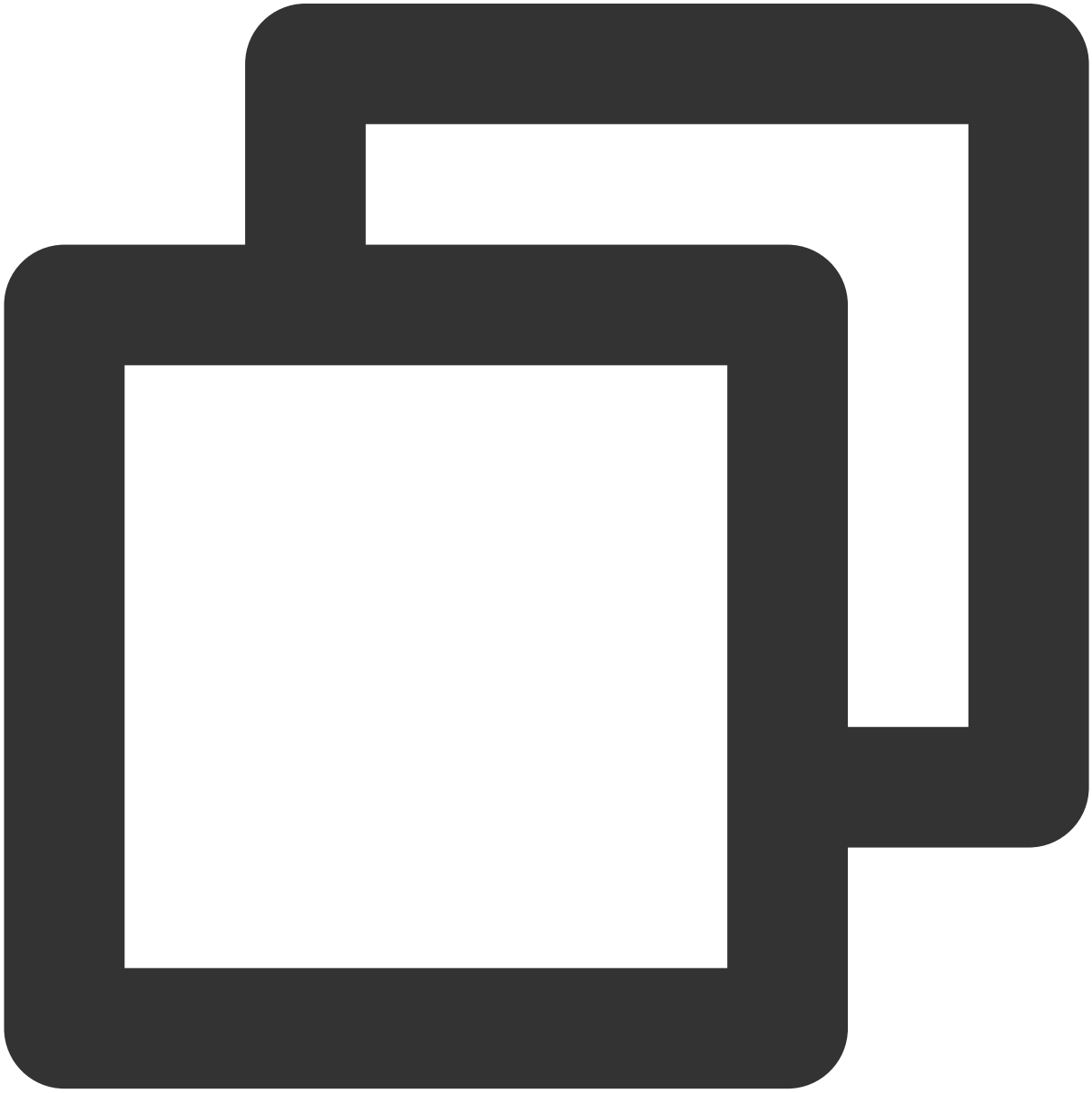
调用 ci_auditing_document_submit 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交文档审核任务 文档。

查询文档审核任务结果

功能说明

本接口用于主动查询指定的文档审核任务结果。文档审核功能为异步任务方式，您可以通过提交文档审核任务来审核您的文档文件，然后通过查询文档审核任务接口查询审核结果。

示例代码



```
    """测试ci查询审核任务结果的接口"""
    # 先创建 cos client
    # 查询审核的结果
    response = client.ci_auditing_document_query(
        Bucket='bucket',
        JobID='v11122zxxxazzz'
```

```
    )
    print response
```

参数说明

调用 ci_auditing_document_query 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

调用 ci_auditing_document_query 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询文档审核任务结果 文档。

网页审核

最近更新时间：2024-03-13 11:32:21

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供网页审核相关的 API 概览以及 SDK 示例代码。

注意：

该功能需要 COS Python SDK v5.1.9.18 及以上版本。旧版本可能存在 bug，使用时建议升级到 最新版本。

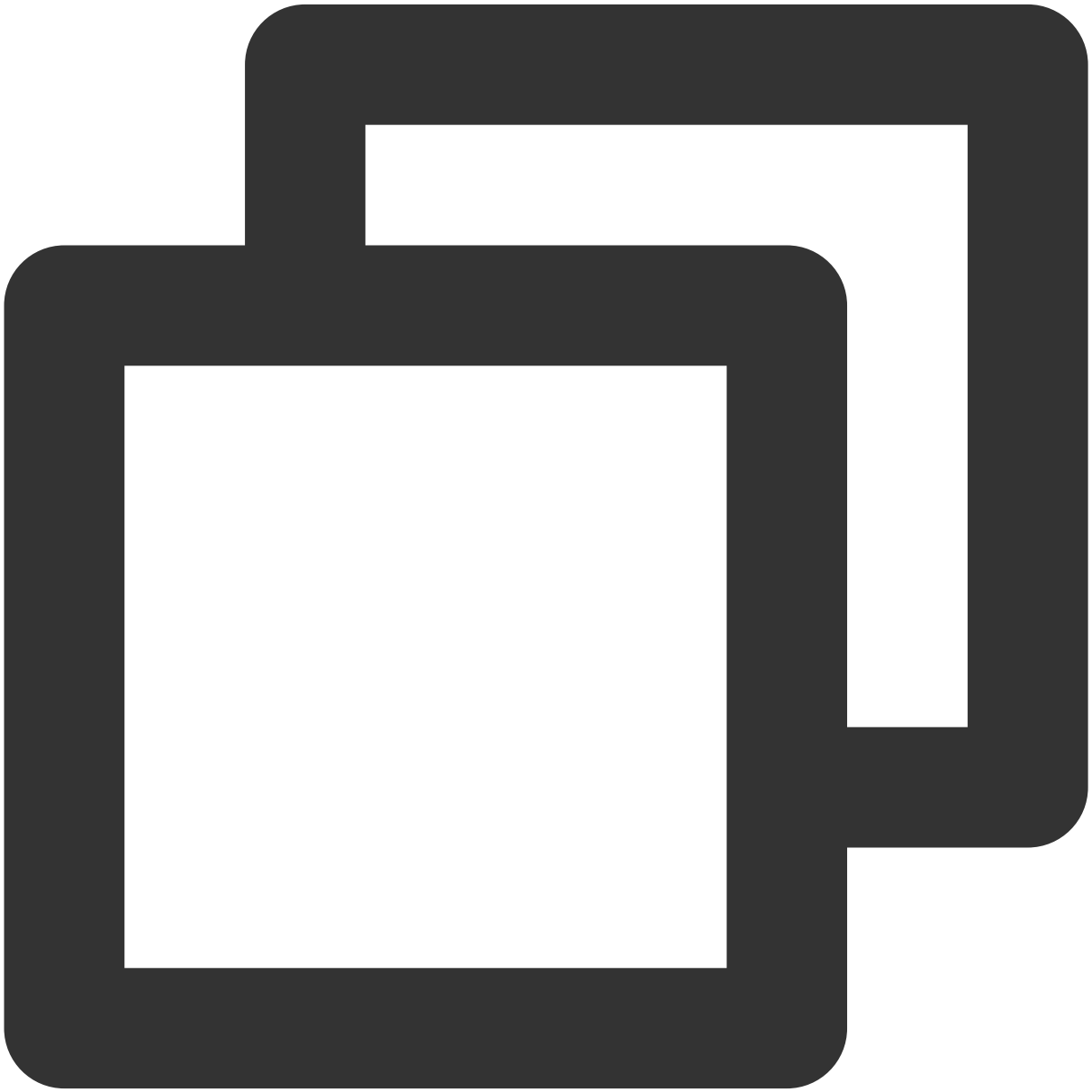
API	操作描述
提交网页审核任务	用于提交一个网页审核任务。
查询网页审核任务结果	用来查询指定的网页审核任务。

提交网页审核任务

功能说明

本接口用于提交一个网页审核任务。网页审核功能为异步任务方式，您可以通过提交网页审核任务来审核您的网页文件，然后通过查询网页审核任务接口查询审核结果。

示例代码



```
"""测试ci网页文件内容审核的接口"""
# 先创建 cos client
# 识别cos上的网页
response= client.ci_auditing_html_submit(
    Bucket='bucket',
    DetectType=CiDetectType.PORN | CiDetectType.ADS,
    Url='http://example.com'
)
print response
```

参数说明

调用 ci_auditing_html_submit 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
Url	网页路径。	String	是
BizType	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，如涉黄、广告、违法违规等，配置指引：设置公共审核策略。您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。BizType 不填写时，将自动使用默认的审核策略。	String	否
DetectType	审核类型，支持 CiDetectType.PORN（涉黄识别）、CiDetectType.ADS（广告识别），用户可选择多种识别类型，例如 `CiDetectType.PORN | CiDetectType.ADS` 表示对网页进行涉黄和广告审核。如果您有更多的审核场景需要，请使用 BizType 参数。	enum	否
Url	支持直接使用非 COS 上的网页 url 进行审核。	String	否
Callback	回调地址，以 http:// 或者 https:// 开头的地址。	String	否
ReturnHighlightHtml	指定是否需要高亮展示网页内的违规文本，查询及回调结果时会根据此参数决定是否返回高亮展示的 html 内容。取值为 true 或者 false，默认为 false。	Bool	否
UserInfo	用户业务字段。可传入的参数请参考 提交网页审核任务 的 UserInfo 字段。	Dict	否
DataId	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否

返回参数说明

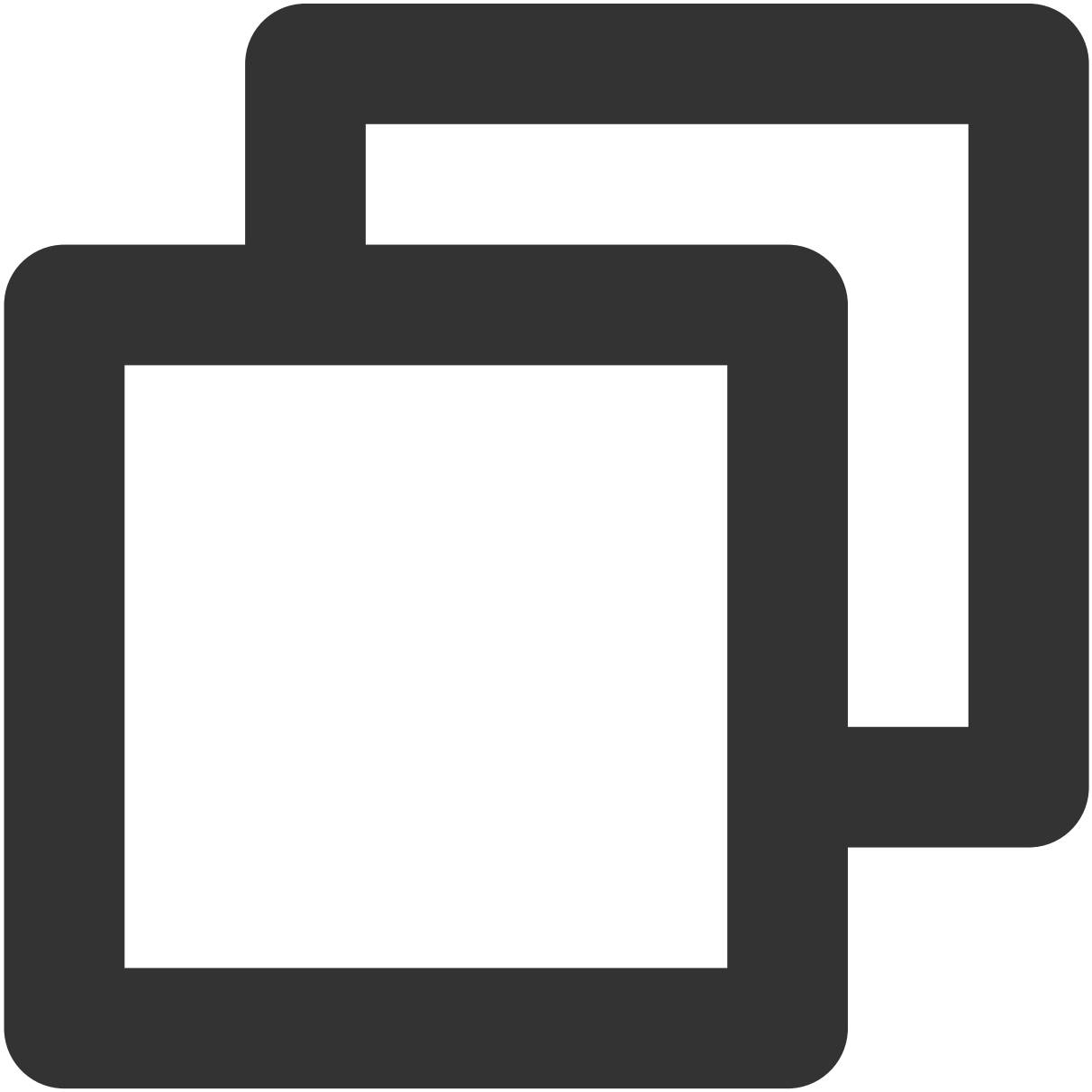
调用 ci_auditing_html_submit 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交网页审核任务 文档。

查询网页审核任务结果

功能说明

本接口用于主动查询指定的网页审核任务结果。网页审核功能为异步任务方式，您可以通过提交网页审核任务来审核您的网页文件，然后通过查询网页审核任务接口查询审核结果。

示例代码



```
    """测试ci查询审核任务结果的接口"""
    # 先创建 cos client
    # 查询审核的结果
    response = client.ci_auditing_html_query(
        Bucket='bucket',
        JobID='v11122zxxxazzz'
```

```
    )
    print response
```

参数说明

调用 ci_auditing_html_query 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

调用 ci_auditing_html_query 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询网页审核任务结果 文档。

内容识别
语音识别
任务接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于语音识别任务接口的 API 概览以及 SDK 示例代码。

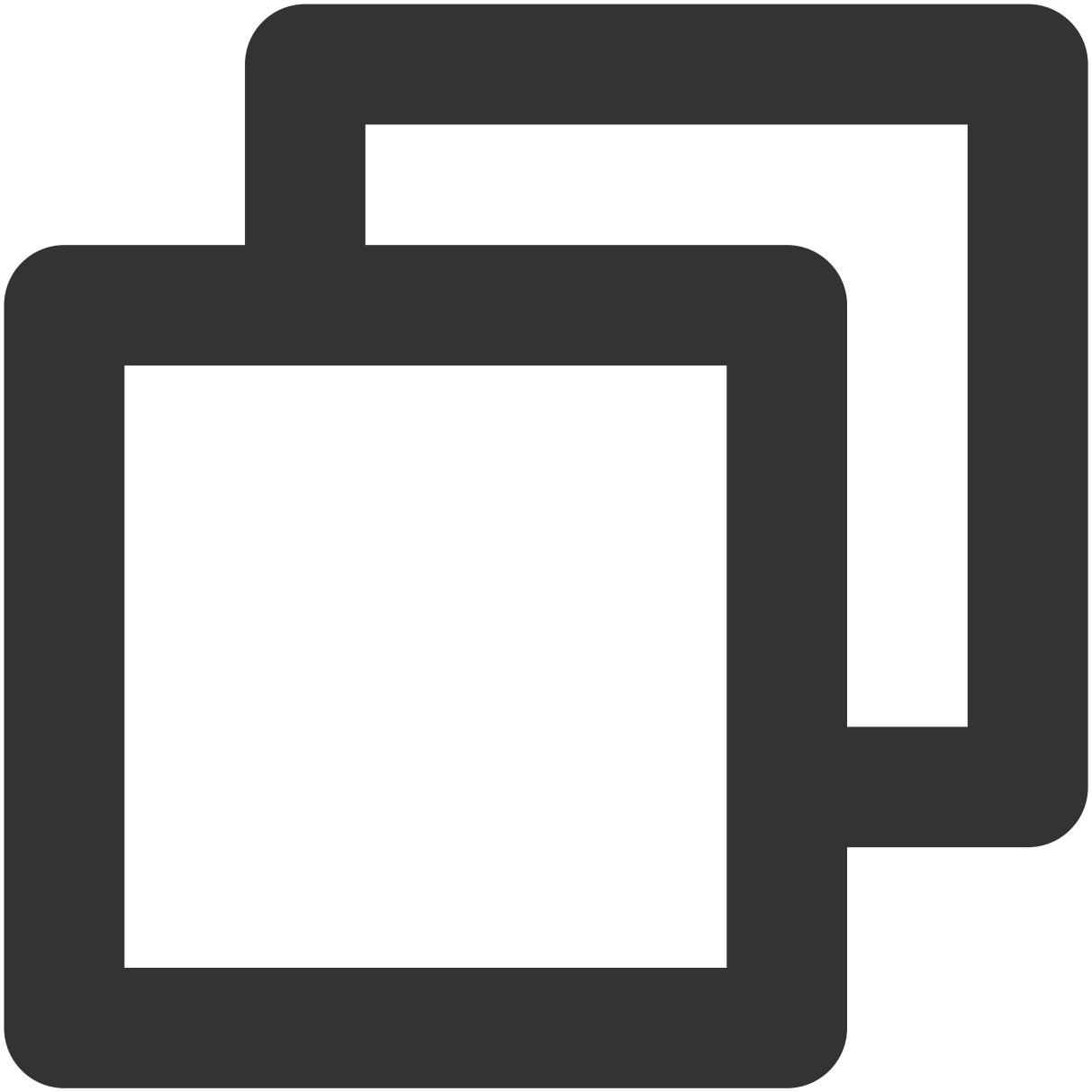
API	操作名	操作描述
提交语音识别任务	提交语音识别任务	用于提交一个语音识别任务
查询指定的语音识别任务	查询指定的语音识别任务	用于查询指定的语音识别任务
批量拉取语音识别任务	拉取符合条件的语音识别任务	用于拉取符合条件的语音识别任务

提交语音识别任务

功能说明

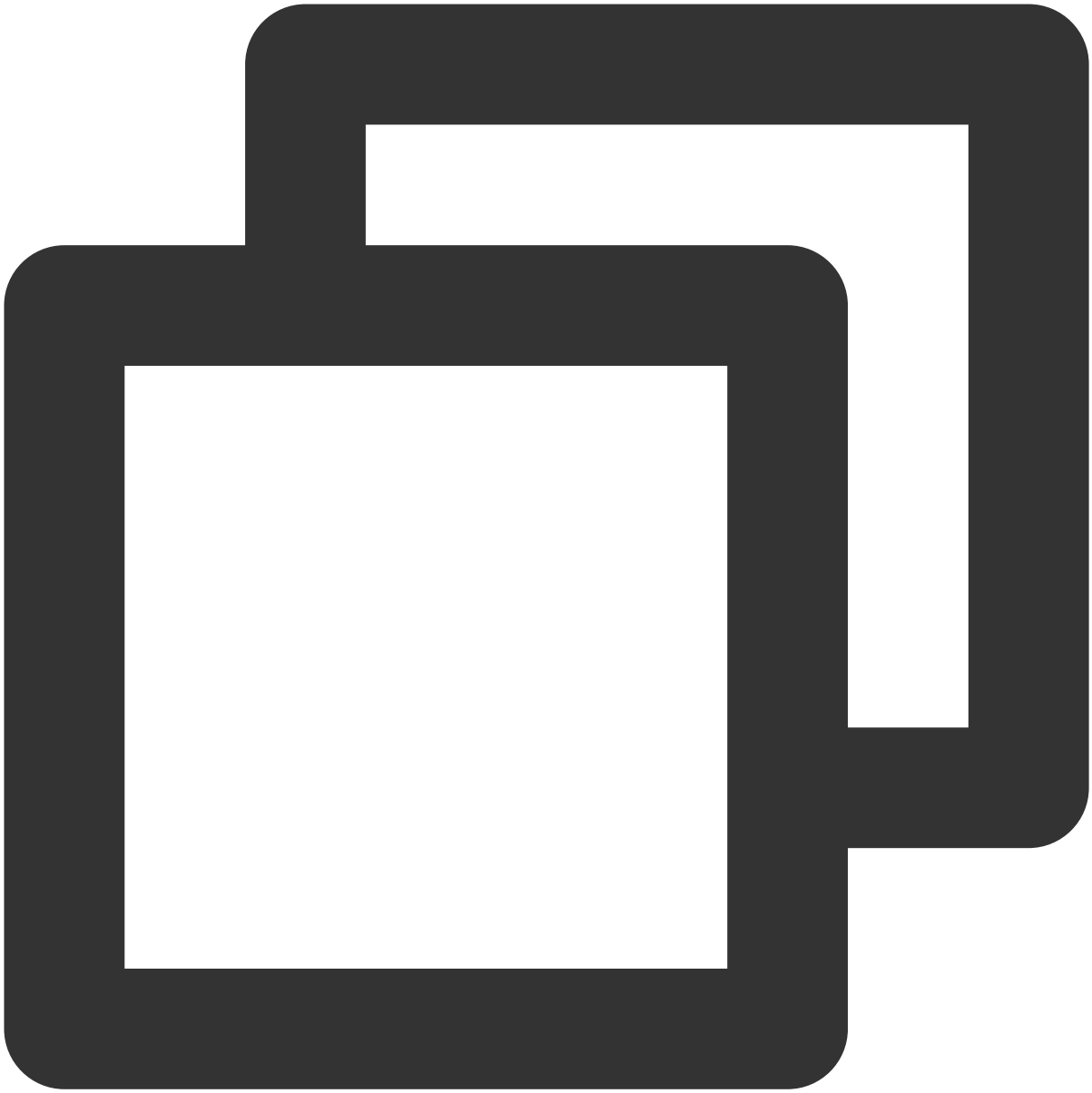
ci_create_asr_job 接口用于提交一个语音识别任务。

方法原型



```
def ci_create_asr_job(self, Bucket, QueueId, InputObject, OutputBucket, OutputRegio
                      SpeechRecognition=None, CallBack=None, CallBackFormat=None, C
```

请求示例



```
def ci_create_asr_jobs():
    # 创建语音识别异步任务(使用模板)
    response = client.ci_create_asr_job(
        Bucket=bucket_name,
        QueueId='s0980xxxxxxxxxxxxxxff12',
        TemplateId='t1ada6f282d29742dxxxxxxxxxxxx',
        InputObject='normal.mp4',
        OutputBucket=bucket_name,
        OutputRegion='ap-chongqing',
        OutputObject='result.txt',
    )
```

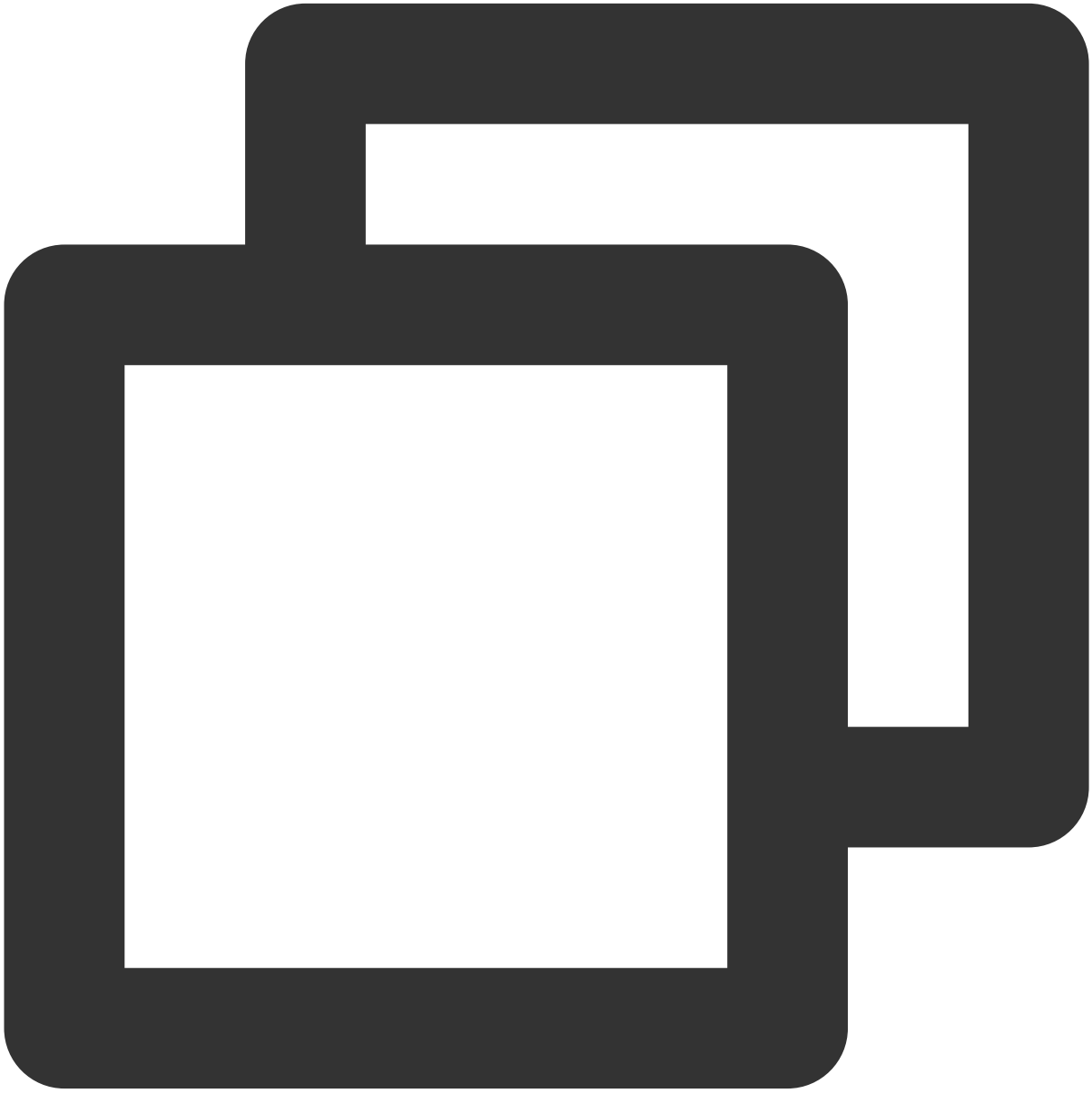
```
    print(response)
    return response

def ci_create_asr_jobs():
    # 创建语音识别异步任务(使用参数)
    body = {
        'EngineModelType': '16k_zh',
        'ChannelNum': '1',
        'ResTextFormat': '1',
    }
    response = client.ci_create_asr_job(
        Bucket=bucket_name,
        QueueId='s0980xxxxxxxxxxxxxxff12',
        InputObject='normal.mp4',
        OutputBucket=bucket_name,
        OutputRegion='ap-chongqing',
        OutputObject='result.txt',
        SpeechRecognition=body
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
QueueId	任务所在的队列 ID	String
InputObject	待操作的文件对象	Container
OutputRegion	存储桶的地域	String
OutputBucket	存储结果的存储桶	String
OutputObject	输出文件路径	String
TemplateId	模板ID	String
SpeechRecognition	语音识别参数，详见 提交语音识别任务请求详情 中的 SpeechRecognition 详情	dict
CallBack	任务回调地址，优先级高于队列的回调地址。设置为 no 时，表示队列的回调地址不产生回调	string
CallBackFormat	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	string

CallBackType	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	string
CallBackMqConfig	任务回调TDMQ配置，当 CallBackType 为 TDMQ 时必填，详见 CallBackMqConfig 详情	dict

返回结果说明



```
{
    'JobsDetail': {
        'Code': 'Success',
        'CreationTime': '2022-09-14T22:31:38+0800',
```

```
        'EndTime': '-',
        'Input': {'BucketId': 'test-12539xxxxx', 'Object': 'normal.mp4', 'Region':
        'JobId': 'sf1398ed0343911eda6d733209028ff12',
        'Message': None,
        'Operation': {
            'JobLevel': '0',
            'Output': {'Bucket': 'test-12539xxxxx', 'Object': 'result.txt', 'Region
            'TemplateId': 't1ada6f282d29742db83244e085exxxxxx',
            'TemplateName': 'name'
        },
        'QueueId': 'p7369ebcf8f93414c8250c966xxxxxxxx',
        'StartTime': '-',
        'State': 'Submitted',
        'Tag': 'SpeechRecognition'
    }
}
```

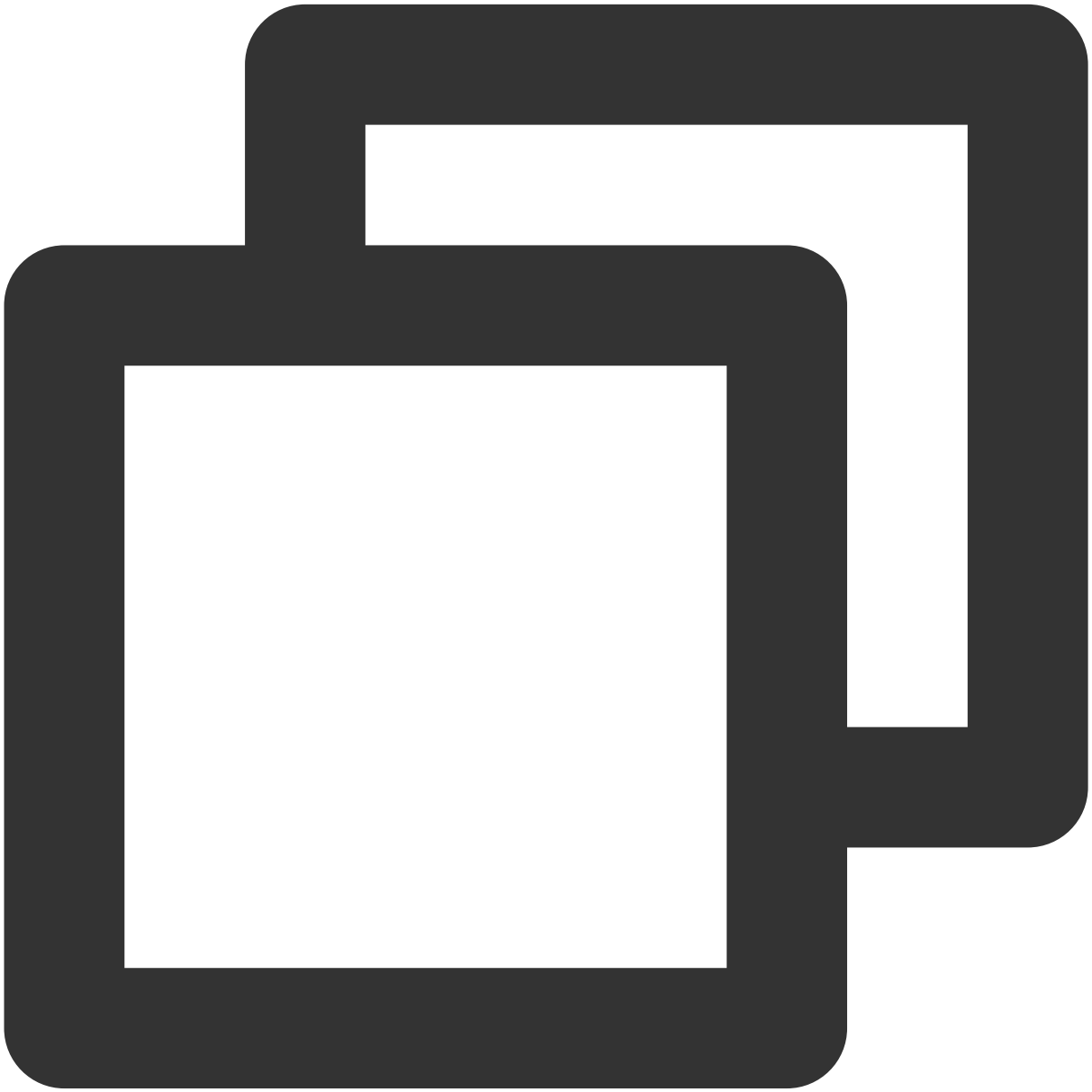
参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：DocProcess	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String
CreationTime	任务的创建时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径	Container
Operation	该任务的规则	Container

查询指定语音识别任务

功能说明

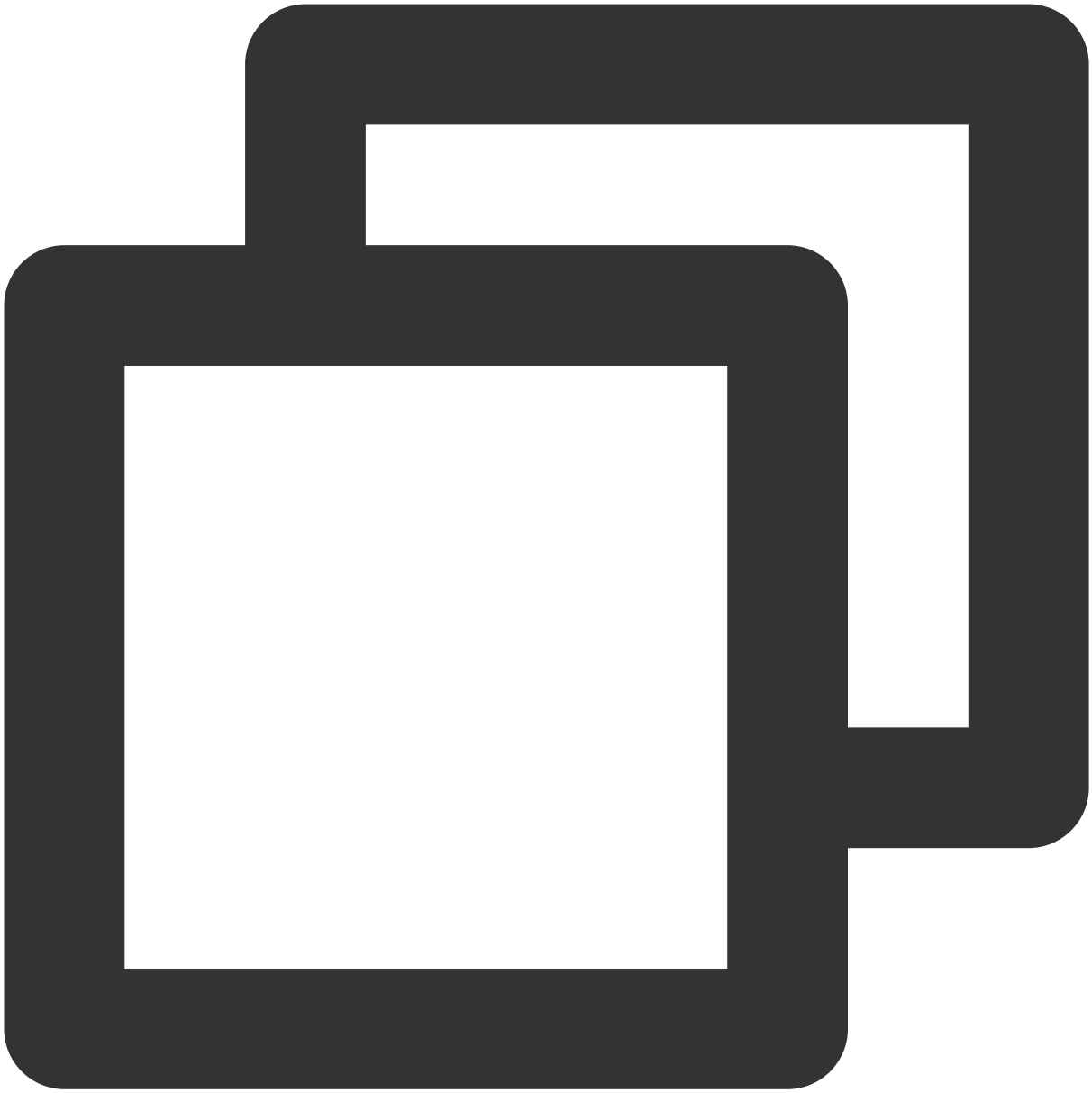
ci_get_asr_job 用于查询指定的语音识别任务。

方法原型



```
def ci_get_asr_job(self, Bucket, JobID, **kwargs):
```

请求示例

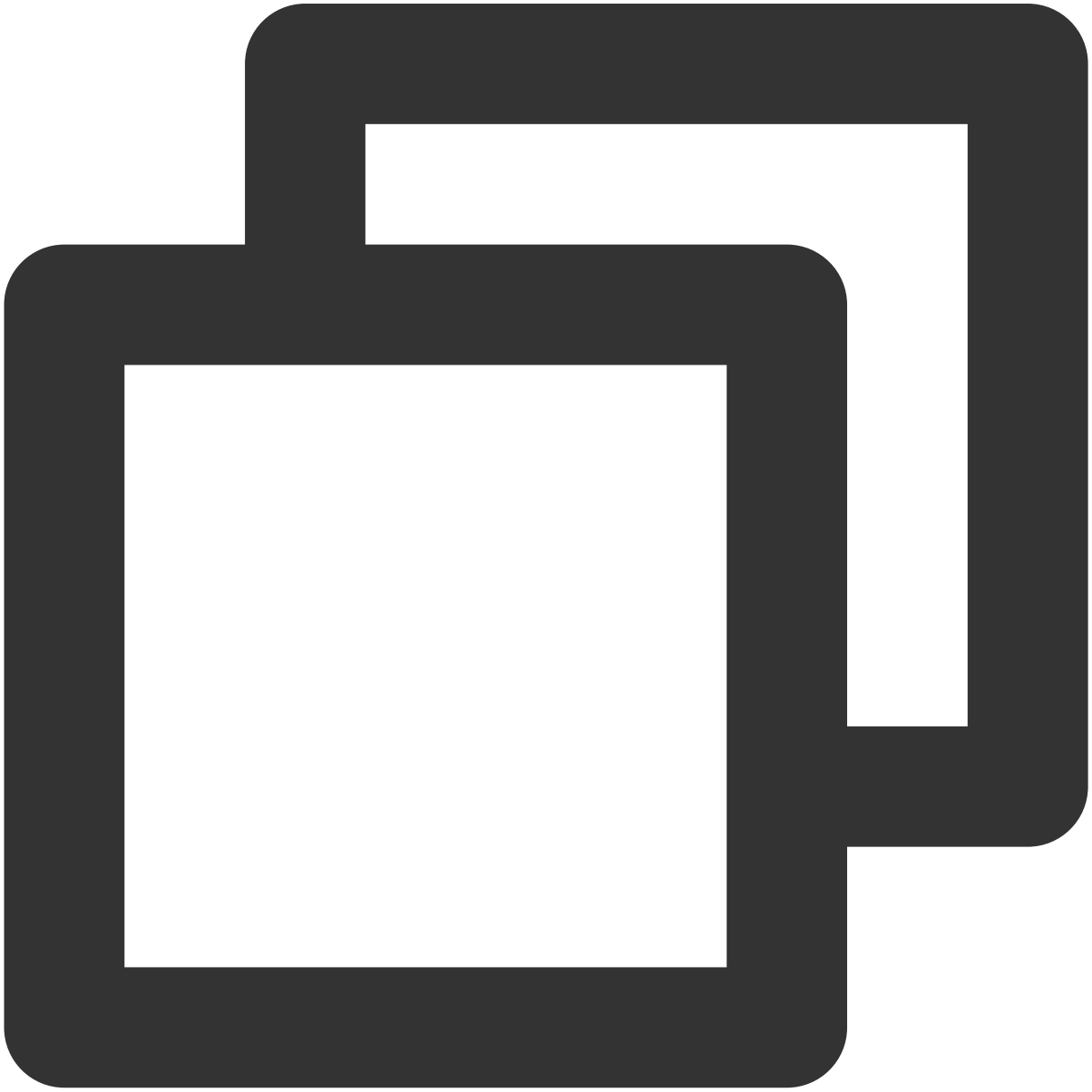


```
def ci_get_asr_jobs():
    # 获取语音识别信息
    response = client.ci_get_asr_job(
        Bucket=bucket_name,
        JobID='s18a9xxxxxxxxxxxxxxxxxxff1aa',
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
JobID	语音识别任务 ID	String

返回结果说明



```
{
    'JobsDetail': [{
```

```
        'Code': 'Success',
        'CreationTime': '2022-09-14T22:36:51+0800',
        'EndTime': '2022-09-14T22:37:01+0800',
        'Input': {'BucketId': 'testpic-1253960454', 'Object': 'gaobai.mp4', 'Region
        'JobId': 'sabfb88d6343a11ed8c375fe3d77ef1d7',
        'Message': None,
        'Operation': {
            'JobLevel': '0',
            'Output': {'Bucket': 'testpic-1253960454', 'Object': 'result.txt', 'Reg
            'SpeechRecognitionResult': {
                'AudioTime': '215.637',
                'ObjectName': 'result.txt',
                'Result': '[0:0.160,1:0.640]  测试测试\\n', 'ResultDetail': None
            },
            'TemplateId': 't1ada6f282d29742db83244e085e920b08',
            'TemplateName': 'name'
        },
        'QueueId': 'p7369ebcf8f93414c8250c9663e1dff5a',
        'StartTime': '2022-09-14T22:36:53+0800',
        'State': 'Success',
        'Tag': 'SpeechRecognition'
    }]
}
```

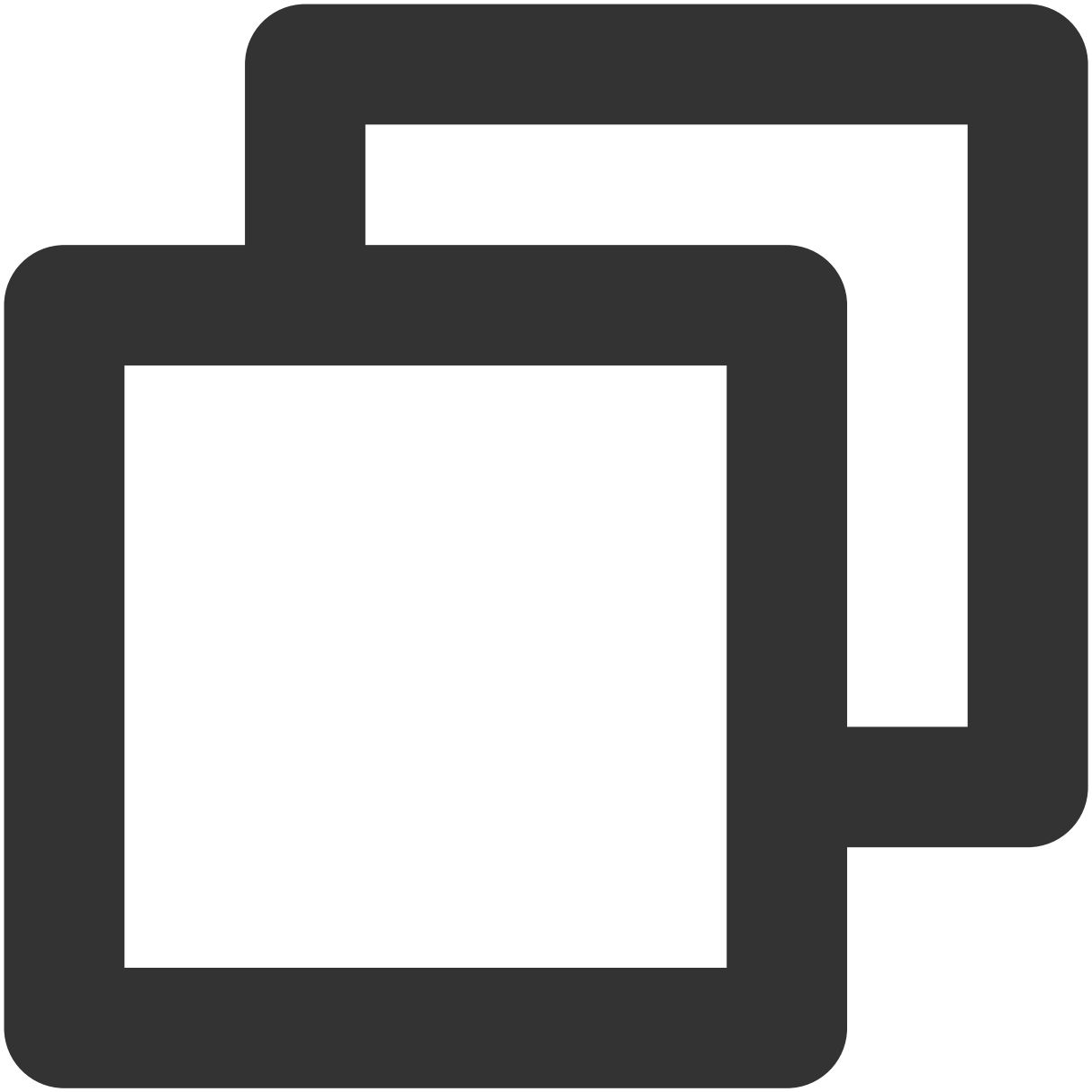
参数名称	描述	类型
JobsDetail	任务的详细信息， 同 CreateSpeechRecognitionJobs 接口的 Response.JobsDetail 节点	Container
NonExistJobIds	查询的 ID 中不存在的任务，所有任务都存在时不返回	String

拉取符合条件的语音识别任务

功能说明

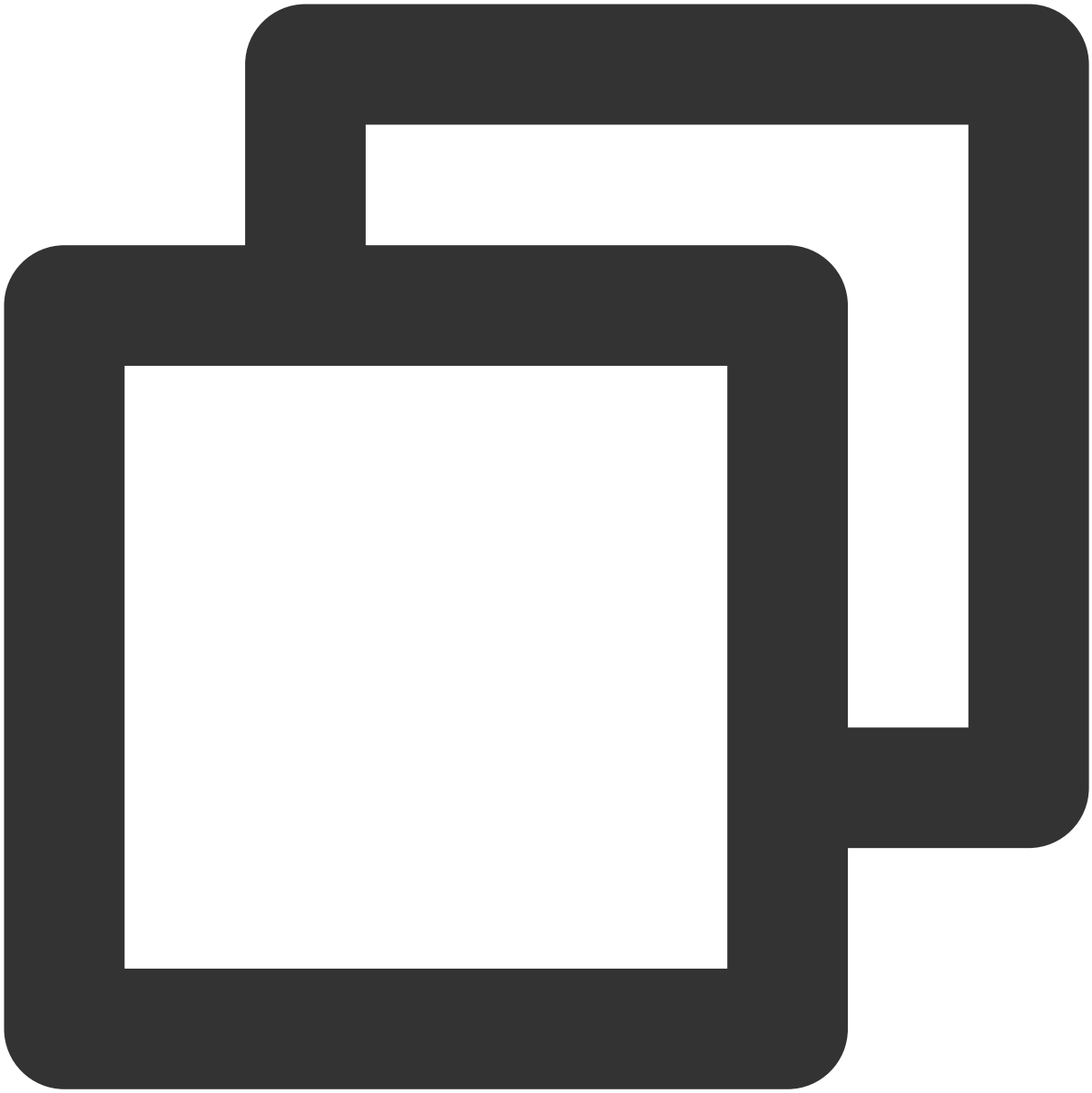
ci_list_doc_jobs 用于拉取符合条件的语音识别任务。

方法原型



```
def ci_list_asr_jobs(self, Bucket, QueueId, StartCreationTime=None, EndCreationTime
```

请求示例



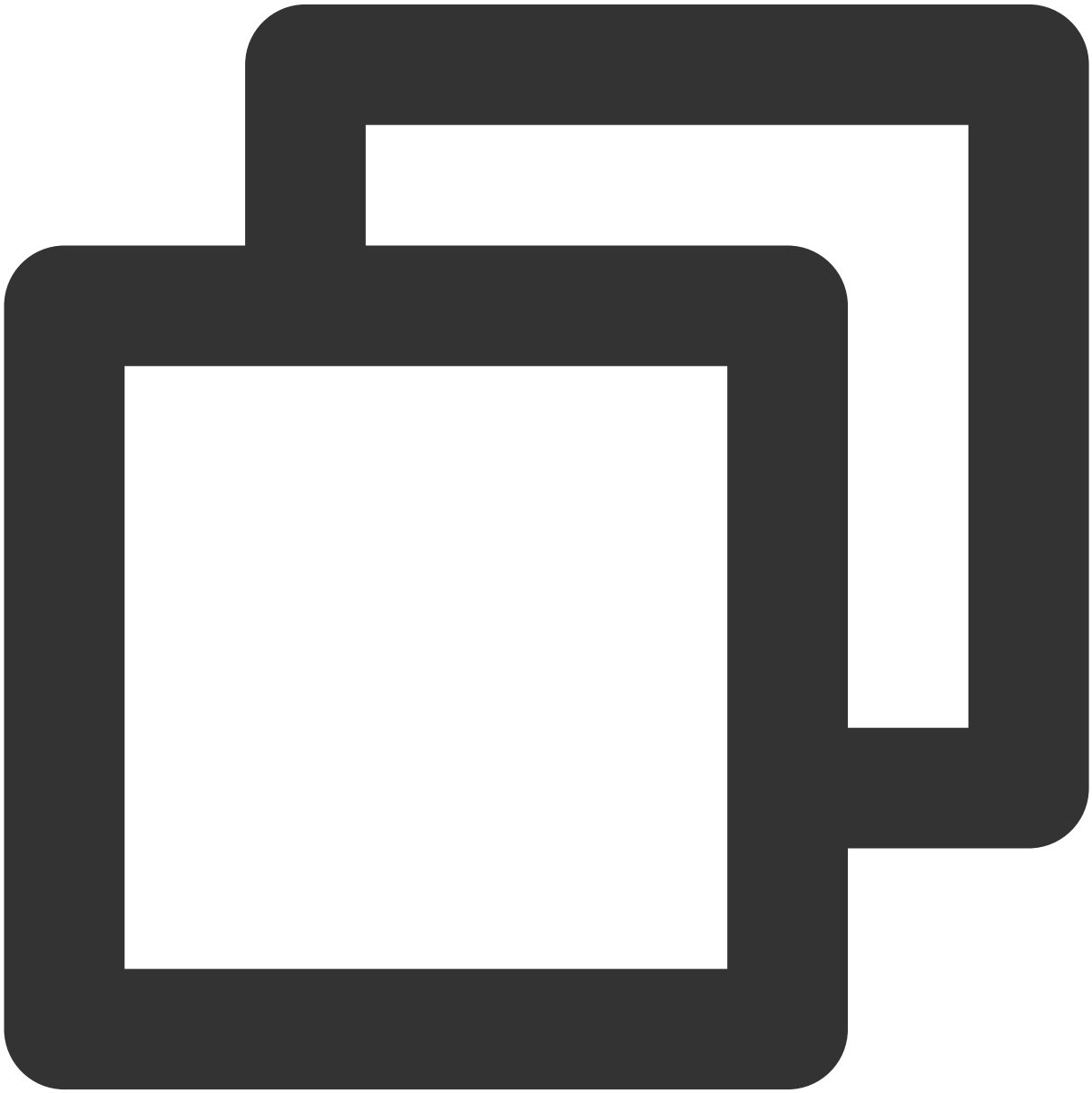
```
def ci_list_asr_jobs():
    # 获取语音识别任务信息列表
    response = client.ci_list_asr_jobs(
        Bucket=bucket_name,
        QueueId='p4bdxxxxxxxxxxxxxxxxxxx57f1',
        Size=10,
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	任务所在存储桶	String
QueueId	拉取该队列 ID 下的任务	String
OrderByTime	Desc 或者 Asc。默认为 Desc	String
NextToken	请求的上下文，用于翻页。上次返回的值	String
Size	拉取的最大任务数。默认为10，最大值为100	Int
States	拉取该状态的任务，以 `,` 分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel。默认为 All	String
StartCreationTime	拉取创建时间大于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String
EndCreationTime	拉取创建时间小于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String

返回结果说明

版权所有：腾讯云计算（北京）有限责任公司



```
{
    'JobsDetail': [{
        'Code': 'Success',
        'CreationTime': '2022-09-14T22:36:51+0800',
        'EndTime': '2022-09-14T22:37:01+0800',
        'Input': {'BucketId': 'test-125396xxxx', 'Object': 'gaobai.mp4', 'Region':
        'JobId': 'sabfb88d6343a11ed8c375fe3dxxxxxxx',
        'Message': None,
        'Operation': {
            'JobLevel': '0',
            'Output': {'Bucket': 'test-125396xxxx', 'Object': 'result.txt', 'Region
```

```
            'SpeechRecognitionResult': {
                'AudioTime': '215.637',
                'ObjectName': 'result.txt',
                'Result': '[0:0.160,1:0.640]  测试测试。\\n',
                'ResultDetail': None
            },
            'TemplateId': 't1ada6f282d29742db83244e085xxxxxx',
            'TemplateName': 'name'
        },
        'QueueId': 'p7369ebcf8f93414c8250c966xxxxxxx',
        'StartTime': '2022-09-14T22:36:53+0800',
        'State': 'Success',
        'Tag': 'SpeechRecognition'
    }],
    'NextToken': '15966'
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息，同 CreateSpeechRecognitionJobs 接口中的 Response.JobsDetail 节点	Container
NextToken	翻页的上下文 Token	String

语音 bucket 接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于查询语音识别开通状态的 API 概览以及 SDK 示例代码。

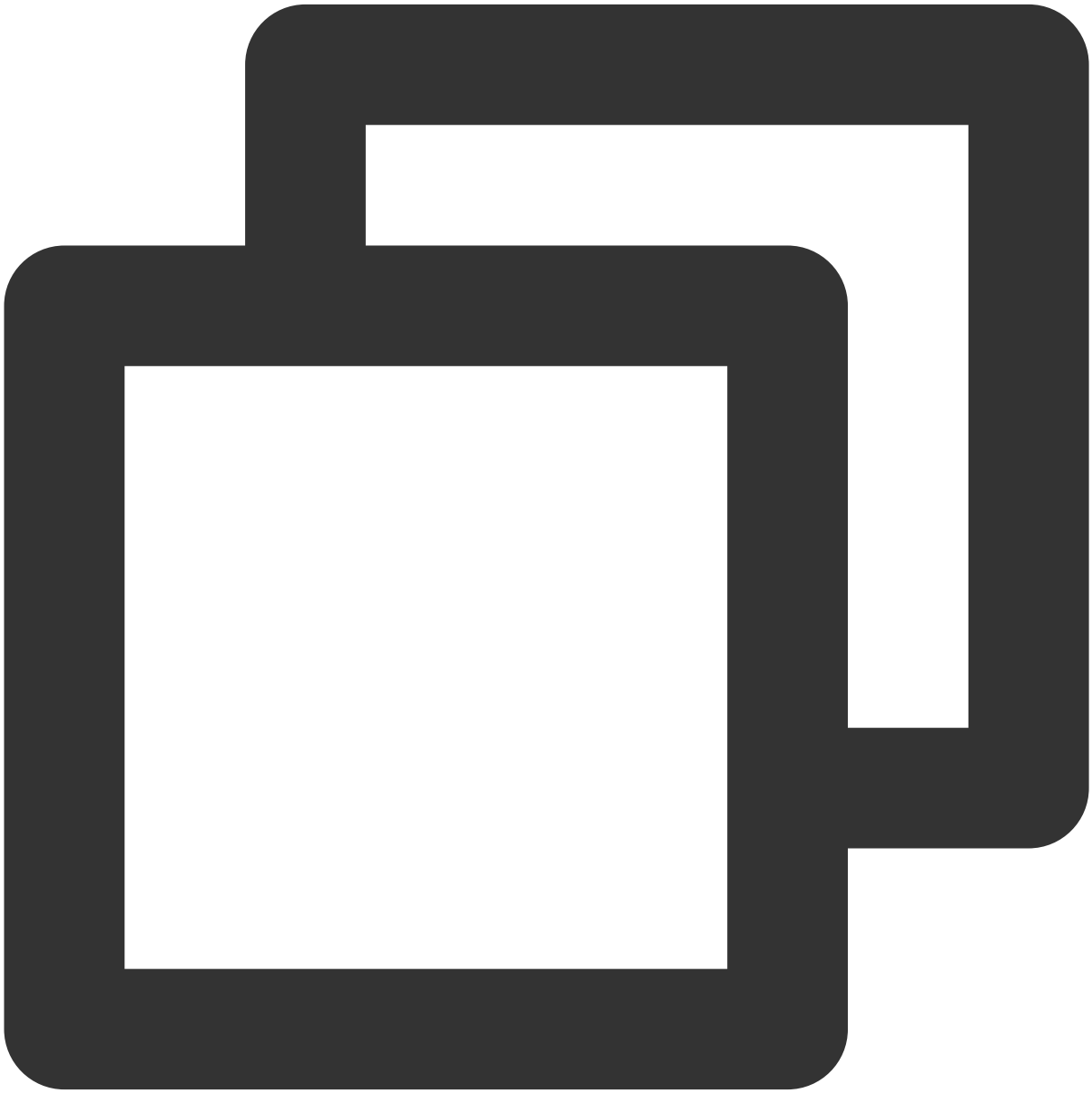
API	说明
查询语音识别开通状态	用于查询存储桶是否已开通语音识别功能

查询语音识别开通状态

功能说明

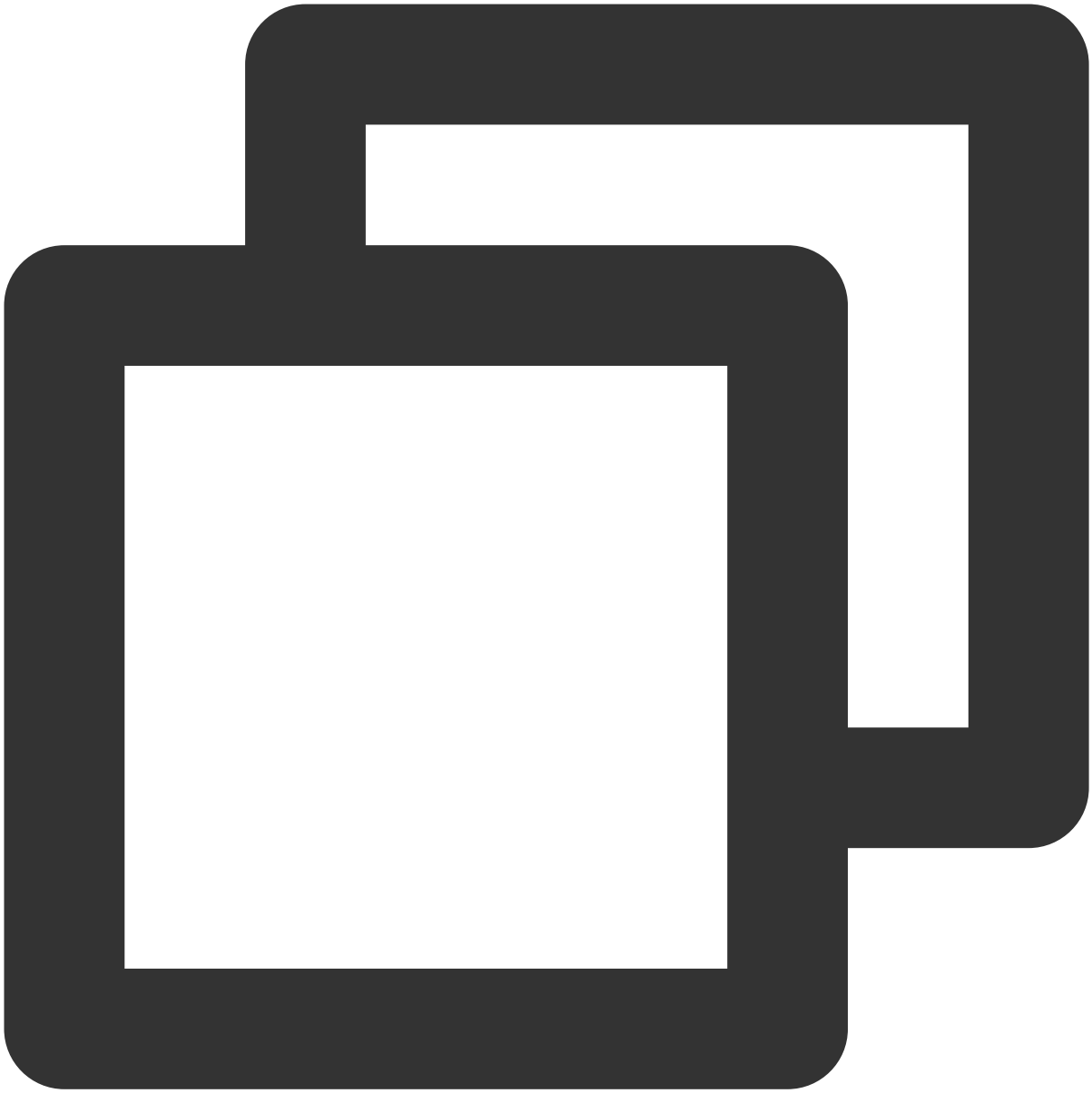
ci_get_asr_bucket 接口用于查询存储桶是否已开通语音识别功能。

方法原型



```
def ci_get_asr_bucket(self, Regions='', BucketName='', BucketNames='', PageNumber='
```

请求示例

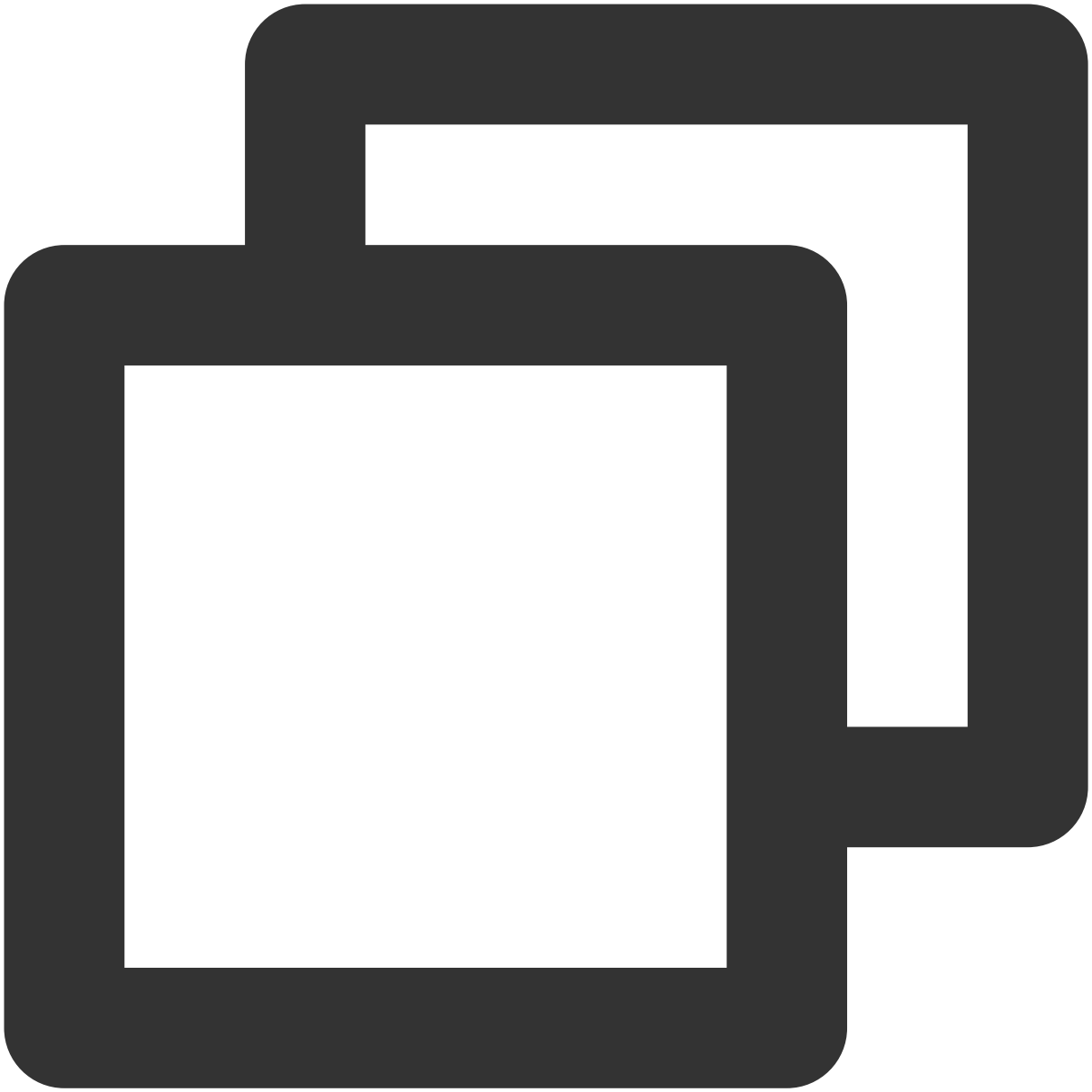


```
def ci_get_asr_bucket():
    # 查询语音识别开通状态
    response = client.ci_get_asr_bucket(
        Regions=region,
        # BucketName='demo',
        BucketNames=bucket_name,
        PageSize=1,
        PageNumber=1
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	string
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string
BucketName	存储桶名称前缀，前缀搜索	string
PageNumber	第几页	int
pageSize	每页个数	int

返回结果说明



```
{
    'TotalCount': '1',
    'RequestId': 'NjMyMWE2ZjZfZWM0YTYyNjRxxxxxxxxxxxx',
    'PageNumber': '1',
    'PageSize': '1',
    'AsrBucketList': [{
        'Name': 'test-125xxxxxxx',
        'CreateTime': '2022-08-11T00:00:00+0800',
        'Region': 'ap-chongqing',
        'AliasBucketId': None,
        'BucketId': 'test-125xxxxxxxx'
```

```
        }]
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	string
TotalCount	语音识别 Bucket 总数	int
PageNumber	当前页数，同请求中的 pageNumber	int
PageSize	每页个数，同请求中的 pageSize	int
AsrBucketList	语音识别 Bucket 列表	Container
BucketId	存储桶 ID	string
Name	存储桶名称，同 BucketId	string
Region	所在的地域	string
CreateTime	创建时间	string
AliasBucketId	存储桶别名	string

版权所有：腾讯云计算（北京）有限责任公司

队列接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于语音识别队列接口的 API 概览以及 SDK 示例代码。

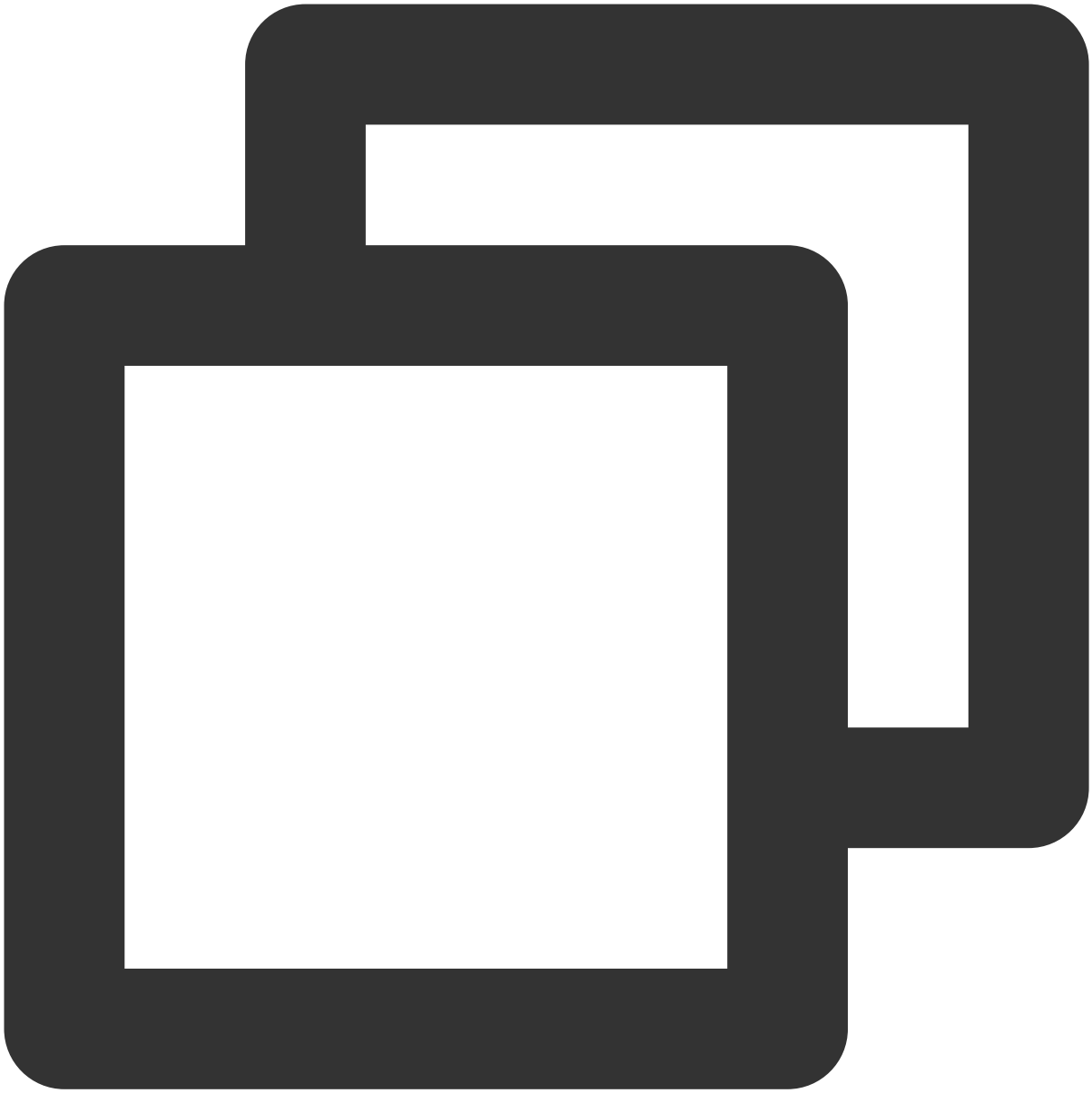
API	操作名	操作描述
查询语音识别队列	查询语音识别队列	查询语音识别队列信息
更新语音识别队列	更新语音识别队列	更新语音识别队列信息

查询语音识别队列

功能说明

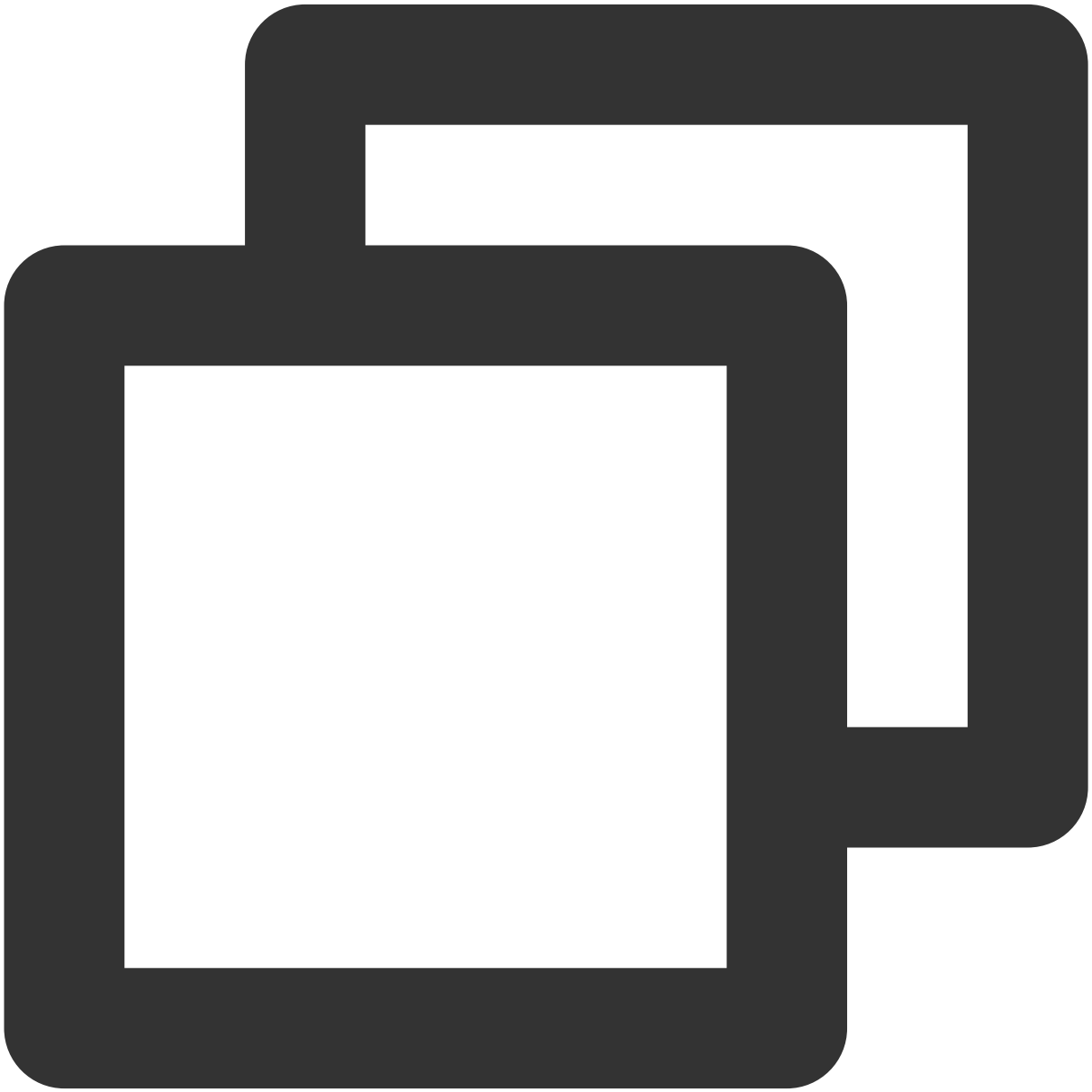
ci_get_asr_queue 接口用于查询语音识别队列。

方法原型



```
def ci_get_asr_queue(self, Bucket, State='All', QueueIds='', PageNumber='', PageSiz
```

请求示例

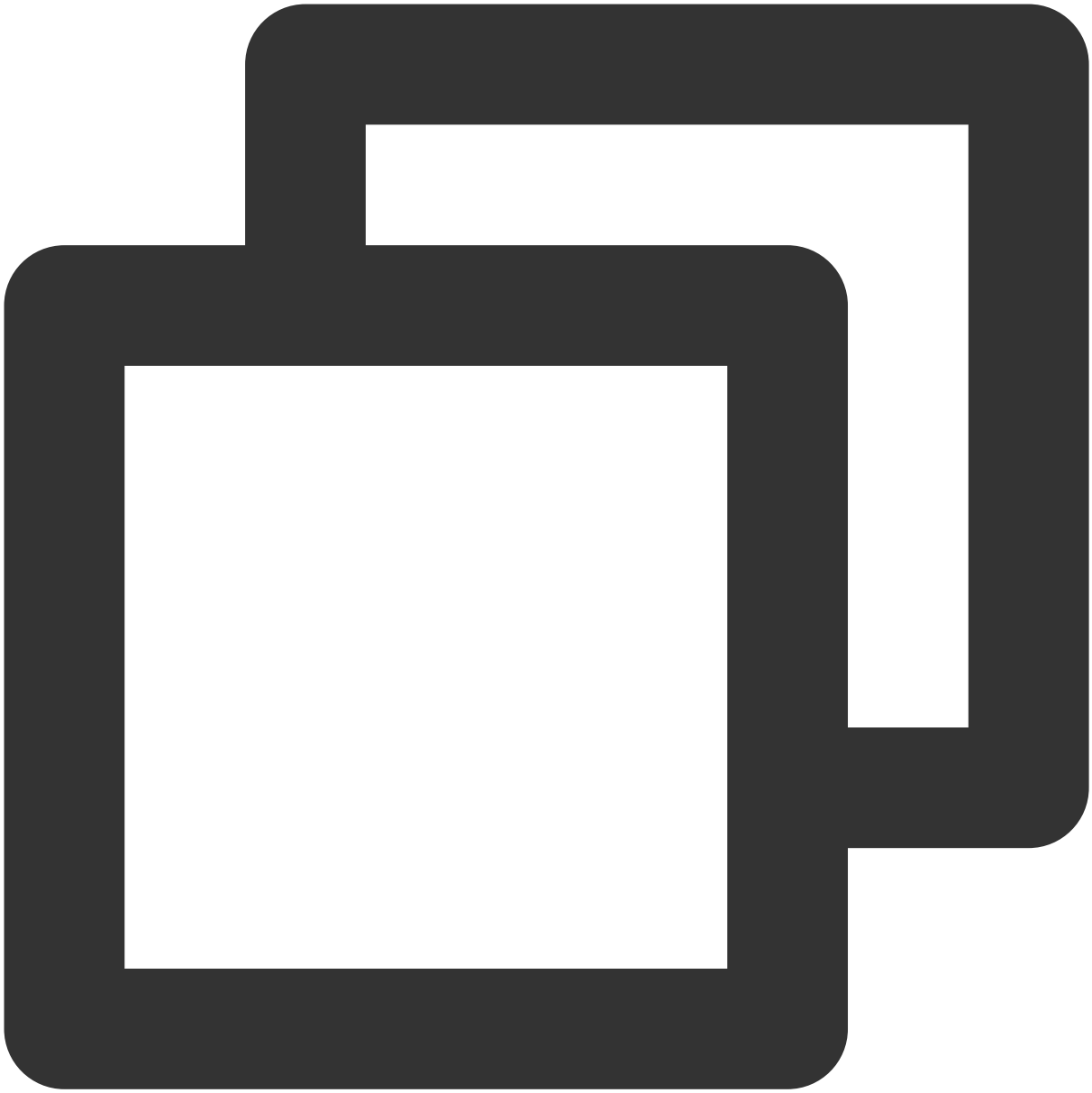


```
def ci_get_asr_queue():
    # 查询语音识别队列信息
    response = client.ci_get_asr_queue(
        Bucket=bucket_name,
        PageNumber=1,
        PageSize=1,
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	队列所在存储桶	String
QueueIds	队列 ID，以“,”符号分割字符串	String
State	1. Active 表示队列内的作业会被文档预览服务调度执行 2. Paused 表示队列暂停，作业不再会被文档预览服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于执行中的任务将继续执行，不受影响	String
PageNumber	第几页	int
PageSize	每页个数	int

返回结果说明



```
{
    'TotalCount': '1',
    'RequestId': 'NjMyMWExY2RfMTIwNjxxxxxxxxxxxxxxxxxx',
    'PageNumber': '1',
    'PageSize': '1',
    'QueueList': [{
        'QueueId': 'p4bdf22746bxxxxxxxxxxxxxxxxxxxxx',
        'Name': 'speech-queue',
        'State': 'Active',
        'NotifyConfig': {
            'Url': 'http://www.demo.callback.com',
```

```
            'Event': 'TaskFinish',
            'Type': 'Url',
            'State': 'On',
            'ResultFormat': 'JSON',
            'MqMode': None,
            'MqName': None,
            'MqRegion': None
        },
        'MaxSize': '10000',
        'MaxConcurrent': '10',
        'CreateTime': '2022-08-11T14:42:01+0800',
        'UpdateTime': '2022-09-14T11:18:08+0800',
        'BucketId': 'test-125xxxxxxxxx',
        'Category': 'Speeching'
    }]
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	String
TotalCount	队列总数	Int
PageNumber	当前页数，同请求中的 pageNumber	Int
PageSize	每页个数，同请求中的 pageSize	Int
QueueList	队列数组	Container
NonExistPIDs	不存在的队列 ID 列表	String 数组
QueueId	队列 ID	String
Name	队列名字	String
State	当前状态，Active 或者 Paused	String
NotifyConfig	回调配置	Container
MaxSize	队列最大长度	Int
MaxConcurrent	当前队列最大并行执行的任务数	Int
UpdateTime	更新时间	String
CreateTime	创建时间	String
Url	回调地址	String

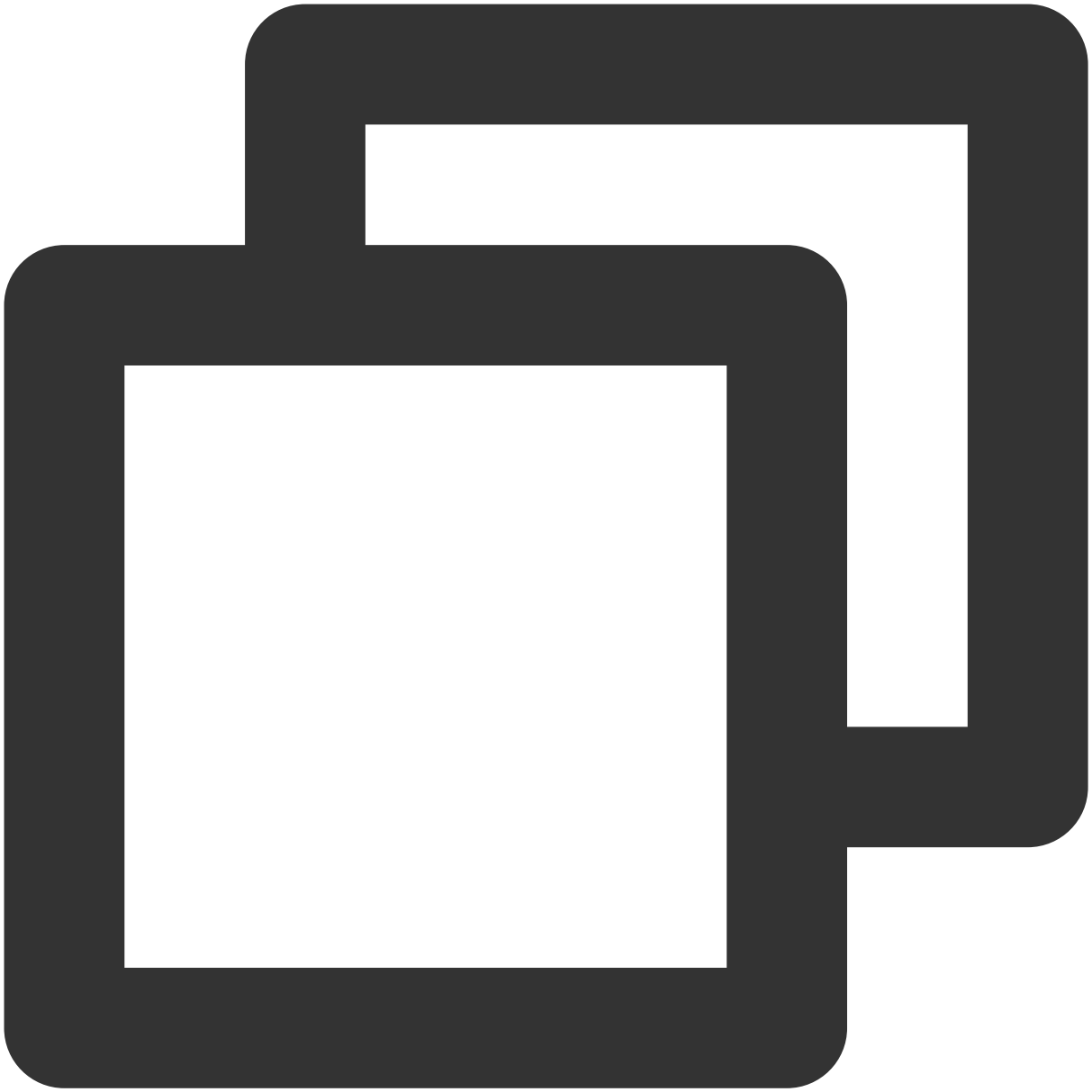
State	开关状态，On 或者 Off	String
Type	回调类型，Url	String
Event	触发回调的事件	String

更新语音识别队列

功能说明

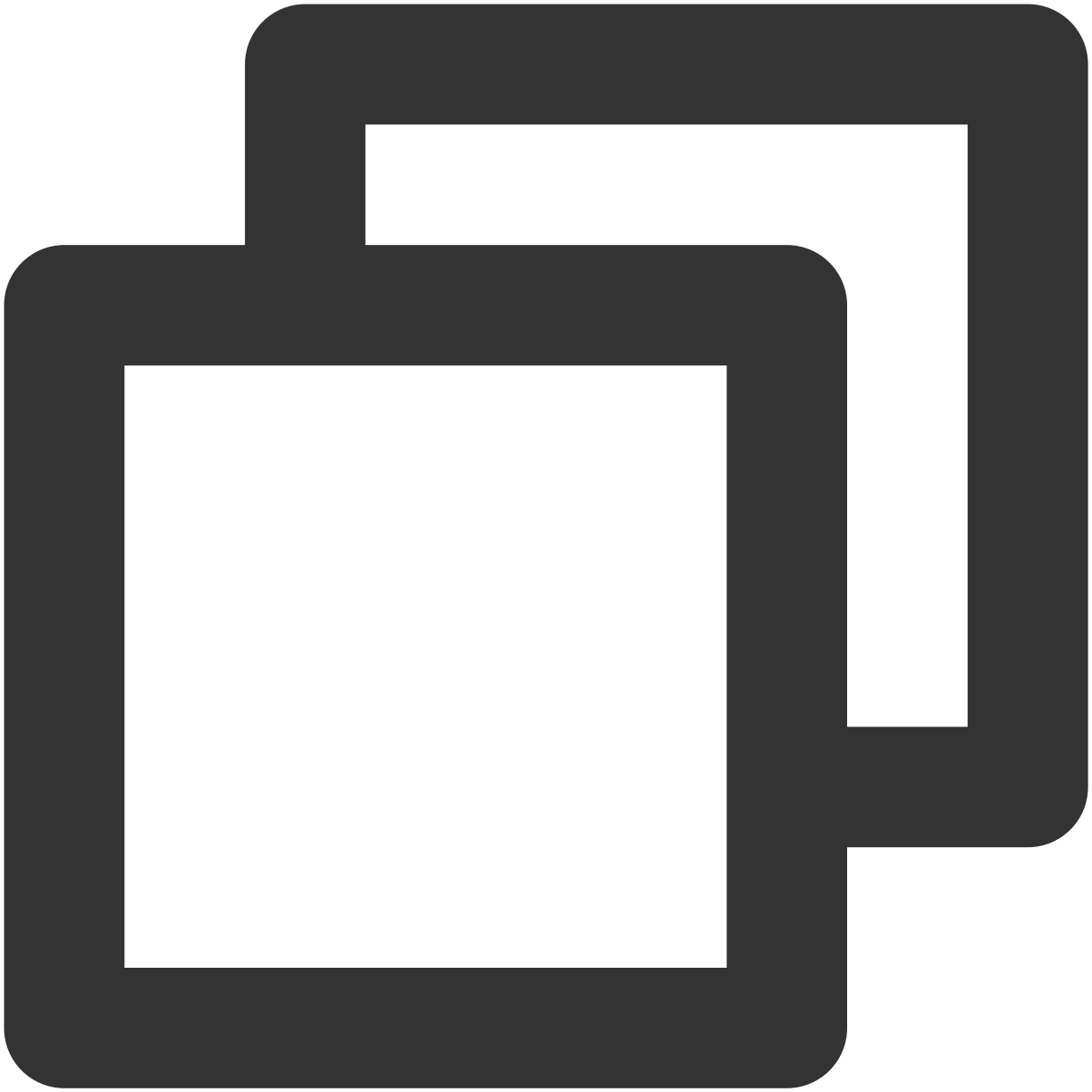
ci_update_asr_queue 接口用于更新语音识别队列。

方法原型



```
def ci_update_asr_queue(self, Bucket, QueueId, Request={}, **kwargs):
```

请求示例



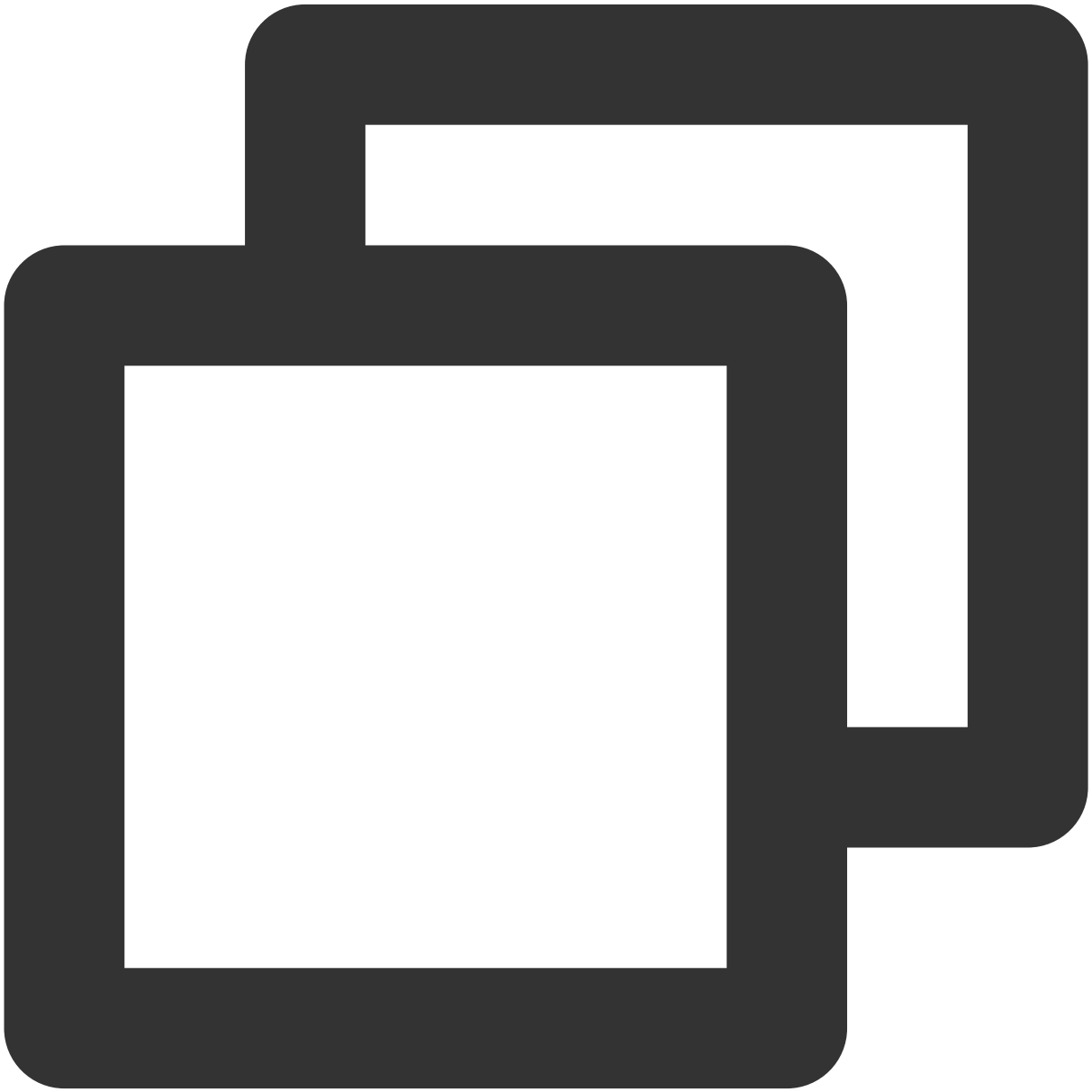
```
def ci_put_asr_queue():
    # 更新语音识别队列信息
    body = {
        'Name': 'speech-queue',
        'QueueID': 'p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxf1',
        'State': 'Active',
        'NotifyConfig': {
            'Type': 'Url',
            'Url': 'http://www.demo.callback.com',
            'Event': 'TaskFinish',
            'State': 'On',
```

```
            'ResultFormat': 'JSON'
        }
    }
    response = client.ci_update_asr_queue(
        Bucket=bucket_name,
        QueueId='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxf1',
        Request=body,
        ContentType='application/xml'
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	存储桶名称	String
Name	队列名称	String
QueueID	队列 ID	String
Request	更新队列配置请求体	String

返回结果说明



```
{
    'RequestId': 'NjMyMWEzMTFfZWMxxxxxxxxxxxxxxxxxxxx',
    'Queue': [{
        'QueueId': 'p4bdf22746b0xxxxxxxxxxxxxxxxx',
        'Name': 'speech-queue',
        'State': 'Active',
        'NotifyConfig': {
            'Url': 'http://www.demo.callback.com',
            'Event': 'TaskFinish',
            'Type': 'Url',
            'State': 'On',
```

```
            'ResultFormat':
            'JSON',
            'MqMode': None,
            'MqName': None,
            'MqRegion': None
        },
        'MaxSize': '10000',
        'MaxConcurrent': '10',
        'CreateTime': '2022-08-11T14:42:01+0800',
        'UpdateTime': '2022-09-14T11:18:08+0800',
        'BucketId': 'test-125xxxxxxxxxx',
        'Category': 'Speeching'
    }]
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	dict
Queue	队列信息，详情请参见 查询语音识别队列中的QueueList	dict

图片二维码

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于图片二维码相关 API 概览以及 SDK 示例代码。

API	操作描述
二维码识别	二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克

二维码识别

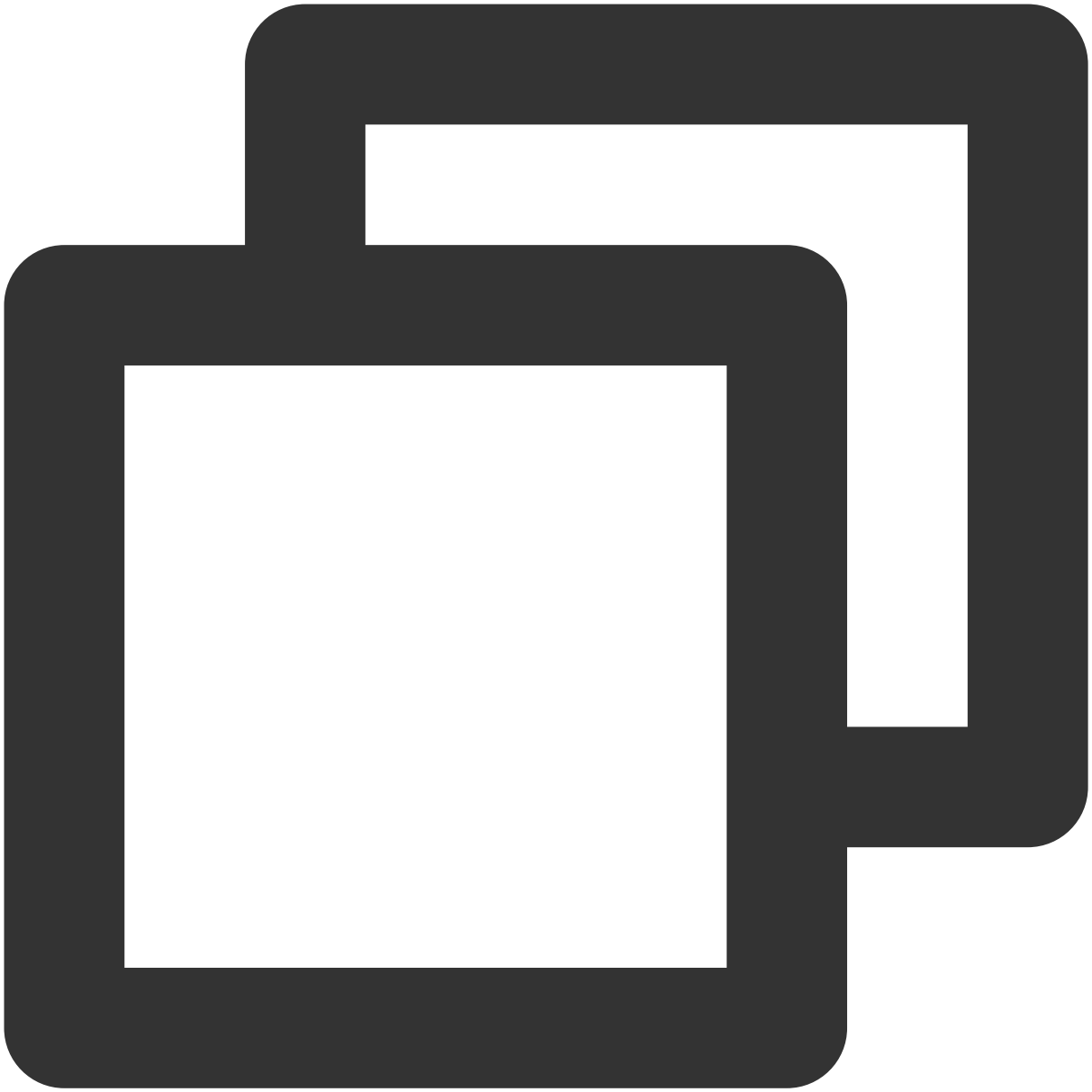
二维码识别功能可识别图片中有效二维码的位置及内容，输出图像中二维码包含的文本信息（每个二维码对应的 URL 或文本），并可对识别出的二维码添加马赛克。

上传时识别二维码

功能说明

图片上传时识别二维码的请求包与 COS 简单上传文件接口一致，只需在请求包头部增加图片处理参数 Pic-Operations。

示例代码



```
# 先创建 cos client
example_object = 'example_object.jpg'
with open(example_object, 'rb') as fp:
    opts = '{"is_pic_info":1,"rules":[{"fileid":"format.jpg","rule":"QRcode/cov
    response,data = client.ci_put_object_from_local_file_and_get_qrcode(
        Bucket='example-bucket-123456789',
        LocalFilePath=example_object,
        Key='example_key',
        EnableMD5=False,
        PicOperations=opts
    )
```

```
        # 查看响应信息，可根据需要读指定数据
        print(response.data)
```

参数说明

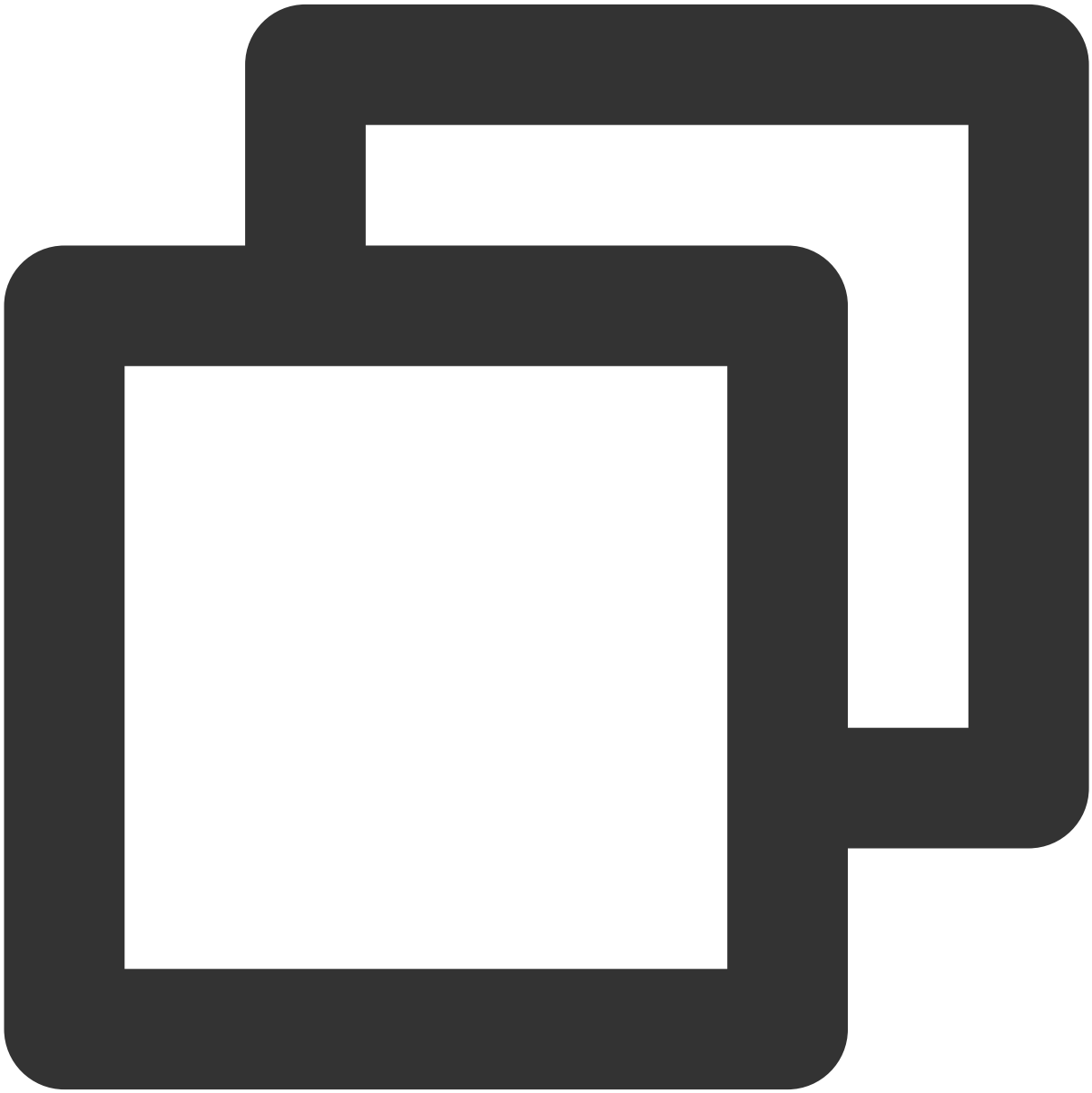
参数名称	参数描述	类型	是否必填
Bucket	存储桶名称，由 BucketName-APPID 构成	String	是
LocalFilePath	图片路径	String	是
Key	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	String	是
EnableMD5	是否需要 SDK 计算 Content-MD5，默认关闭，打开后将增加上传耗时	Bool	否
PicOperations	万象图片处理参数，请参见二维码识别	String	是

云上数据识别二维码

功能说明

对云上图片进行二维码识别，返回二维码识别结果。

示例代码



```
# 先创建 cos client
response,data = client.ci_get_object_qrcode(
    Bucket='example_bucket-123456789',
    Key='example_object',
    Cover=0
)
# 查看响应信息，可根据需要读指定数据
print(response,data)
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶名称，由 BucketName-APPID 构成	String	是
Key	对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` 中，对象键为 doc/pic.jpg	String	是
Cover	万象图片处理二维码覆盖开关，请参见二维码识别	Int	是

文档处理
文档转码

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文档预览的相关的 API 概览以及 SDK 示例代码。

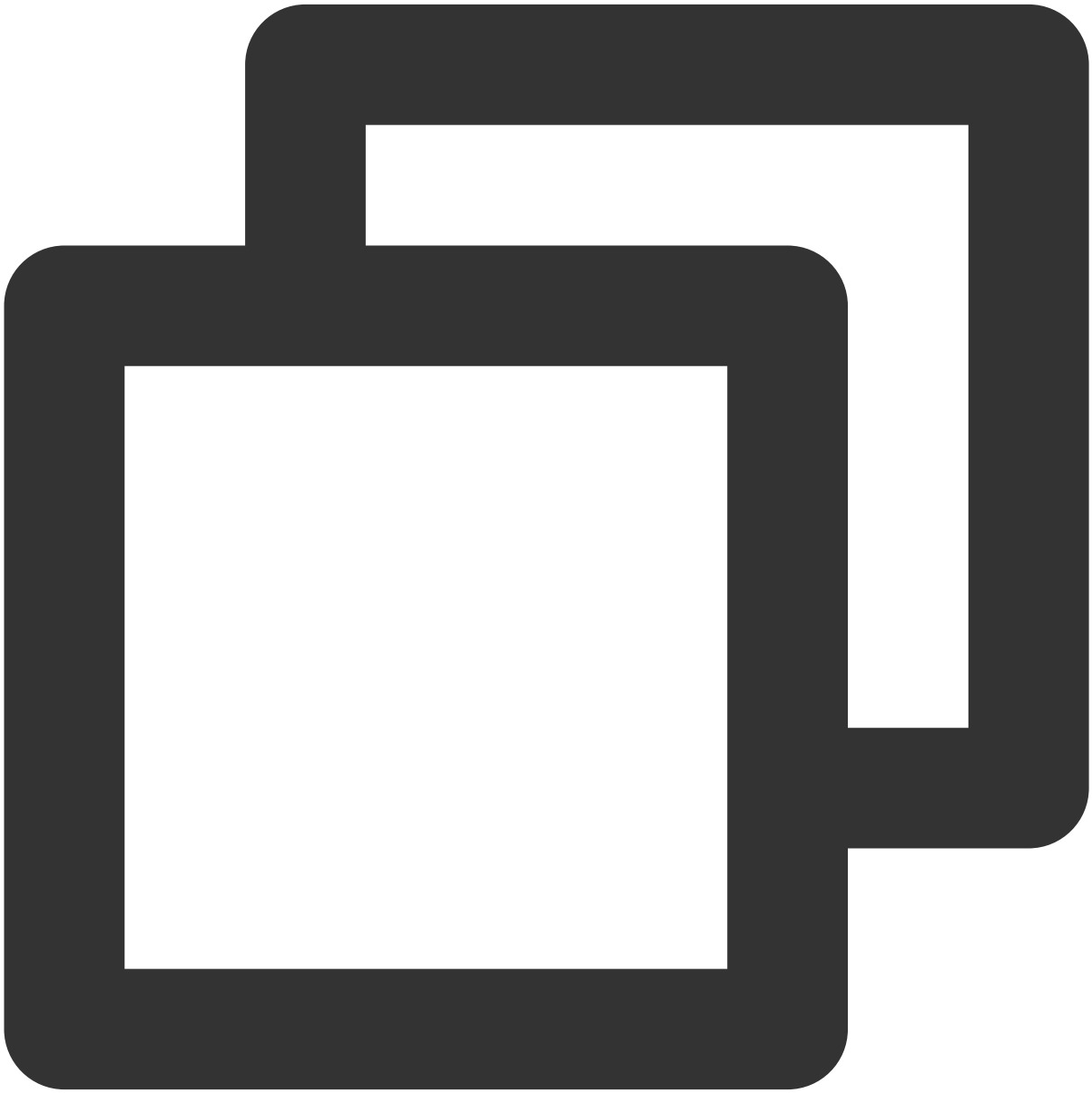
API	操作名	操作描述
CreateDocProcessJobs	提交文档预览任务	用于提交一个文档预览任务
DescribeDocProcessJob	查询文档预览任务	查询指定的文档预览任务
DescribeDocProcessJobs	拉取文档预览任务	拉取符合条件的文档预览任务

提交文档预览任务

功能说明

ci_create_doc_job 接口用于提交一个文档预览任务。

示例代码



```
response = client.ci_create_doc_job(
                Bucket="examplebucket-1250000000",
                QueueId='pbbbcc56b344e422da78c984be45*****',
                InputObject='normal.pptx',
                OutputBucket="examplebucket-1250000000",
                OutputRegion='ap-chongqing',
                OutputObject='/test_doc/normal/abc_${Number}.jpg',
            )
 print(response)
 return response
```

参数说明

调用 ci_create_doc_job 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
QueueId	任务所在的队列 ID。	String	是
InputObject	文件在 COS 上的文件路径。	String	是
OutputBucket	存储结果的存储桶。	String	是
OutputRegion	存储结果的存储桶的地域。	String	是
OutputObject	输出文件路径。 **非表格文件输出文件名需包含 ${Number} 或 ${Page} 参数。**多个输出文件，${Number} 表示序号从1开始，${Page} 表示序号与预览页码一致。 ${Number} 表示多个输出文件，序号从1开始，例如输入 abc_${Number}.jpg，预览某文件5 - 6页，则输出文件名为 abc_1.jpg，abc_2.jpg ${Page} 表示多个输出文件，序号与预览页码一致，例如输入 abc_${Page}.jpg，预览某文件5 - 6页，则输出文件名为 abc_5.jpg，abc_6.jpg。 **表格文件输出路径需包含 ${SheetID} 占位符，输出文件名必须包含 ${Number} 参数。** 例如 `/${SheetID}/abc_${Number}.jpg`，先根据 excel 转换的表格数，生成对应数量的文件夹，再在对应的文件夹下，生成对应数量的图片文件	String	是
SrcType	源数据的后缀类型，当前文档转换根据 COS 对象的后缀名来确定源数据类型，当 COS 对象没有后缀名时，可以设置该值。	String	否
TgtType	转换输出目标文件类型： jpg，转成 jpg 格式的图片文件。如果传入的格式未能识别，默认使用 jpg 格式。 png，转成 png 格式的图片文件。 pdf，转成 pdf 格式文件（暂不支持指定页数）。	String	否
StartPage	从第 X 页开始转换。在表格文件中，一张表可能分割为多页转换，生成多张图片。StartPage 表示从指定 SheetId 的第 X 页开始转换。默认为1。	Int	否
EndPage	转换至第 X 页。在表格文件中，一张表可能分割为多页转换，生成多张图片。EndPage 表示转换至指定 SheetId 的第 X 页。默认为-1，即	Int	否

版权所有：腾讯云计算（北京）有限责任公司

	转换全部页。		
SheetId	表格文件参数，转换第 X 个表，默认为0。设置 SheetId 为0，即转换文档中全部表。	Int	否
PaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0。	Int	否
PaperSize	设置纸张（画布）大小，对应信息为：`0 → A4` 、`1 → A2` 、`2 → A0` ，默认 A4 纸张。	Int	否
DocPassword	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段。	String	否
Comments	是否隐藏批注和应用修订，默认为 0。 0：隐藏批注，应用修订。 1：显示批注和修订。	Int	否
ImageParams	转换后的图片处理参数，支持基础图片处理所有处理参数，多个处理参数可通过管道操作符分隔，从而实现在一次访问中按顺序对图片进行不同处理。	String	否
Quality	生成预览图的图片质量，取值范围 [1-100]，默认值100。 例：值为100，代表生成图片质量为100%。	Int	否
Zoom	预览图片的缩放参数，取值范围[10-200]， 默认值100。 例：值为200，代表图片缩放比例为200% 即放大两倍。	Int	否
ImageDpi	按指定 dpi 渲染图片，该参数与 Zoom 共同作用，取值范围 96-600，默认值为 96。转码后的图片单边宽度需小于65500像素。	Int	否
PicPagination	是否转换成单张长图，设置为 1 时，最多仅支持将 20 标准页面合成单张长图，超过可能会报错，分页范围可以通过 StartPage、EndPage 控制。默认值为 0，按页导出图片，TgtType="png"/"jpg" 时生效。	Int	否

返回参数说明

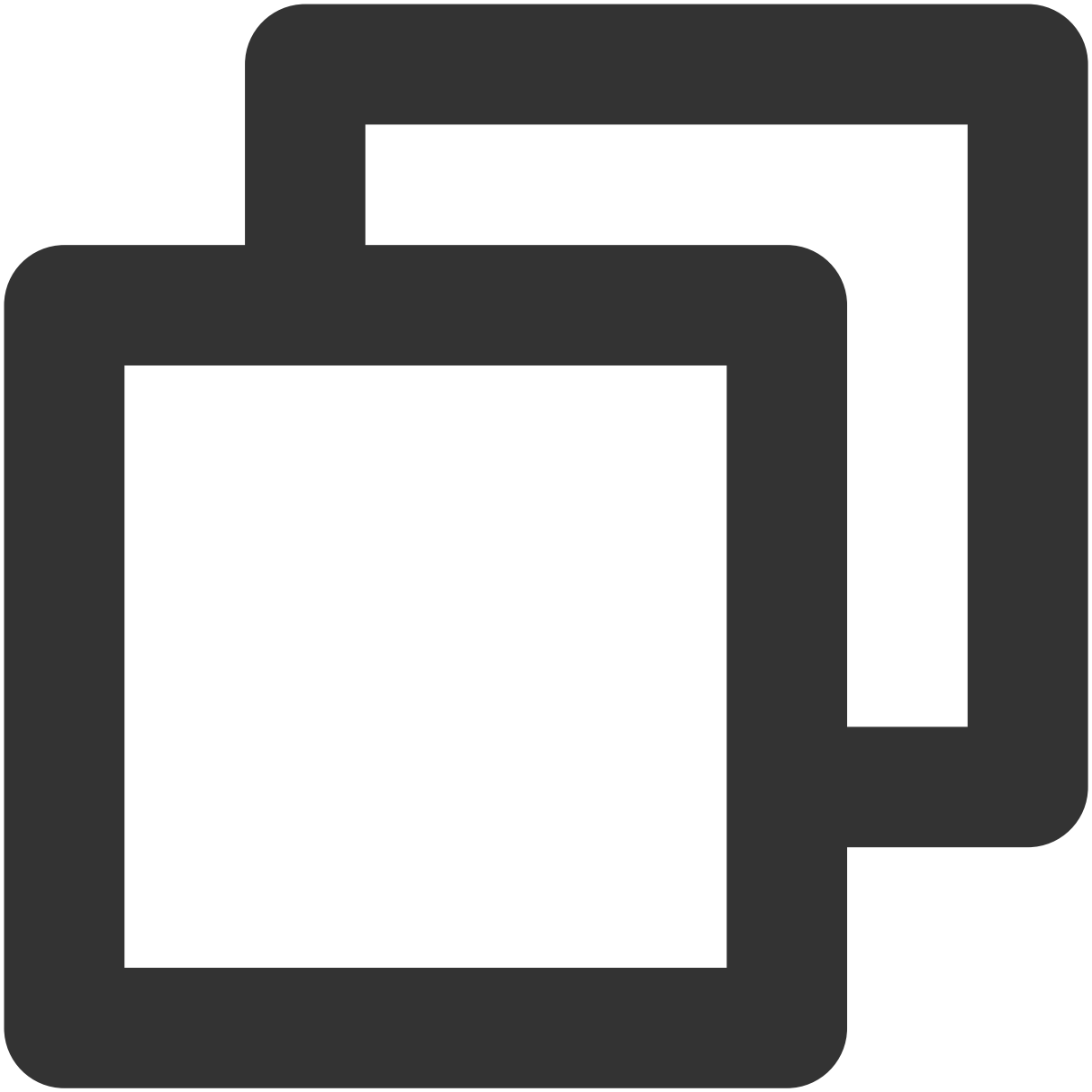
调用 ci_create_doc_jobs 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 提交文档预览任务 文档。

查询文档预览任务

功能说明

ci_get_doc_job 用于查询指定的文档预览任务。

示例代码



```
    response = client.ci_get_doc_job(
                    Bucket="examplebucket-1250000000",
                    JobID='d31d414c8c07811ec894cd30d17*****',
                )
    print(response)
    return response
```

参数说明

调用 ci_get_doc_job 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
JobID	任务 ID。	String	是

返回参数说明

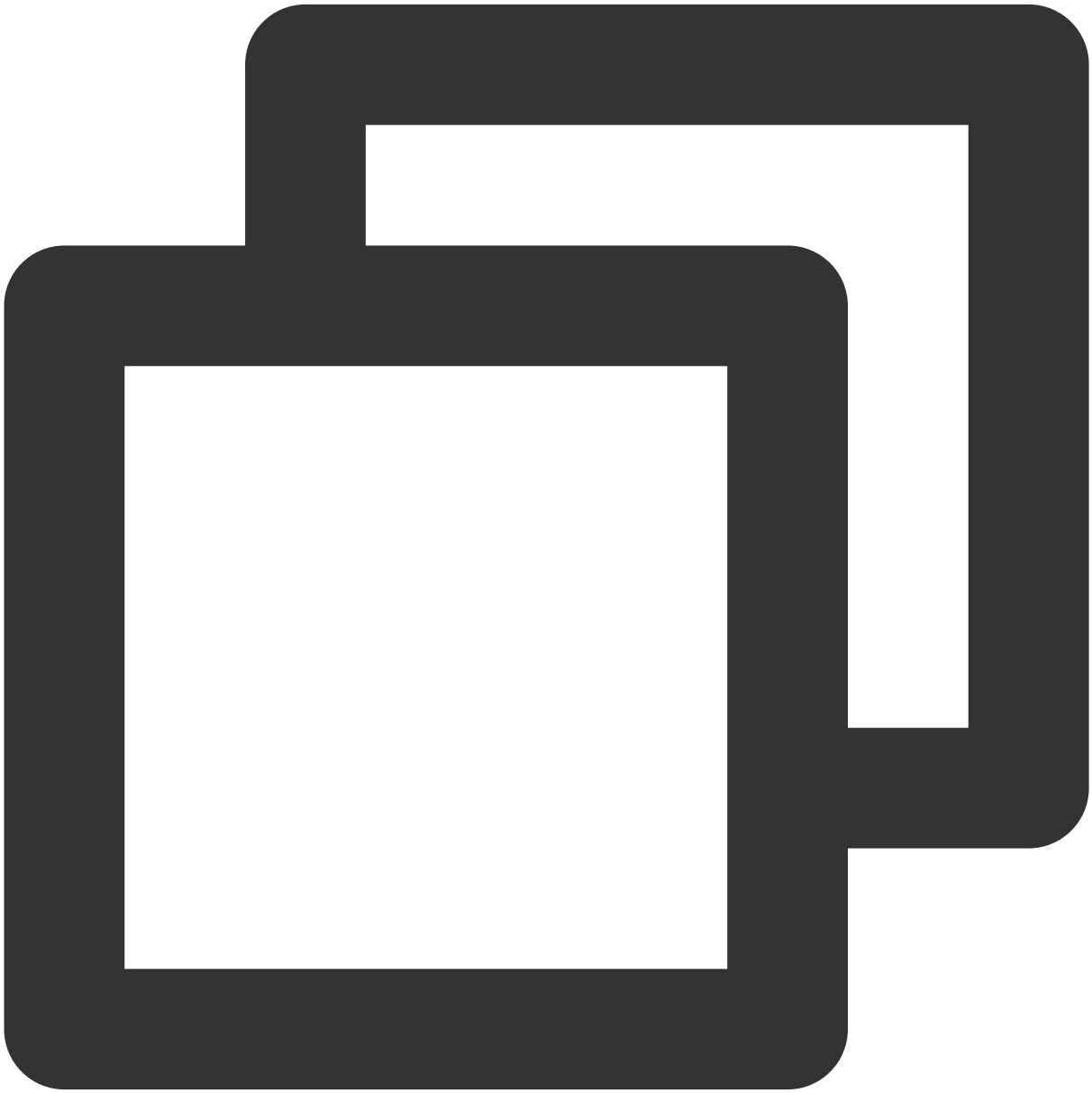
调用 ci_get_doc_job 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 查询文档预览任务 文档。

查询所有文档预览任务

功能说明

ci_list_doc_jobs 用于查询指定的文档预览任务。

示例代码



```
    response = client.ci_list_doc_jobs(
                    Bucket="examplebucket-1250000000",
                    QueueId='pbbbcc56b344e422da78c984be45*****',
                )
    print(response)
    return response
```

参数说明

调用 ci_list_doc_jobs 函数，具体请求参数如下：

参数名称	描述	类型	是否必选
Bucket	存储桶名称。	String	是
QueueId	任务所在的队列 ID。	String	是
StartCreationTime	开始时间。	String	否
EndCreationTime	结束时间。	String	否
OrderByTime	排序方式。Desc 或者 Asc。默认为 Desc。	String	否
States	拉取该状态的任务，以 `,` 分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel。默认为 All。	String	否
Size	拉取的最大任务数。默认为10。最大为100。	String	否
NextToken	请求的上下文，用于翻页。	String	否

返回参数说明

调用 ci_list_doc_jobs 函数，会把 api 里面的 xml 返回转换成 dict，具体返回参数可查看 拉取文档预览任务 文档。

同步请求接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文档预览同步请求的 API 概览以及 SDK 示例代码。

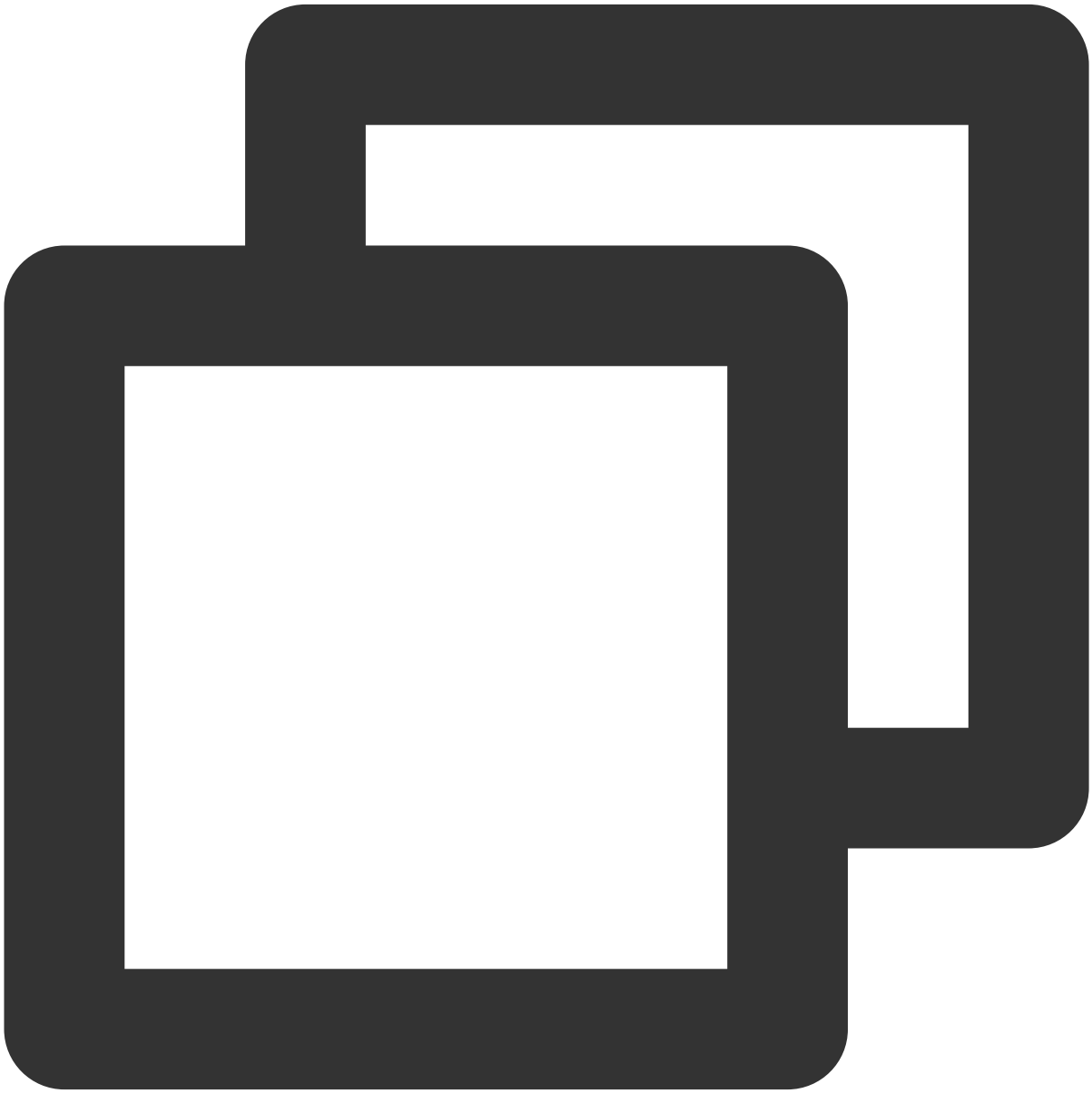
API	说明
同步请求接口	用于同步请求文档预览功能

同步请求接口

功能说明

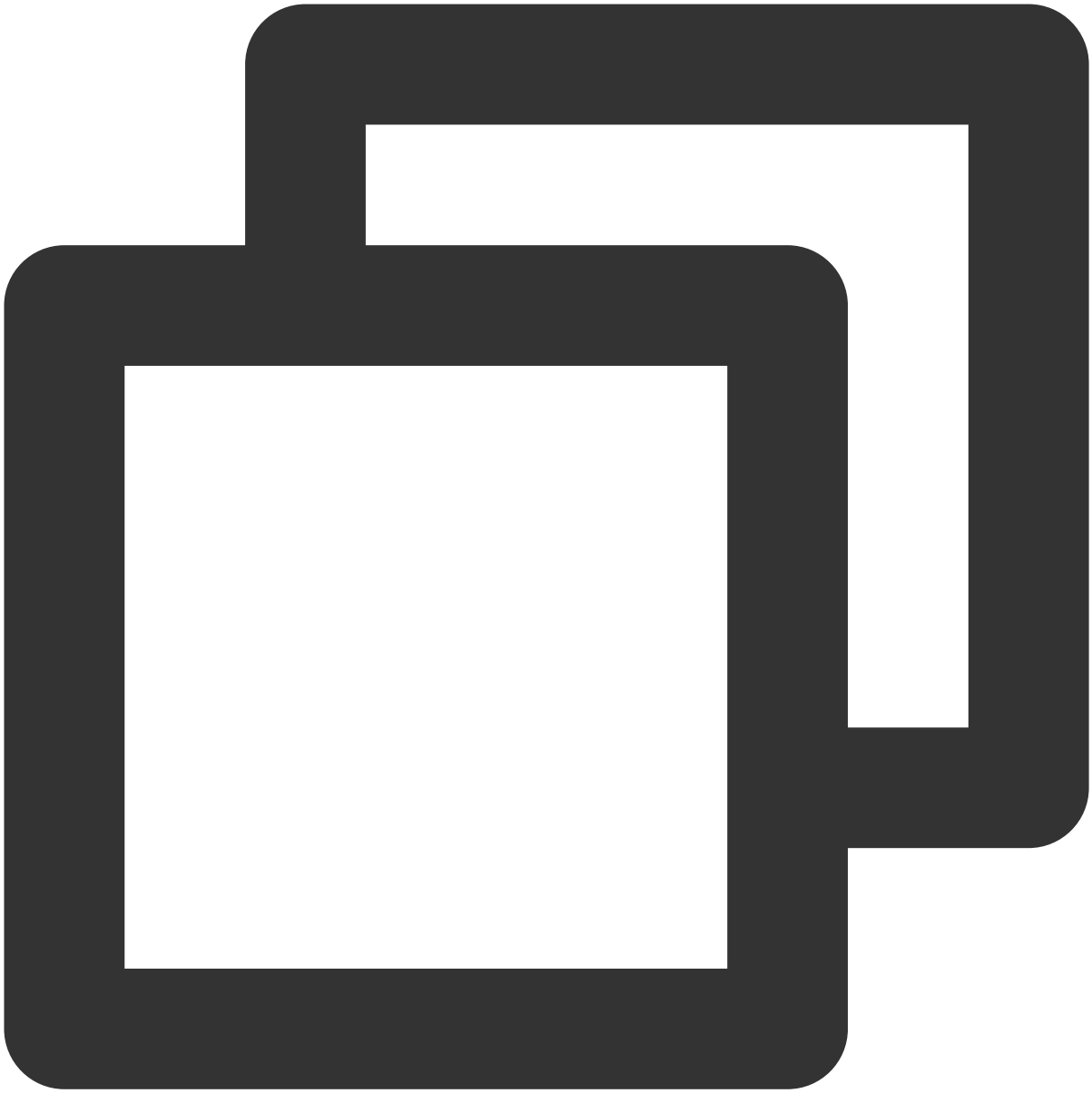
文档预览功能同步请求。

方法原型



```
    def ci_doc_preview_process(self, Bucket, Key, SrcType=None, Page=None, DstType=
                               ExcelPaperDirection=0, ExcelRow=0, ExcelCol=0, Excel
                               ImageParams=None, Quality=100, Scale=100, ImageDpi=9
```

请求示例



```
def ci_doc_preview_process():
    # 文档预览同步接口
    response = client.ci_doc_preview_process(
        Bucket=bucket_name,
        Key='1.txt',
    )
    print(response)
    response['Body'].get_stream_to_file('result.png')
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
Key	对象文件名，例如 folder/document.pdf	String
SrcType	源数据的后缀类型，当前文档转换根据 COS 对象的后缀名来确定源数据类型。当 COS 对象没有后缀名时，可以设置该值	String
Page	需转换的文档页码，默认从1开始计数；表格文件中 page 表示转换的第 x 个 sheet 的第 x 张图	int
DstType	转换输出目标文件类型，png 表示转换成 png 格式的图片文件；jpg 表示转换成 jpg 格式的图片文件；pdf 表示转换成 pdf 格式的图片文件。如果传入的格式未能识别，默认使用 jpg 格式	String
Password	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String
Comment	是否隐藏批注和应用修订，默认为 0。 0：隐藏批注，应用修订；1：显示批注和修订	int
Sheet	表格文件参数，转换第 x 个表，默认为1	int
ExcelPaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	int
ExcelRow	值为1表示将所有列放到 1 页进行排版，默认值为 0	int
ExcelCol	值为1表示将所有行放到 1 页进行排版，默认值为 0	int
ExcelPaperSize	设置纸张（画布）大小，对应信息为： 0 → A4 、 1 → A2 、 2 → A0 ，默认 A4 纸张 （需配合 excelRow 或 excelCol 一起使用）	int
TxtPagination	是否转换成长文本，设置为 true 时，可以将需要导出的页中的文字合并导出，分页范围可以通过 Ranges 控制。默认值为 false ，按页导出 txt。(ExportType="txt" 时生效)	bool
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String
Quality	生成预览图的图片质量，取值范围：[1-100]，默认值100。 例如：值为100，代表生成图片质量为100%	int
Scale	预览图片的缩放参数，取值范围为 [10, 200]， 默认值100。 例如取值为200，代表图片缩放比例为200% 即放大两倍	int
ImageDpi	按指定 dpi 渲染图片，该参数与 scale 共同作用，取值范围 96-600 ，默认值为 96 。转码后的图片单边宽度需小于65500像素	int

异步处理任务接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文档预览任务接口的 API 概览以及 SDK 示例代码。

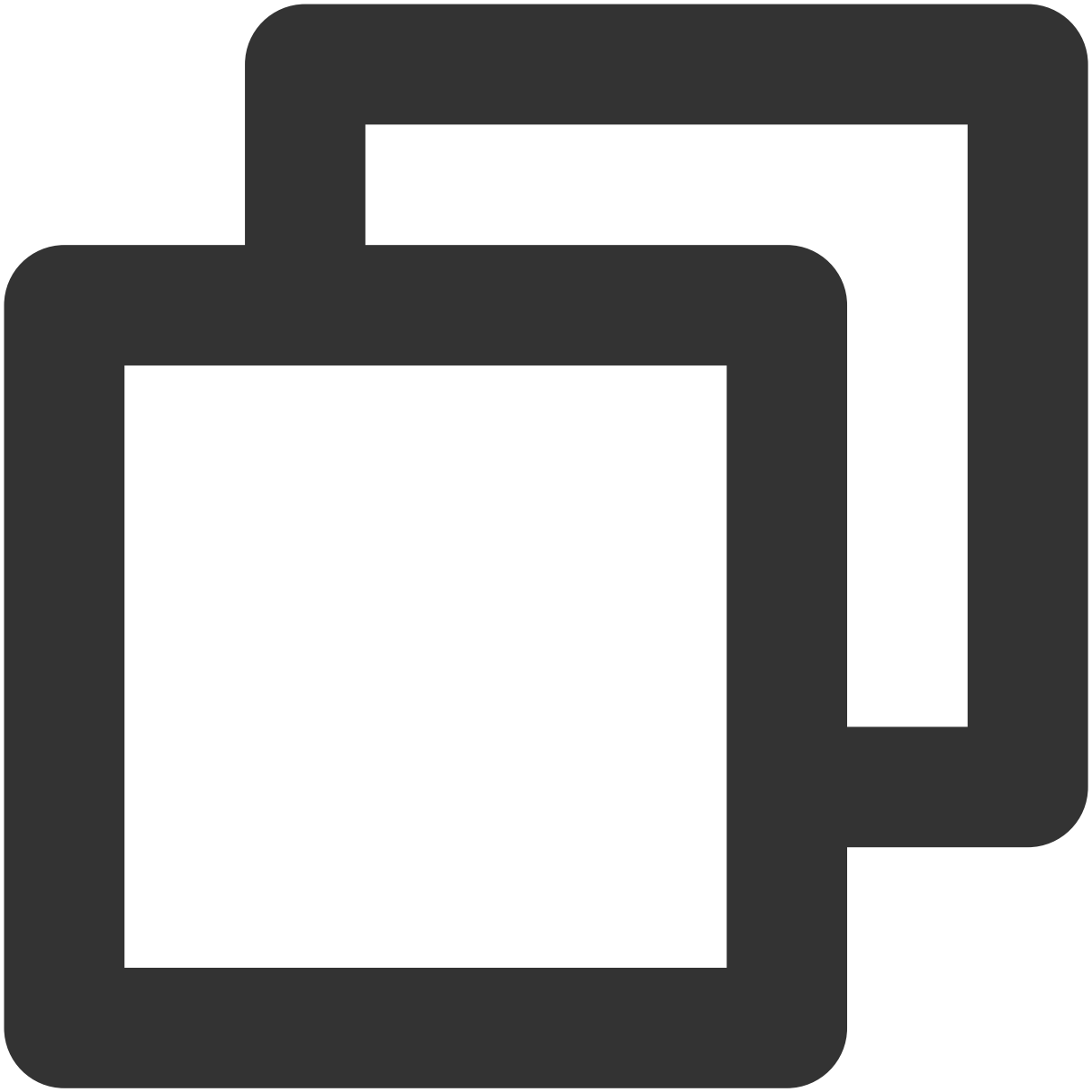
API	操作名	操作描述
CreateDocProcessJobs	提交文档预览任务	用于提交一个文档预览任务
DescribeDocProcessJob	查询指定的文档预览任务	用于查询指定的文档预览任务
DescribeDocProcessJobs	拉取符合条件的文档预览任务	用于拉取符合条件的文档预览任务

提交文档预览任务

功能说明

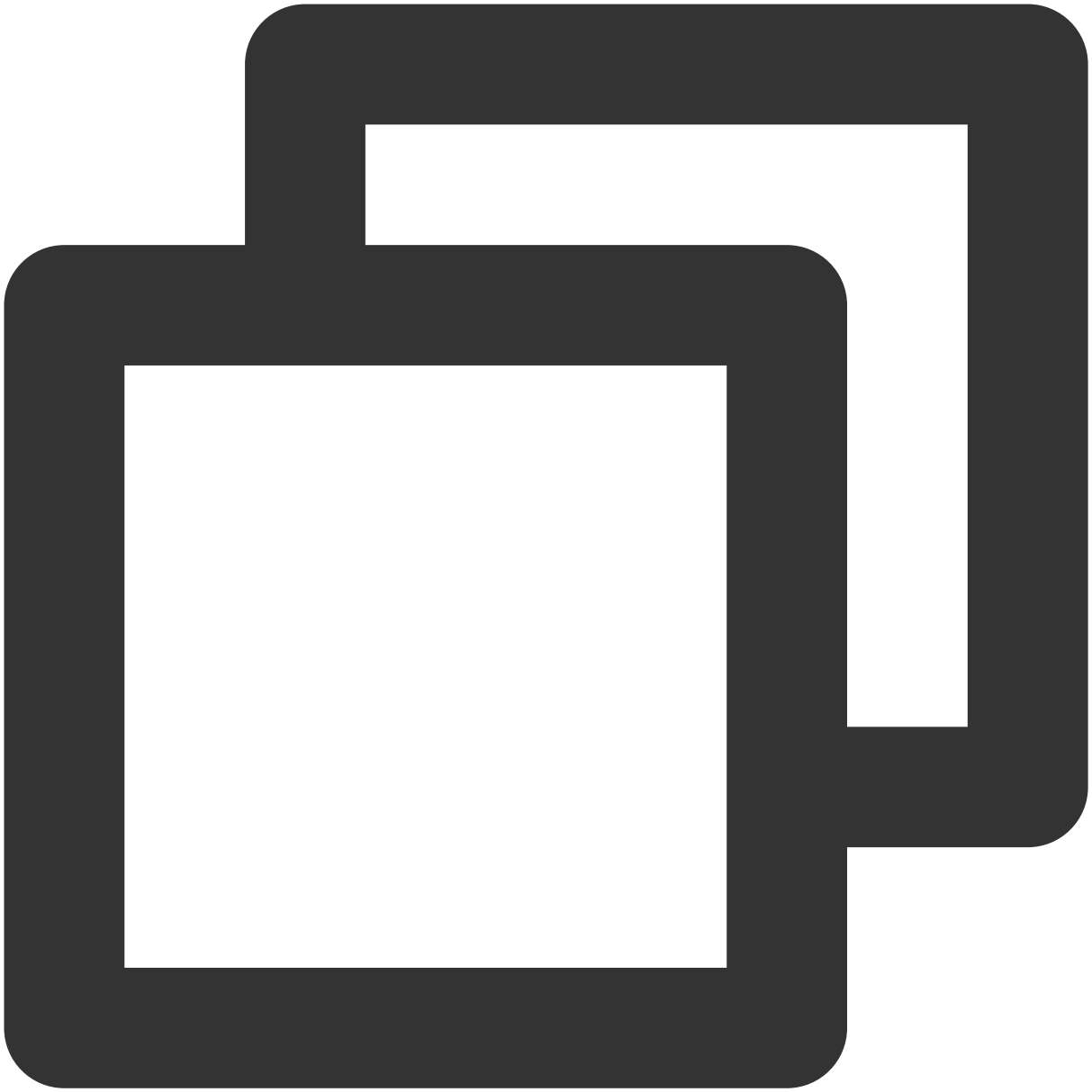
ci_create_doc_job 接口用于提交一个文档预览任务。

方法原型



```
def ci_create_doc_job(self, Bucket, QueueId, InputObject, OutputBucket, OutputRegio
                      StartPage=None, EndPage=-1, SheetId=0, PaperDirection=0,
```

请求示例



```
def ci_create_doc_jobs():
    # 创建文档预览异步任务
    response = client.ci_create_doc_job(
        Bucket=bucket_name,
        QueueId='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxf1',
        InputObject='normal.pptx',
        OutputBucket=bucket_name,
        OutputRegion='ap-chongqing',
        OutputObject='/test_doc/normal/abc_${Number}.jpg',
        # DocPassword='123',
        Quality=109,
```

```
        PageRanges='1,3',
    )
    print(response)
    return response
```

参数说明

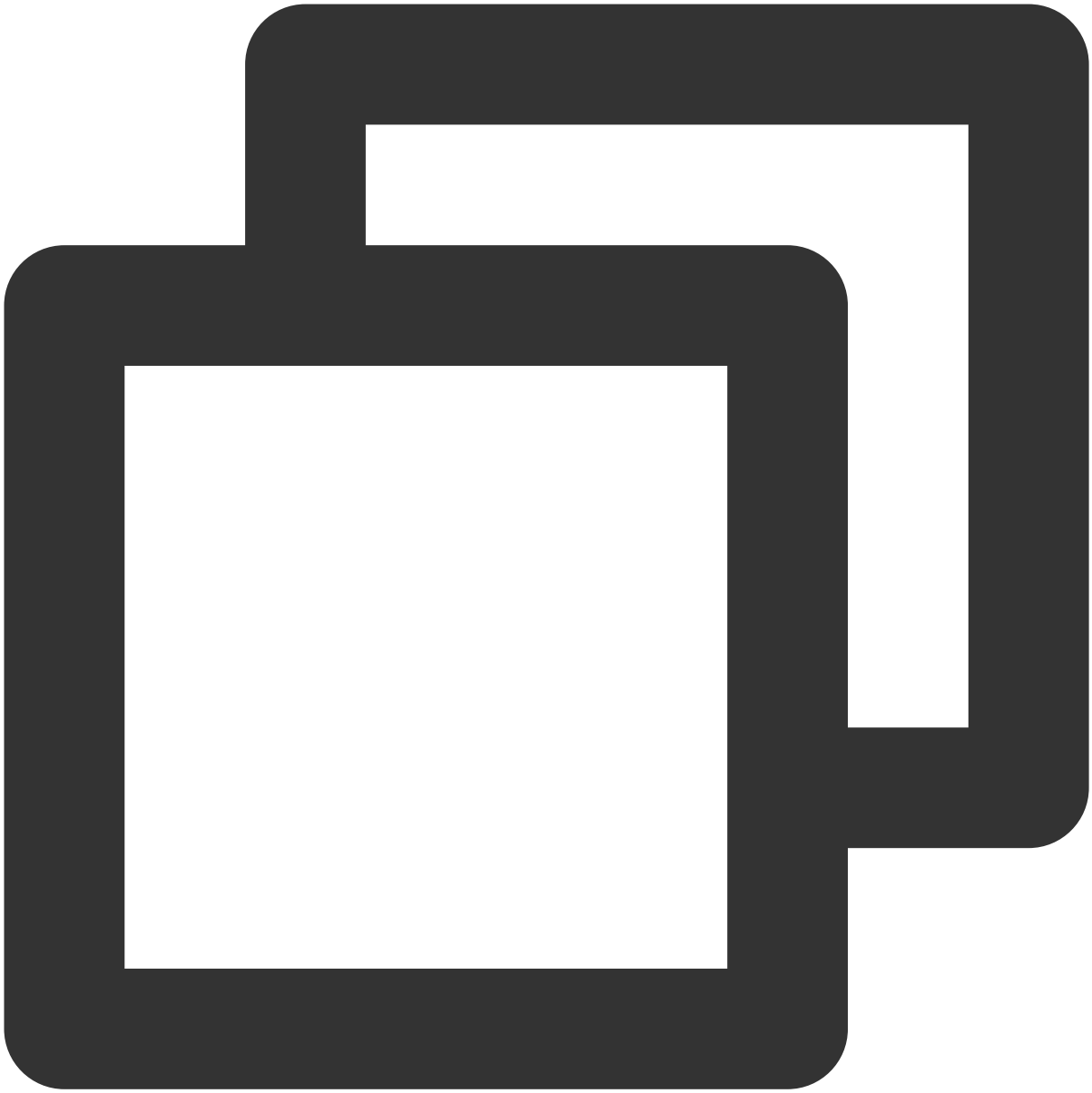
参数名称	描述	类型
Bucket	对象所在存储桶	String
QueueId	任务所在的队列 ID	String
InputObject	待操作的文件对象	Container
OutputRegion	存储桶的地域	String
OutputBucket	存储结果的存储桶	String
OutputObject	输出文件路径。**非表格文件输出文件名需包含${Number}或${Page}参数。**多个输出文件，${Number} 表示序号从1开始，${Page}表示序号与预览页码一致。${Number} 表示多个输出文件，序号从1开始，例如输入 abc_${Number}.jpg，预览某文件5-6页，则输出文件名为 abc_1.jpg，abc_2.jpg${Page}表示多个输出文件，序号与预览页码一致，例如输入 abc_${Page}.jpg，预览某文件5-6页，则输出文件名为 abc_5.jpg，abc_6.jpg。**表格文件输出路径需包含${SheetID}占位符，输出文件名必须包含${Number}参数。**例如 /${SheetID}/ abc_${Number}.jpg，先根据 excel 转换的表格数，生成对应数量的文件夹，再在对应的文件夹下，生成对应数量的图片文件	String
SrcType	源数据的后缀类型。当前文档转换根据 COS 对象的后缀名来确定源数据类型，当 COS 对象没有后缀名时，可以设置该值	String
TgtType	转换输出目标文件类型，png 表示转换成 png 格式的图片文件；jpg 表示转换成 jpg 格式的图片文件；pdf 表示转换成 pdf 格式的图片文件。如果传入的格式未能识别，默认使用 jpg 格式	String
StartPage	从第 x 页开始转换；在表格文件中，一张表可能分割为多页转换，生成多张图片。StartPage 表示从指定 SheetId 的第 x 页开始转换。默认为1	Int
EndPage	转换至第 x 页；在表格文件中，一张表可能分割为多页转换，生成多张图片。EndPage 表示转换至指定 SheetId 的第 x 页。默认为-1，即转换全部页	Int
SheetId	表格文件参数，转换第 x 个表，默认为1；设置 SheetId 为0，即转换文档中全部表格	Int

版权所有：腾讯云计算（北京）有限责任公司

PaperDirection	表格文件转换纸张方向，0代表垂直方向，非0代表水平方向，默认为0	Int
PaperSize	设置纸张（画布）大小，对应信息为： 0 → A4 、 1 → A2 、 2 → A0 ，默认 A4 纸张	Int
DocPassword	Office 文档的打开密码，如果需要转换有密码的文档，请设置该字段	String
Comments	是否隐藏批注和应用修订，默认为 0。 0：隐藏批注，应用修订；1：显示批注和修订	Int
PageRanges	自定义需要转换的分页范围，例如： "1,2-4,7" ，则表示转换文档的 1、2、3、4、7 页面	String
ImageParams	转换后的图片处理参数，支持 基础图片处理 所有处理参数，多个处理参数可通过 管道操作符 分隔，从而实现在一次访问中按顺序对图片进行不同处理	String
Quality	生成预览图的图片质量，取值范围：[1-100]，默认值100。 例如：值为100，代表生成图片质量为100%	Int
Zoom	预览图片的缩放参数，取值范围：[10-200]，默认值100。 例如：值为200，代表图片缩放比例为200% 即放大两倍	Int
ImageDpi	按指定 dpi 渲染图片，该参数与 Zoom 共同作用，取值范围 96-600 ，默认值为 96 。转码后的图片单边宽度需小于65500像素	Int
PicPagination	是否转换成单张长图，设置为 1 时，最多仅支持将 20 标准页面合成单张长图，超过可能会报错，分页范围可以通过 StartPage、EndPage 控制。默认值为 0 ，按页导出图片，TgtType="png"/"jpg" 时生效	Int

返回结果说明

版权所有：腾讯云计算（北京）有限责任公司



```
{
    'JobsDetail': {
        'Code': 'Success',
        'CreationTime': '2022-09-14T00:00:00+0800',
        'EndTime': '-',
        'Input': {
            'BucketId': 'test-125xxxxxxxxxx',
            'Object': '1.txt',
            'Region': 'ap-chongqing'
        },
        'JobId': 'dcf48xxxxxxxxxxxxxxxxxxxxxxxxc523',
```

```
        'Message': None,
        'Operation': {
            'DocProcess': {
                'Comments': '1',
                'DocPassword': None,
                'EndPage': '-1',
                'ImageDpi': '96',
                'ImageParams': None,
                'PageRanges': '1,3',
                'PaperDirection': '0',
                'PaperSize': '0',
                'PicPagination': '0',
                'Quality': '109',
                'SheetId': '0',
                'SrcType': None,
                'StartPage': '0',
                'TgtType': None,
                'Zoom': '100'
            },
            'JobLevel': '0',
            'Output': {
                'Bucket': 'test-125xxxxxxxxxx',
                'Object': '/test_doc/normal/abc_${Number}.jpg',
                'Region': 'ap-chongqing'
            }
        },
        'QueueId': 'p4bxxxxxxxxxxxxxxxxxxxxxxxxxxxx57f1',
        'StartTime': '-',
        'State': 'Submitted',
        'Tag': 'DocProcess'
    }
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息	Container
Code	错误码，只有 State 为 Failed 时有意义	String
Message	错误描述，只有 State 为 Failed 时有意义	String
JobId	新创建任务的 ID	String
Tag	新创建任务的 Tag：DocProcess	String
State	任务的状态，为 Submitted、Running、Success、Failed、Pause、Cancel 其中一个	String

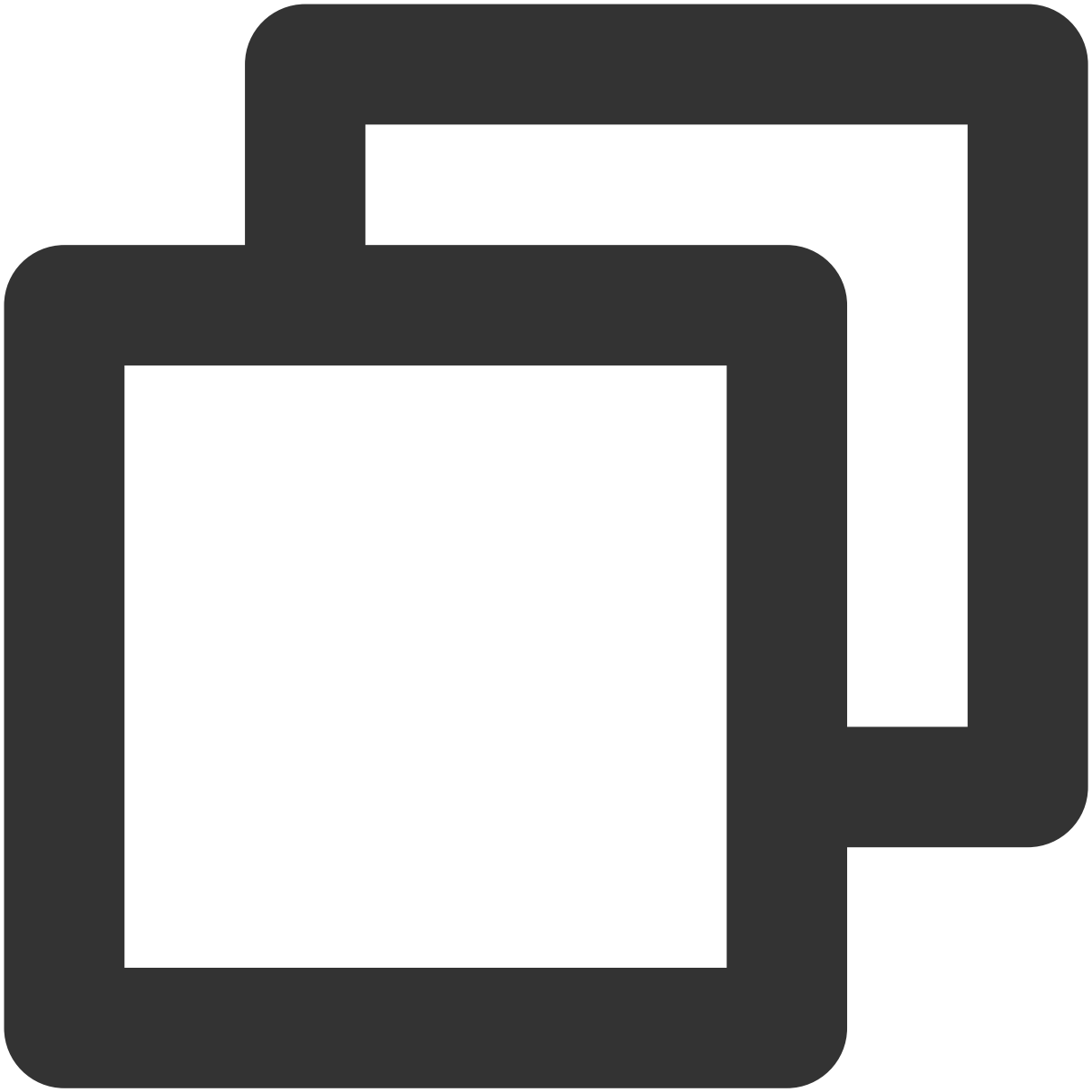
CreationTime	任务的创建时间	String
QueueId	任务所属的队列 ID	String
Input	该任务的输入文件路径	Container
Operation	该任务的规则	Container

查询指定文档预览任务

功能说明

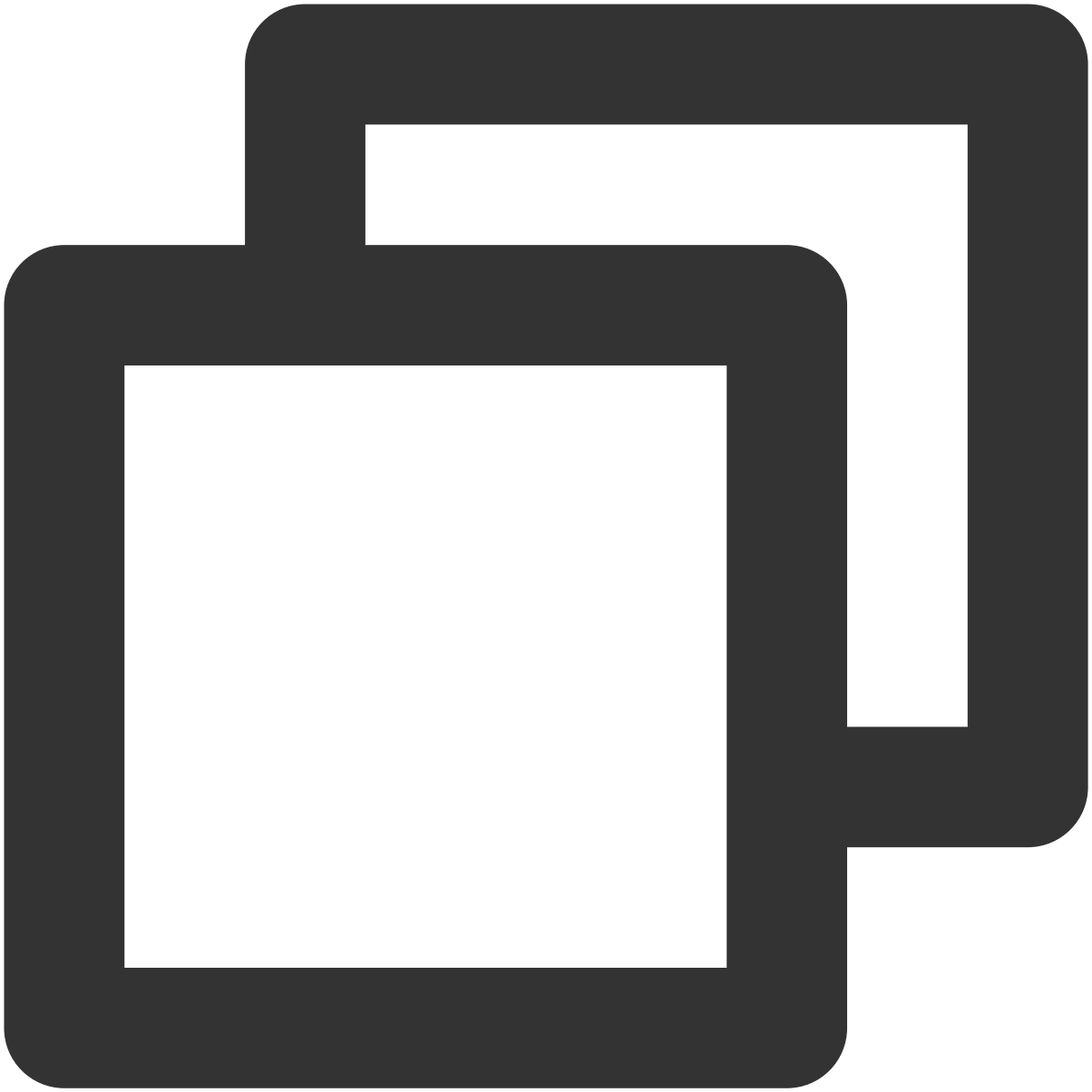
ci_get_doc_job 用于查询指定的文档预览任务。

方法原型



```
def ci_get_doc_job(self, Bucket, JobID, **kwargs):
```

请求示例

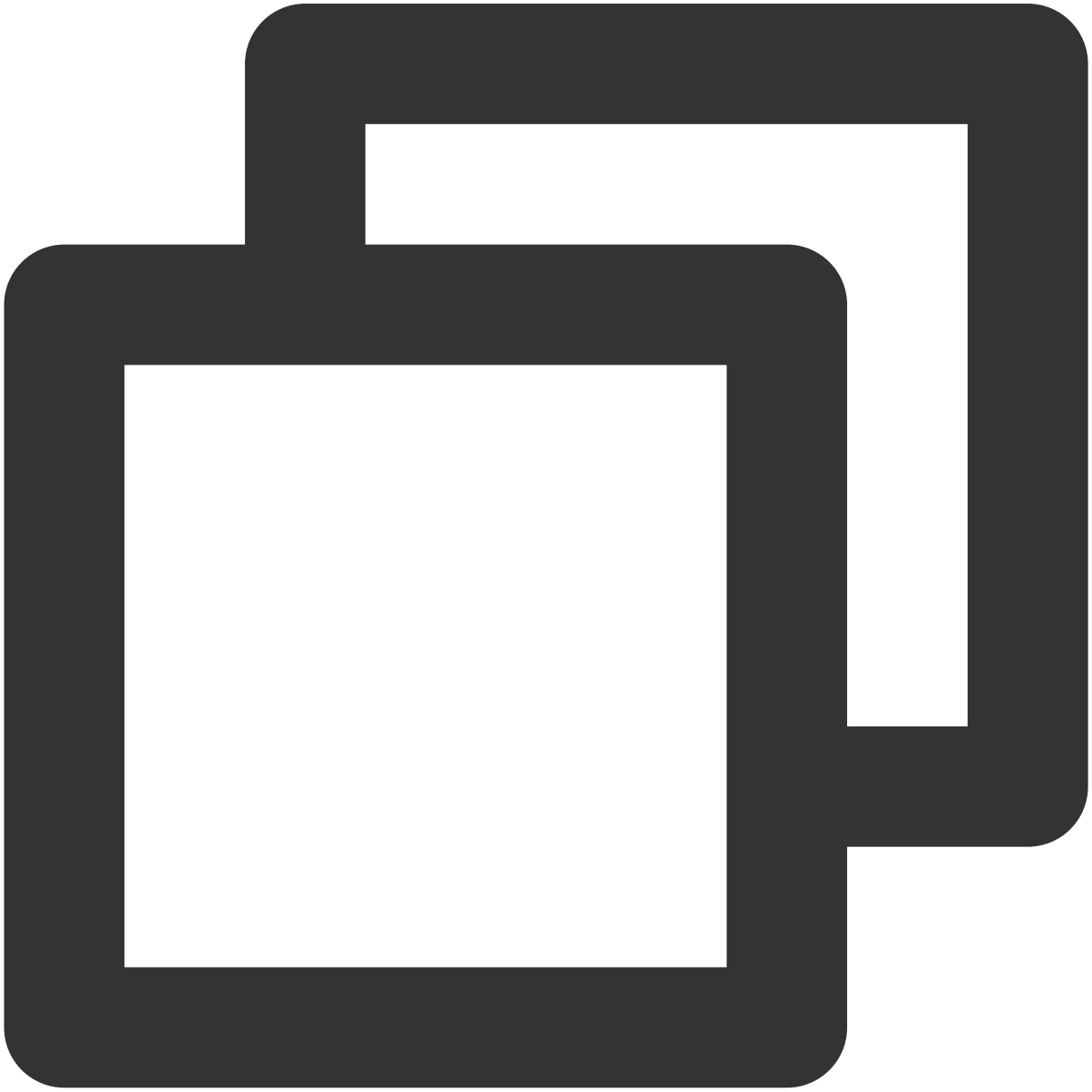


```
def ci_get_doc_jobs():
    # 获取文档预览异步任务信息
    response = client.ci_get_doc_job(
        Bucket=bucket_name,
        JobID='d18a9xxxxxxxxxxxxxxxxxxff1aa',
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
JobID	文档预览任务 ID	String

返回结果说明



```
{
    'JobsDetail':
```

```
    [{
        'Code': 'Success',
        'CreationTime': '2022-09-14T17:15:43+0800',
        'EndTime': '2022-09-14T17:15:45+0800',
        'Input': {'BucketId': 'testpic-1253960454', 'Object': '1.txt', 'Region': 'a
        'JobId': 'dcf488492340d11edb2a0e3aa2e38c523',
        'Message': None,
        'Operation': {
            'DocProcess': {'Comments': '1',
                'DocPassword': None,
                'EndPage': '-1',
                'ImageDpi': '96',
                'ImageParams': None,
                'PageRanges': '1,3',
                'PaperDirection': '0',
                'PaperSize': '0',
                'PicPagination': '0',
                'Quality': '109',
                'SheetId': '0',
                'SrcType': None,
                'StartPage': '0',
                'TgtType': None,
                'Zoom': '100'
            },
            'DocProcessResult': {
                'FailPageCount': '0',
                'PageInfo': [{'PageNo': '1', 'TgtUri': '/test_doc/normal/abc_1.jpg'
                'SuccPageCount': '2',
                'TaskId': None,
                'TgtType': None,
                'TotalPageCount': '7'
            },
            'JobLevel': '0',
            'Output': {'Bucket': 'testpic-1253960454', 'Object': '/test_doc/normal/
        },
        'QueueId': 'p4bdf22746b014a22a5a097b1f1d057f1',
        'StartTime': '2022-09-14T17:15:44+0800',
        'State': 'Success',
        'Tag': 'DocProcess'
    }]
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息， 同 CreateDocProcessJobs 接口的 Response.JobsDetail 节点	Container

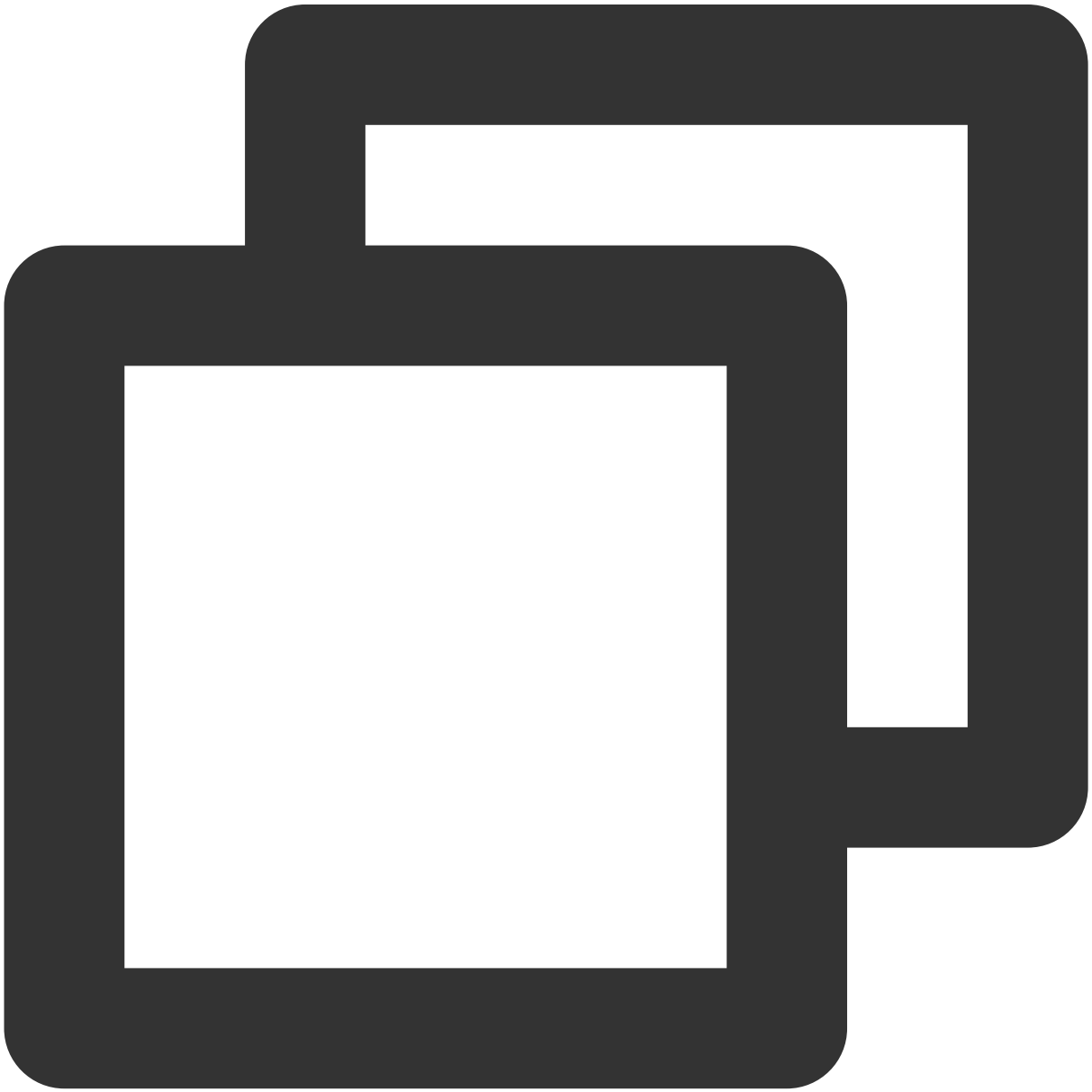
NonExistJobIds	查询的 ID 中不存在的任务，所有任务都存在时不返回	String

拉取符合条件的文档预览任务

功能说明

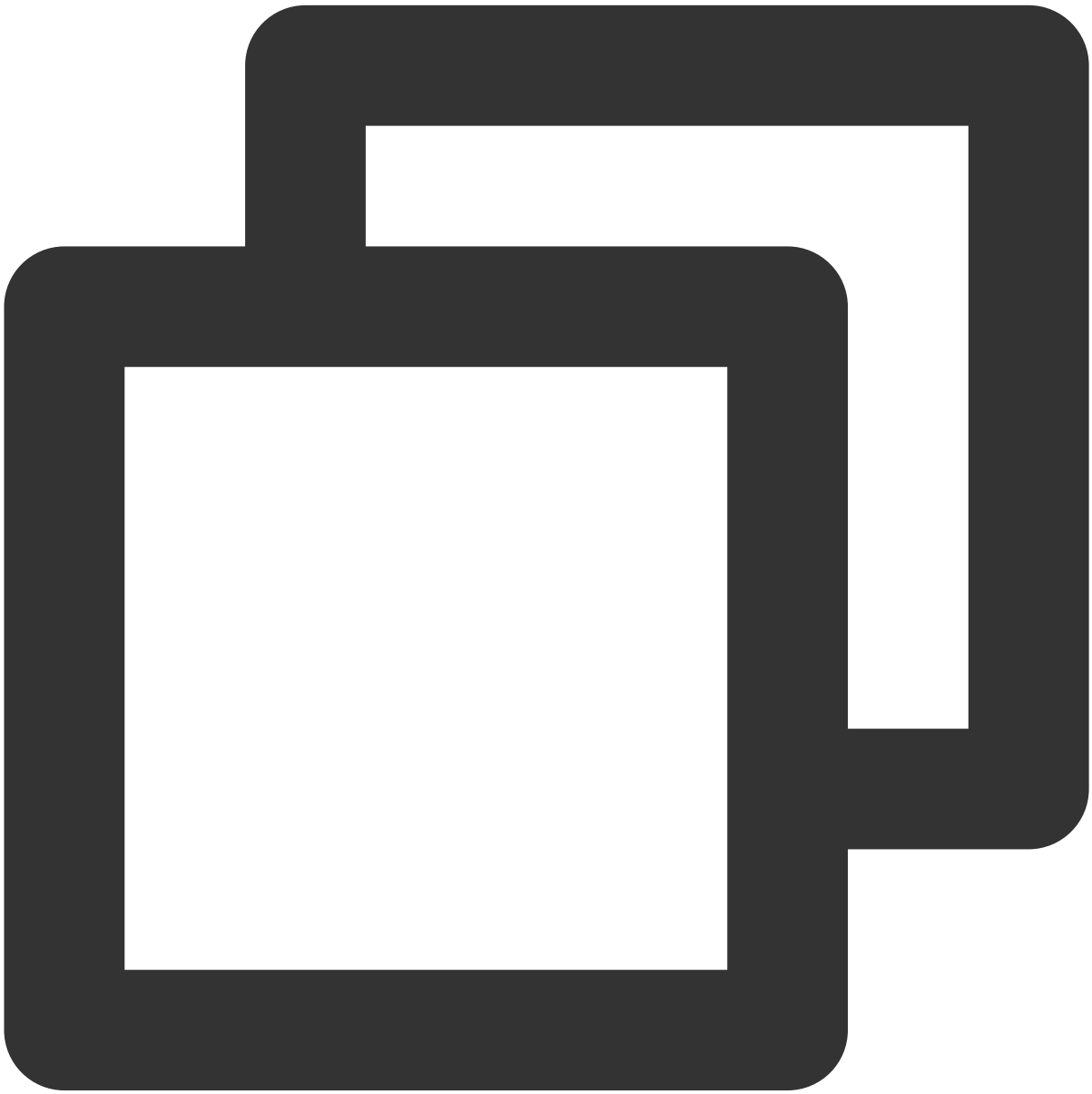
ci_list_doc_jobs 用于拉取符合条件的文档预览任务。

方法原型



```
def ci_list_doc_jobs(self, Bucket, QueueId, StartCreationTime=None, EndCreationTime
```

请求示例



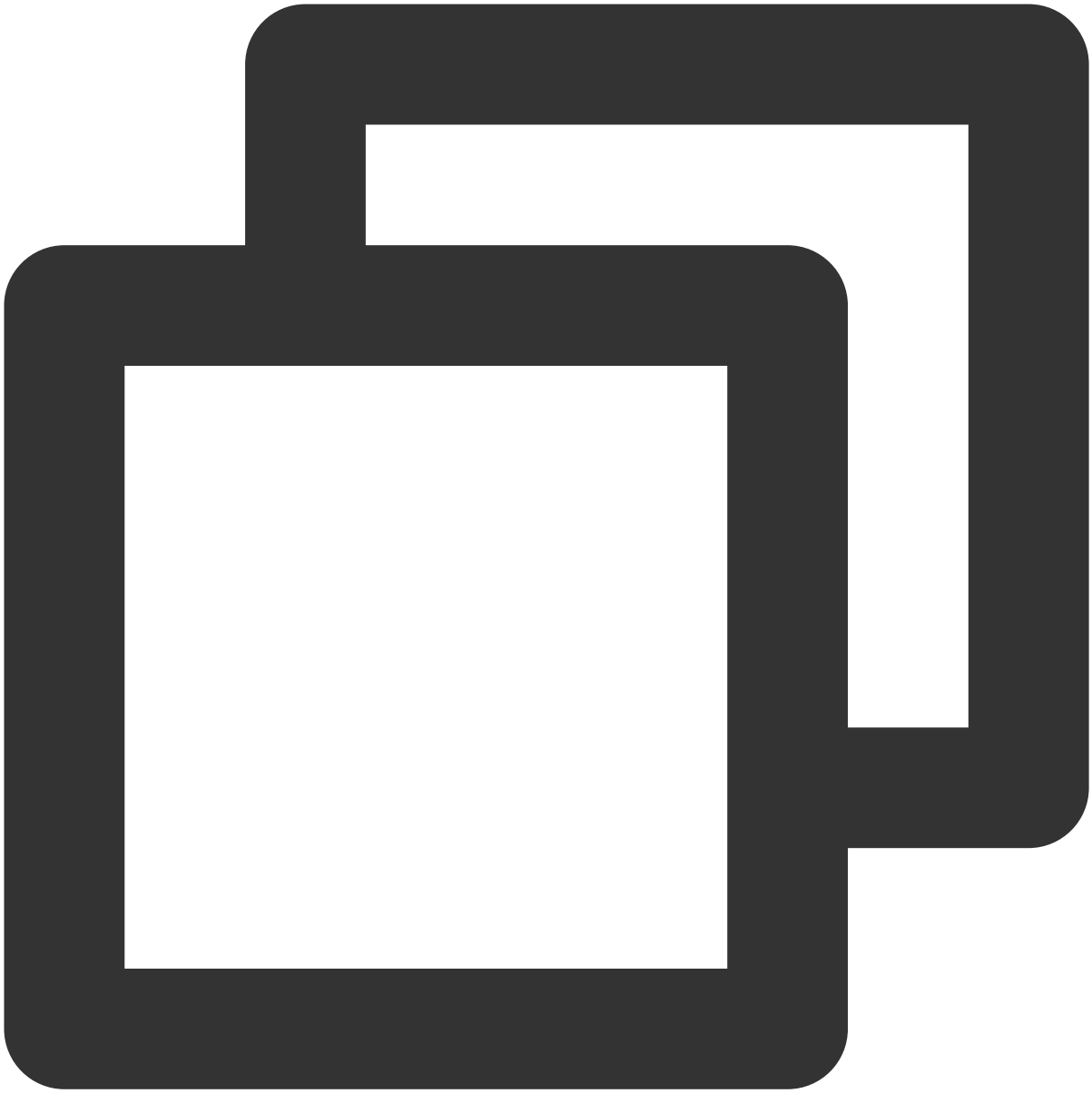
```
def ci_list_doc_jobs():
    # 获取文档预览异步任务信息列表
    response = client.ci_list_doc_jobs(
        Bucket=bucket_name,
        QueueId='p4bdxxxxxxxxxxxxxxxxxxx57f1',
```

```
        Size=10,
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	任务所在存储桶	String
QueueId	拉取该队列 ID 下的任务	String
OrderByTime	Desc 或者 Asc。默认为 Desc	String
NextToken	请求的上下文，用于翻页。上次返回的值	String
Size	拉取的最大任务数。默认为10，最大值为100	Int
States	拉取该状态的任务，以 `,` 分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel。默认为 All	String
StartCreationTime	拉取创建时间大于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String
EndCreationTime	拉取创建时间小于该时间的任务。格式为：`%Y-%m-%dT%H:%m:%S%z`	String

返回结果说明



```
{
    'JobsDetail': [{
        'Code': 'Success',
        'CreationTime': '2022-09-14T17:15:43+0800',
        'EndTime': '2022-09-14T17:15:45+0800',
        'Input': {'BucketId': 'testpic-1253960454', 'Object': '1.txt', 'Region': 'a
        'JobId': 'dcf488492340d11edb2a0e3aa2e38c523',
        'Message': None,
        'Operation': {
            'DocProcess': {
                'Comments': '1',
```

```
                'DocPassword': None,
                'EndPage': '-1',
                'ImageDpi': '96',
                'ImageParams': None,
                'PageRanges': '1,3',
                'PaperDirection': '0',
                'PaperSize': '0',
                'PicPagination': '0',
                'Quality': '109',
                'SheetId': '0',
                'SrcType': None,
                'StartPage': '0',
                'TgtType': None,
                'Zoom': '100'
            },
            'DocProcessResult': {
                'FailPageCount': '0',
                'PageInfo': [{'PageNo': '1', 'TgtUri': '/test_doc/normal/abc_1.jpg'
                'SuccPageCount': '2',
                'TaskId': None,
                'TgtType': None,
                'TotalPageCount': '7'
            },
            'JobLevel': '0',
            'Output': {'Bucket': 'testpic-1253960454', 'Object': '/test_doc/normal/
        },
        'QueueId': 'p4bdf22746b014a22a5a097b1f1d057f1',
        'StartTime': '2022-09-14T17:15:44+0800',
        'State': 'Success',
        'Tag': 'DocProcess'
    }],
    'NextToken': '75789'
}
```

参数名称	描述	类型
JobsDetail	任务的详细信息，同 CreateDocProcessJobs 接口中的 Response.JobsDetail 节点	Container
NextToken	翻页的上下文 Token	String

异步处理队列接口

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文档预览队列接口的 API 概览以及 SDK 示例代码。

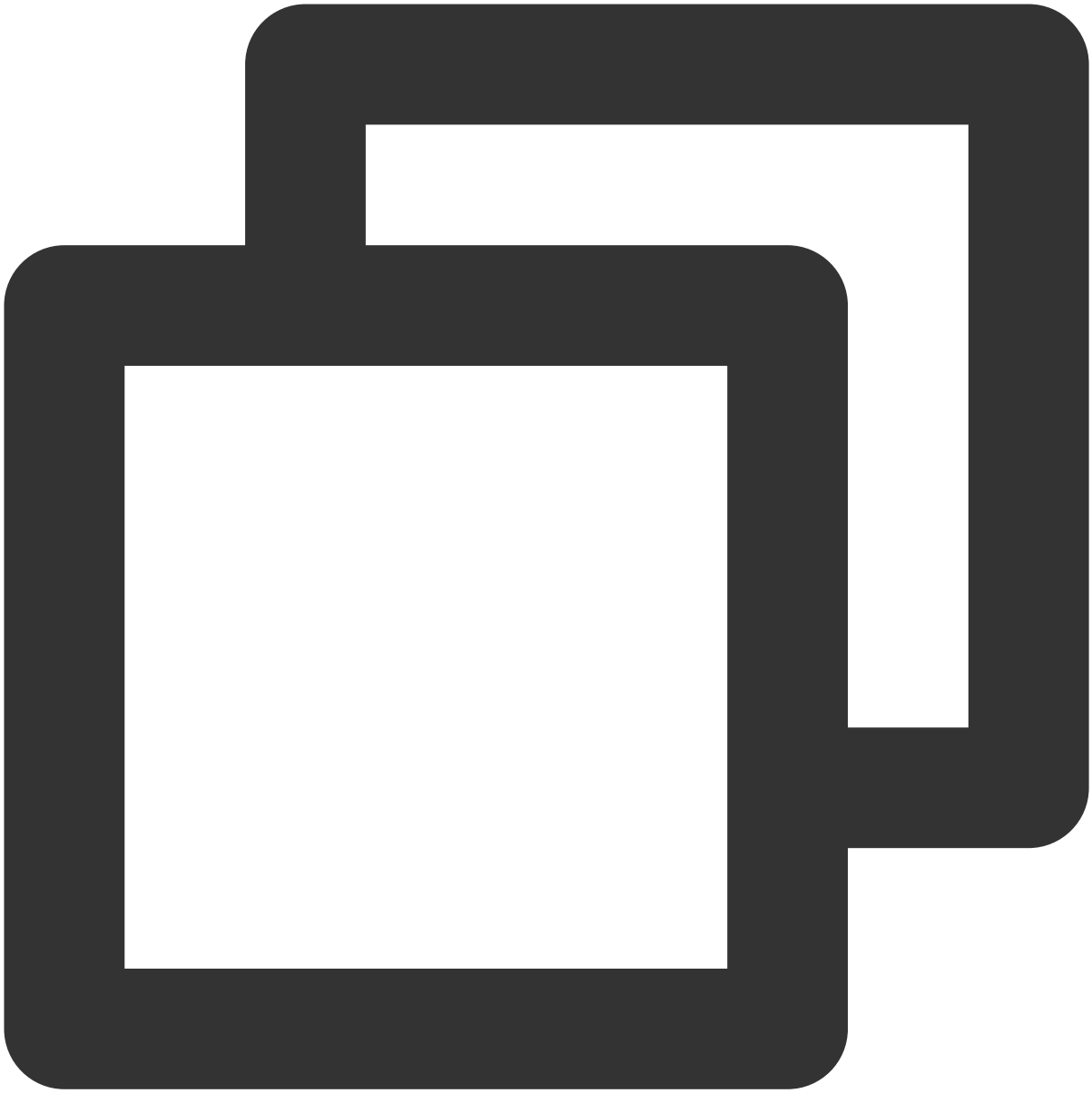
API	操作名	操作描述
DescribeDocProcessQueues	查询文档预览队列	查询文档预览队列信息
UpdateDocProcessQueue	更新文档预览队列	更新文档预览队列信息

查询文档预览队列

功能说明

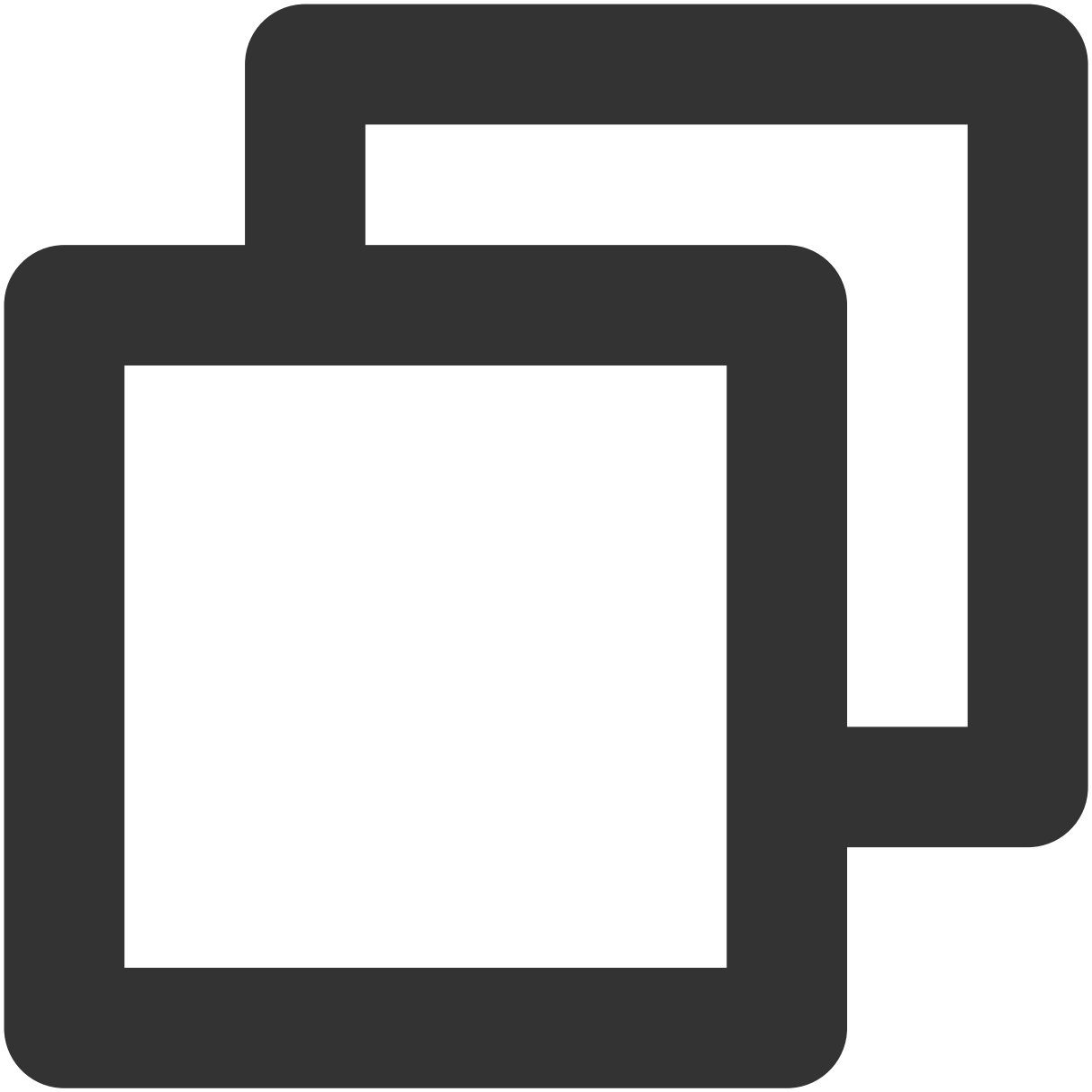
ci_get_doc_queue 接口用于查询文档预览队列。

方法原型



```
def ci_get_doc_queue(self, Bucket, State='All', QueueIds='', PageNumber='', PageSiz
```

请求示例



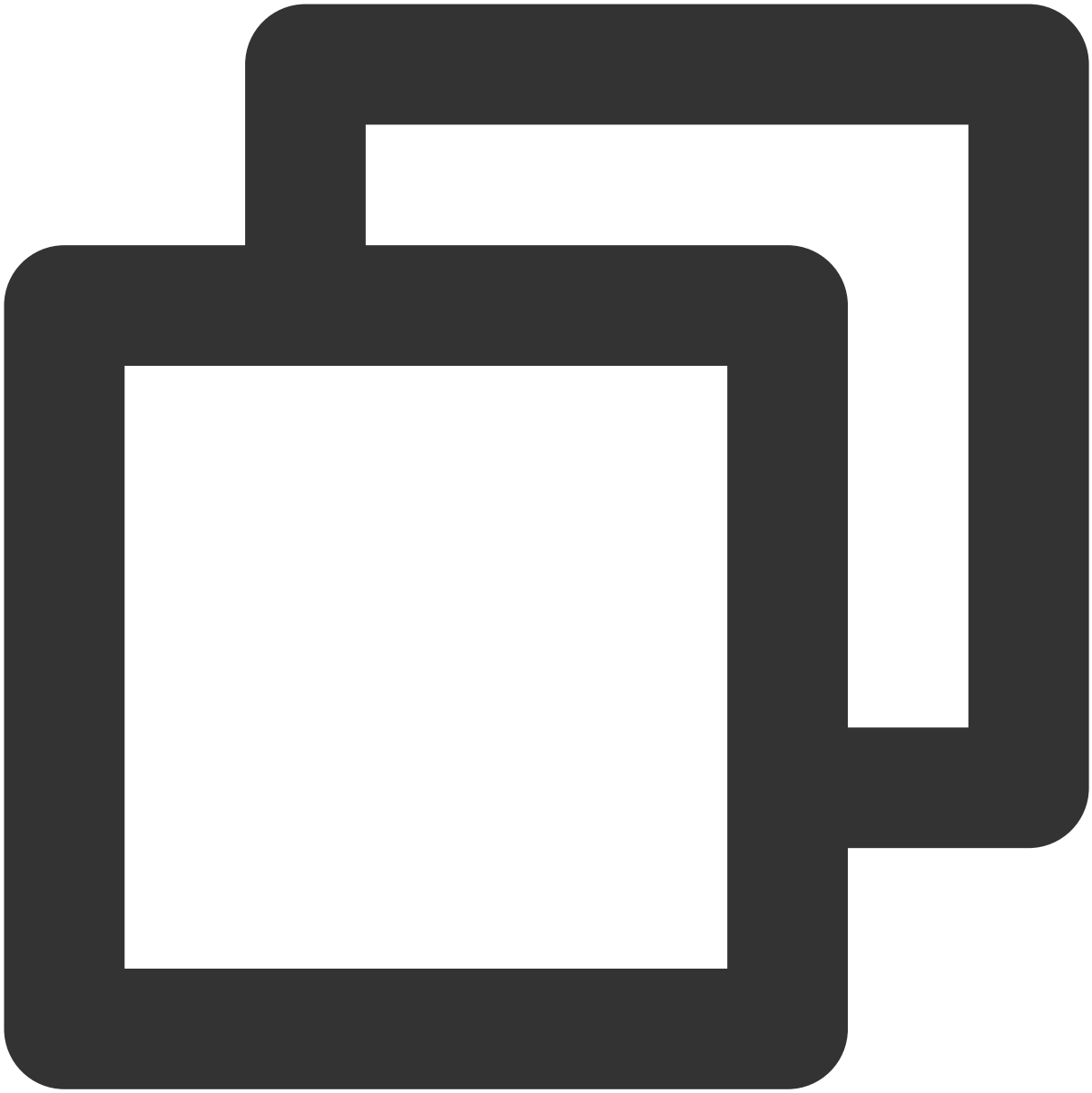
```
def ci_get_doc_queue():
    # 查询文档预览队列信息
    response = client.ci_get_doc_queue(
        Bucket=bucket_name,
        PageNumber=1,
        PageSize=1,
    )
    print(response)
    return response
```

版权所有：腾讯云计算（北京）有限责任公司

参数说明

参数名称	描述	类型
Bucket	队列所在存储桶	String
QueueIds	队列 ID，以“,”符号分割字符串	String
State	1. Active 表示队列内的作业会被文档预览服务调度执行 2. Paused 表示队列暂停，作业不再会被文档预览服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于执行中的任务将继续执行，不受影响	String
PageNumber	第几页	int
PageSize	每页个数	int

返回结果说明



```
{
    'TotalCount': '1',
    'RequestId': 'NjMyMWExY2RfMTIwNjxxxxxxxxxxxxxxxxxx',
    'PageNumber': '1',
    'PageSize': '1',
    'QueueList': [{
        'QueueId': 'p4bdf22746bxxxxxxxxxxxxxxxxxxxxx',
        'Name': 'doc-queue',
        'State': 'Active',
        'NotifyConfig': {
            'Url': 'http://www.demo.callback.com',
```

```
            'Event': 'TaskFinish',
            'Type': 'Url',
            'State': 'On',
            'ResultFormat': 'JSON',
            'MqMode': None,
            'MqName': None,
            'MqRegion': None
        },
        'MaxSize': '10000',
        'MaxConcurrent': '10',
        'CreateTime': '2022-08-11T14:42:01+0800',
        'UpdateTime': '2022-09-14T11:18:08+0800',
        'BucketId': 'test-125xxxxxxxxx',
        'Category': 'DocProcessing'
    }]
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	String
TotalCount	队列总数	Int
PageNumber	当前页数，同请求中的 pageNumber	Int
PageSize	每页个数，同请求中的 pageSize	Int
QueueList	队列数组	Container
NonExistPIDs	不存在的队列 ID 列表	String 数组
QueueId	队列 ID	String
Name	队列名字	String
State	当前状态，Active 或者 Paused	String
NotifyConfig	回调配置	Container
MaxSize	队列最大长度	Int
MaxConcurrent	当前队列最大并行执行的任务数	Int
UpdateTime	更新时间	String
CreateTime	创建时间	String
Url	回调地址	String

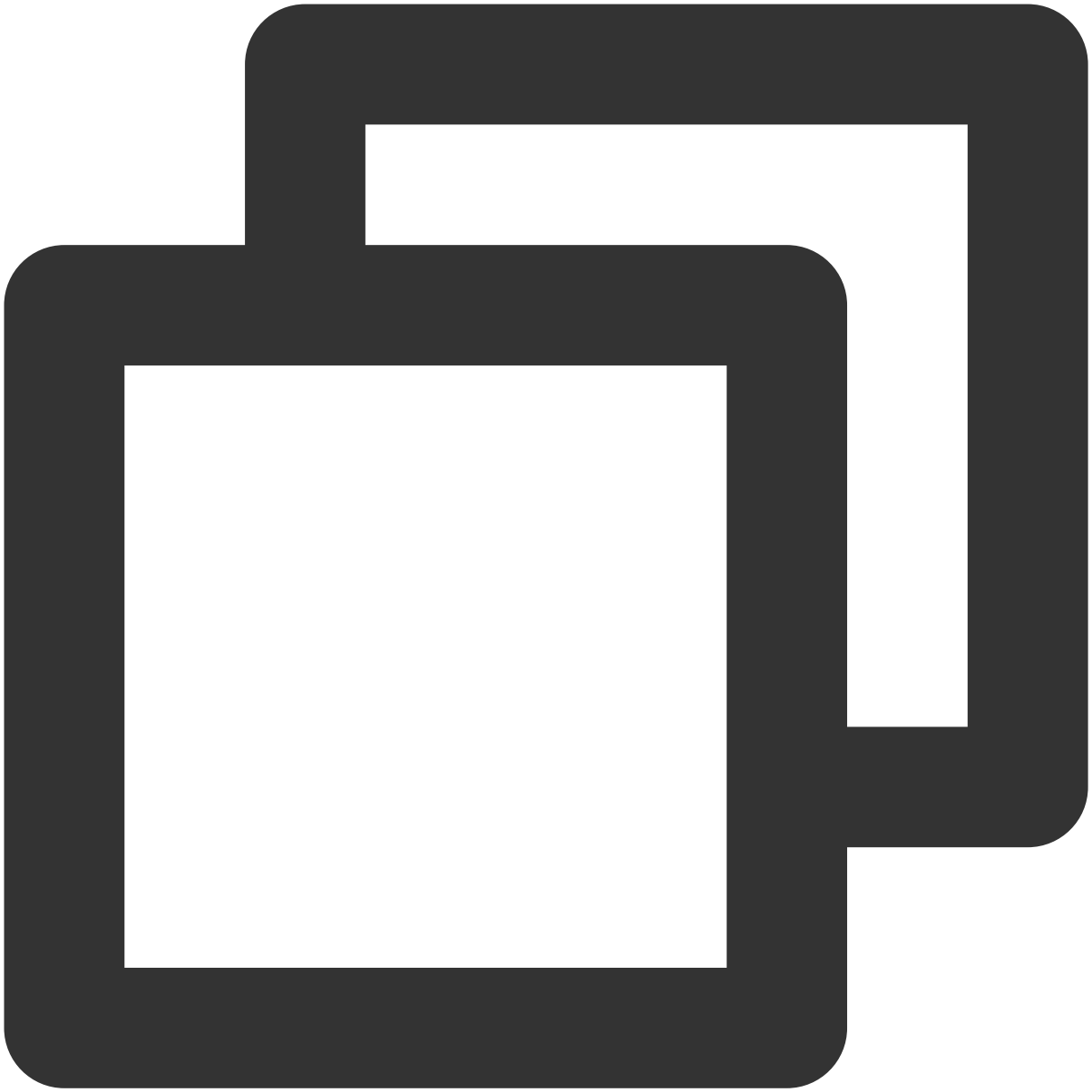
State	开关状态，On 或者 Off	String
Type	回调类型，Url	String
Event	触发回调的事件	String

更新文档预览队列

功能说明

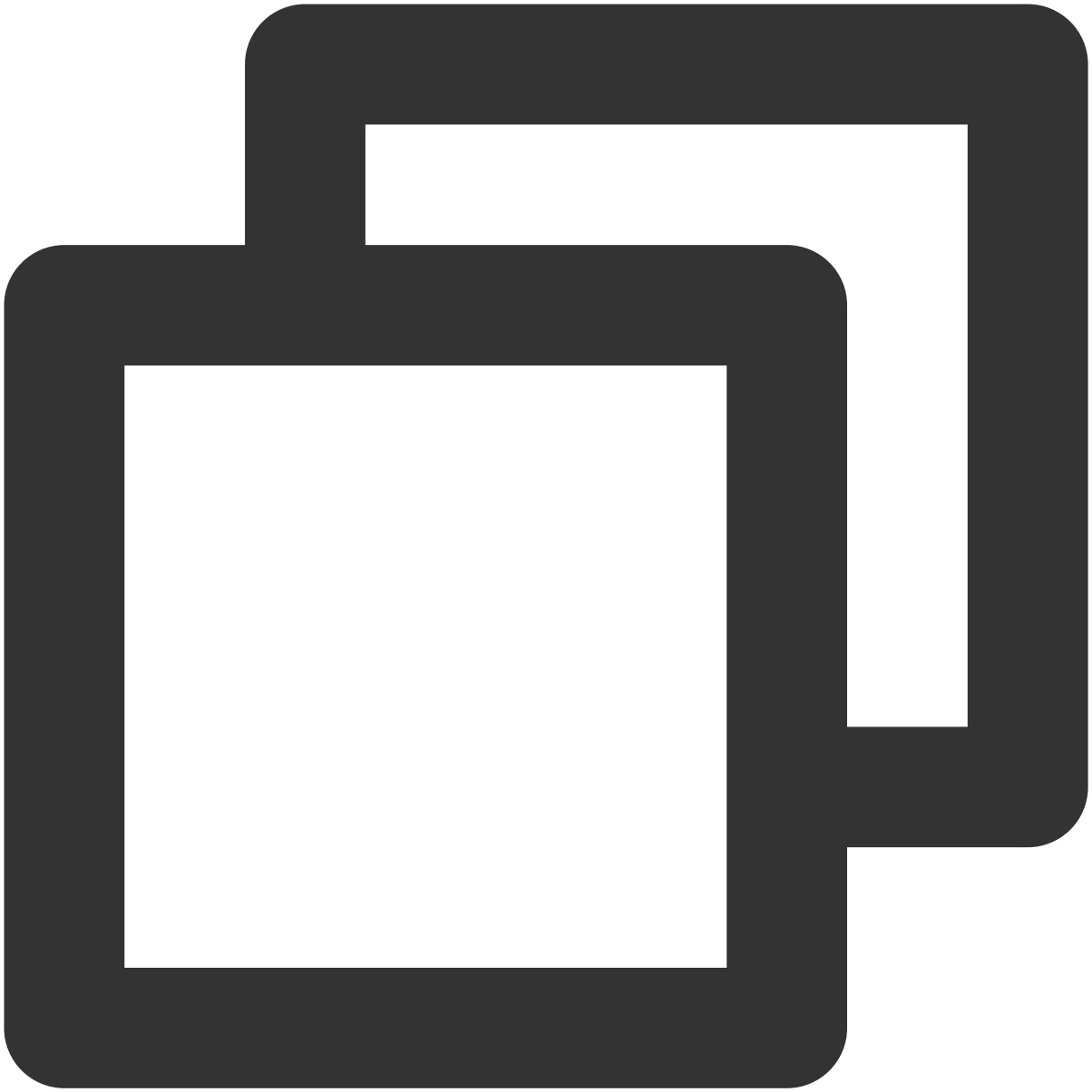
ci_update_doc_queue 接口用于更新文档预览队列。

方法原型



```
def ci_update_doc_queue(self, Bucket, QueueId, Request={}, **kwargs):
```

请求示例



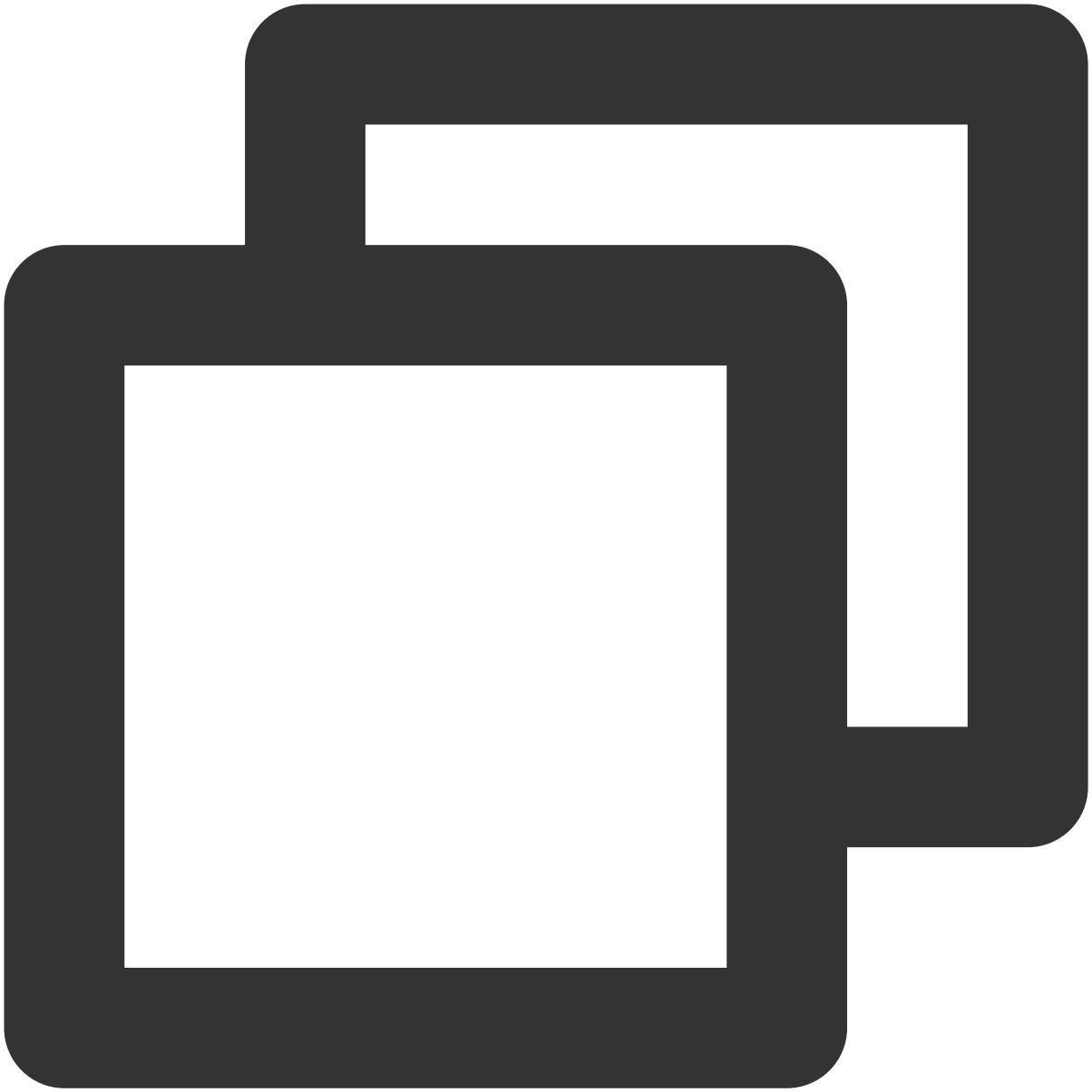
```
def ci_put_doc_queue():
    # 更新文档预览队列信息
    body = {
        'Name': 'doc-queue',
        'QueueID': 'p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxf1',
        'State': 'Active',
        'NotifyConfig': {
            'Type': 'Url',
            'Url': 'http://www.demo.callback.com',
            'Event': 'TaskFinish',
            'State': 'On',
```

```
            'ResultFormat': 'JSON'
        }
    }
    response = client.ci_update_doc_queue(
        Bucket=bucket_name,
        QueueId='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxf1',
        Request=body,
        ContentType='application/xml'
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	存储桶名称	String
Name	队列名称	String
QueueID	队列 ID	String
Request	更新队列配置请求体	String

返回结果说明



```
{
    'RequestId': 'NjMyMWEzMTFfZWMxxxxxxxxxxxxxxxxxxxx',
    'Queue': [{
        'QueueId': 'p4bdf22746b0xxxxxxxxxxxxxxxxx',
        'Name': 'doc-queue',
        'State': 'Active',
        'NotifyConfig': {
            'Url': 'http://www.demo.callback.com',
            'Event': 'TaskFinish',
            'Type': 'Url',
            'State': 'On',
```

```
            'ResultFormat':
            'JSON',
            'MqMode': None,
            'MqName': None,
            'MqRegion': None
        },
        'MaxSize': '10000',
        'Maxconcurrent': '10',
        'CreateTime': '2022-08-11T14:42:01+0800',
        'UpdateTime': '2022-09-14T11:18:08+0800',
        'BucketId': 'test-125xxxxxxxxxx',
        'Category': 'DocProcessing'
    }]
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	dict
Queue	队列信息，详情请参见 查询文档转码队列响应信息中的QueueList	dict

文档转 HTML

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文档预览同步请求的 API 概览以及 SDK 示例代码。

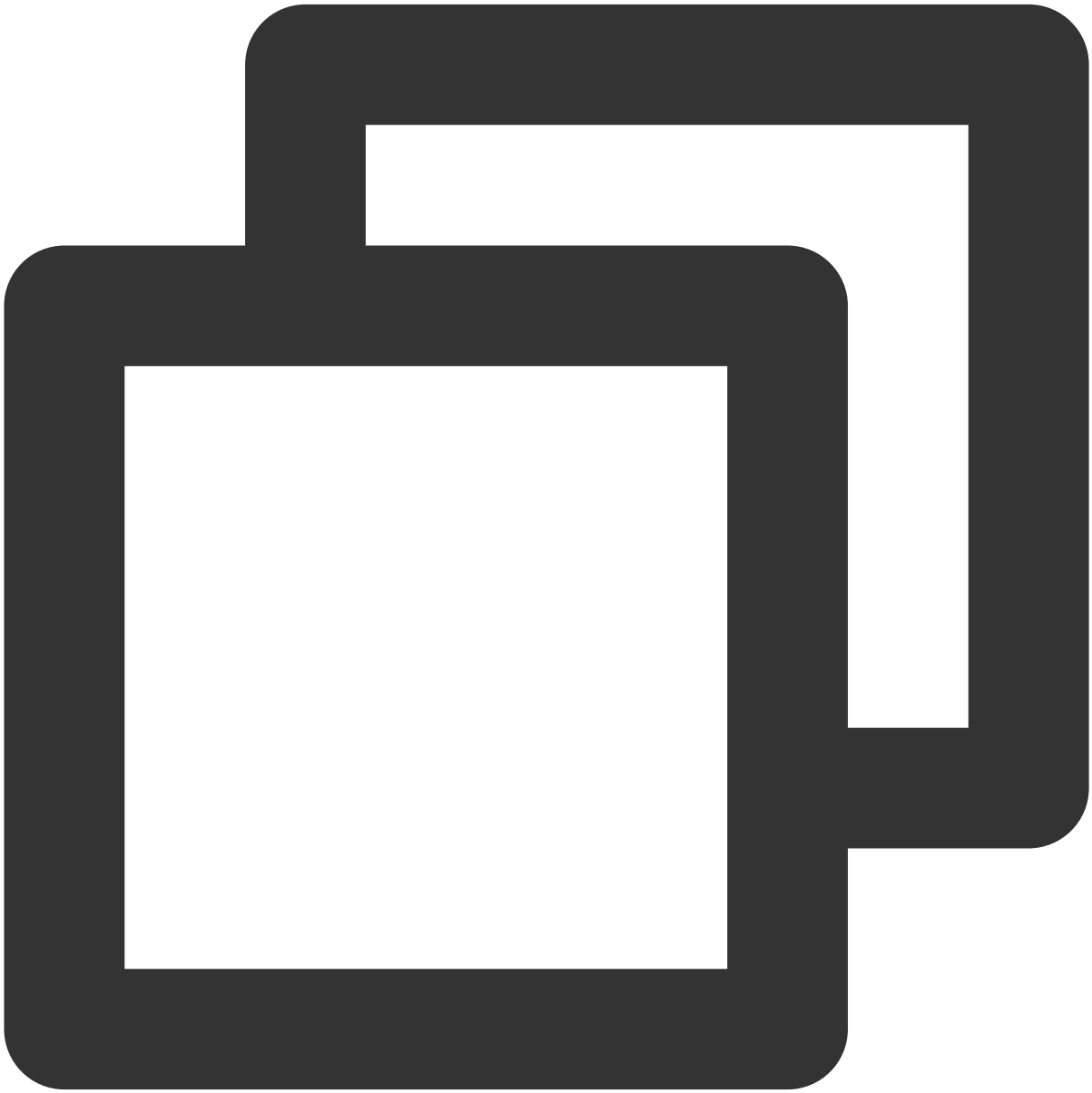
API	说明
文档转 HTML	文档转 HTML 功能支持对多种文档类型的文件生成 HTML 格式预览，满足 PC、App 等多个用户端的文档在线浏览需求，适用于在线教育、企业 OA、在线知识库、网盘文档预览等业务场景。

文档转 HTML

功能说明

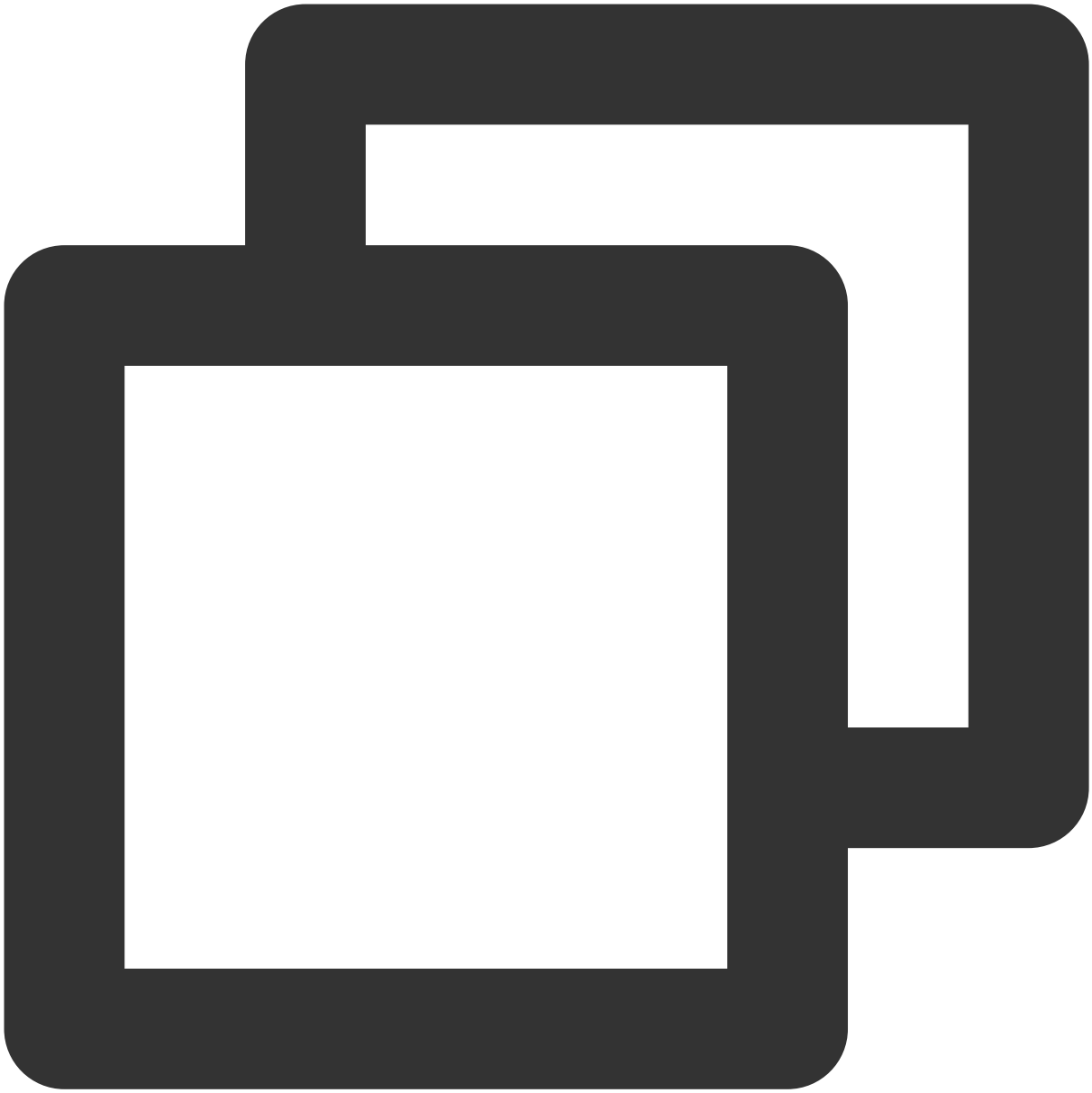
支持对多种文档类型的文件生成 HTML 格式预览。

方法原型



```
def ci_doc_preview_html_process(self, Bucket, Key, SrcType=None, Copyable='1', DstT
                                HtmlFront=None, HtmlRotate=None, HtmlHorizontal
```

请求示例



```
def ci_doc_preview_to_html_process():
    # 文档预览同步接口（生成html）
    response = client.ci_doc_preview_html_process(
        Bucket=bucket_name,
        Key='1.txt',
    )
    print(response)
    response['Body'].get_stream_to_file('result.html')
```

参数说明

名称	参数说明	类型	是否必选
Bucket	对象所在存储桶	String	是
Key	对象名	String	是
SrcType	指定目标文件类型，支持的文件类型请见 文档转 HTML 文档	String	否
Copyable	是否可复制。默认为可复制，填入值为1；不可复制，填入值为0	String	否
DstType	转换输出目标文件类型，文档 HTML 预览固定为 html（需为小写字母）	String	是
HtmlParams	自定义配置参数，json结构，需要经过 URL 安全 的 Base64 编码，默认配置为：{ commonOptions: { isShowTopArea: true, isShowHeader: true } }，支持的配置参考 自定义配置项说明	String	否
HtmlWaterword	水印文字，需要经过 URL 安全 的 Base64 编码，默认为空	String	否
HtmlFillStyle	水印 RGBA（颜色和透明度），需要经过 URL 安全 的 Base64 编码，默认为：rgba(192,192,192,0.6)	String	否
HtmlFront	水印文字样式，需要经过 URL 安全 的 Base64 编码，默认为：bold 20px Serif	String	否
HtmlRotate	水印文字旋转角度，0 - 360，默认315度	String	否
HtmlHorizontal	水印文字水平间距，单位 px，默认为50	String	否
HtmlVertical	水印文字垂直间距，单位 px，默认为100	String	否

版权所有：腾讯云计算（北京）有限责任公司

查询文档预览开通状态

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于查询文档预览开通状态的 API 概览以及 SDK 示例代码。

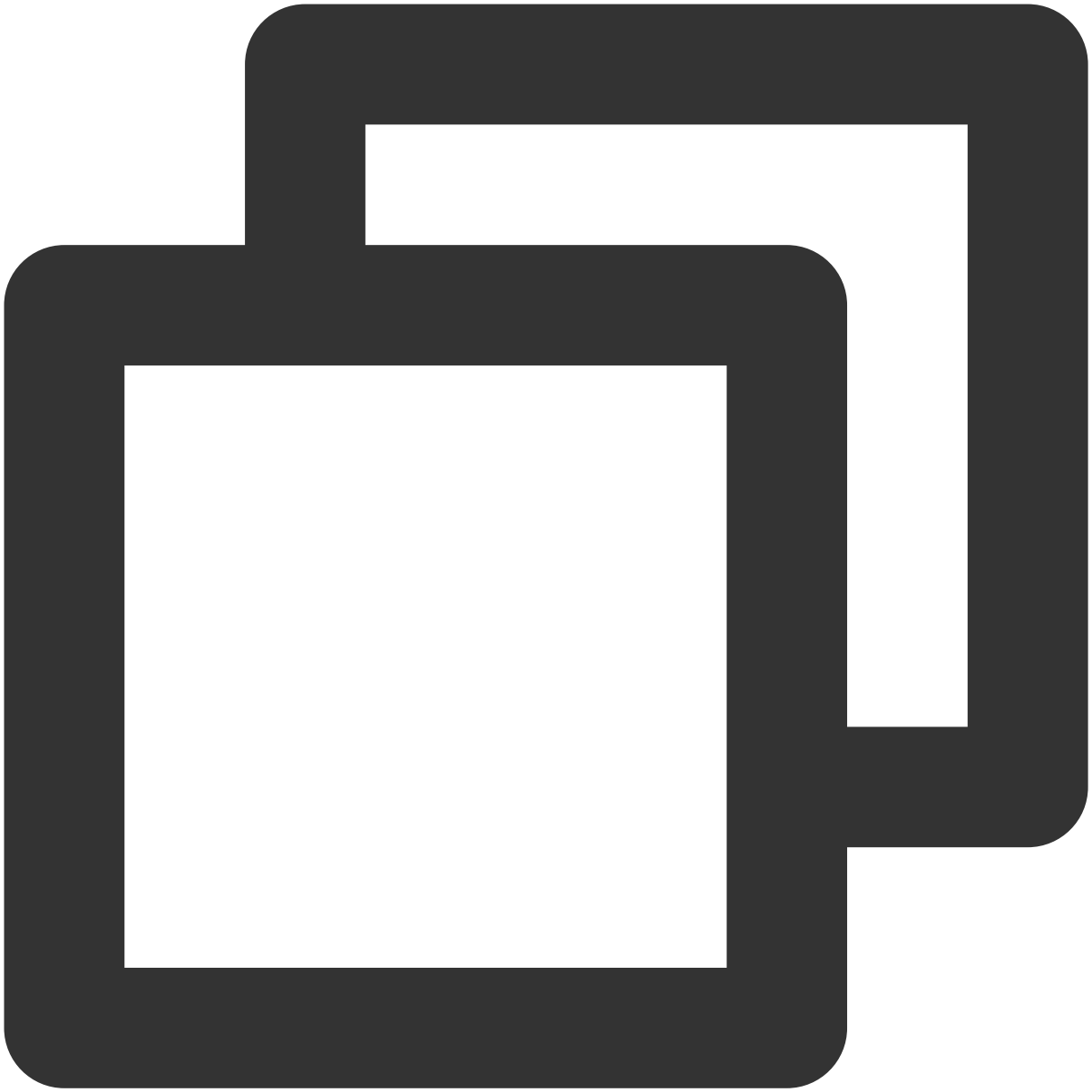
API	说明
查询文档预览开通状态	用于查询存储桶是否已开通文档预览功能

查询文档预览开通状态

功能说明

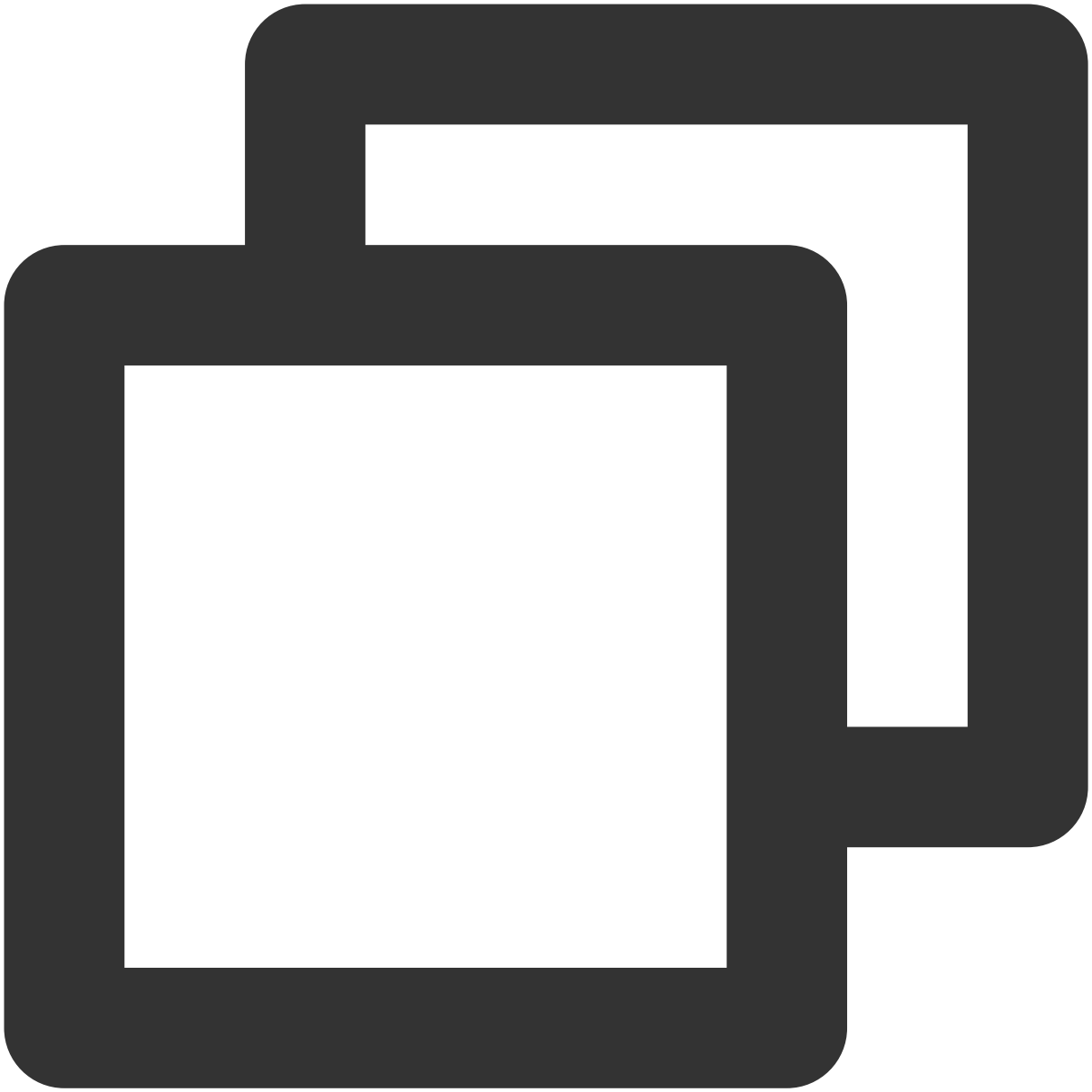
ci_get_doc_bucket 接口用于查询存储桶是否已开通文档预览功能。

方法原型



```
def ci_get_doc_bucket(self, Regions='', BucketName='', BucketNames='', PageNumber='
```

请求示例

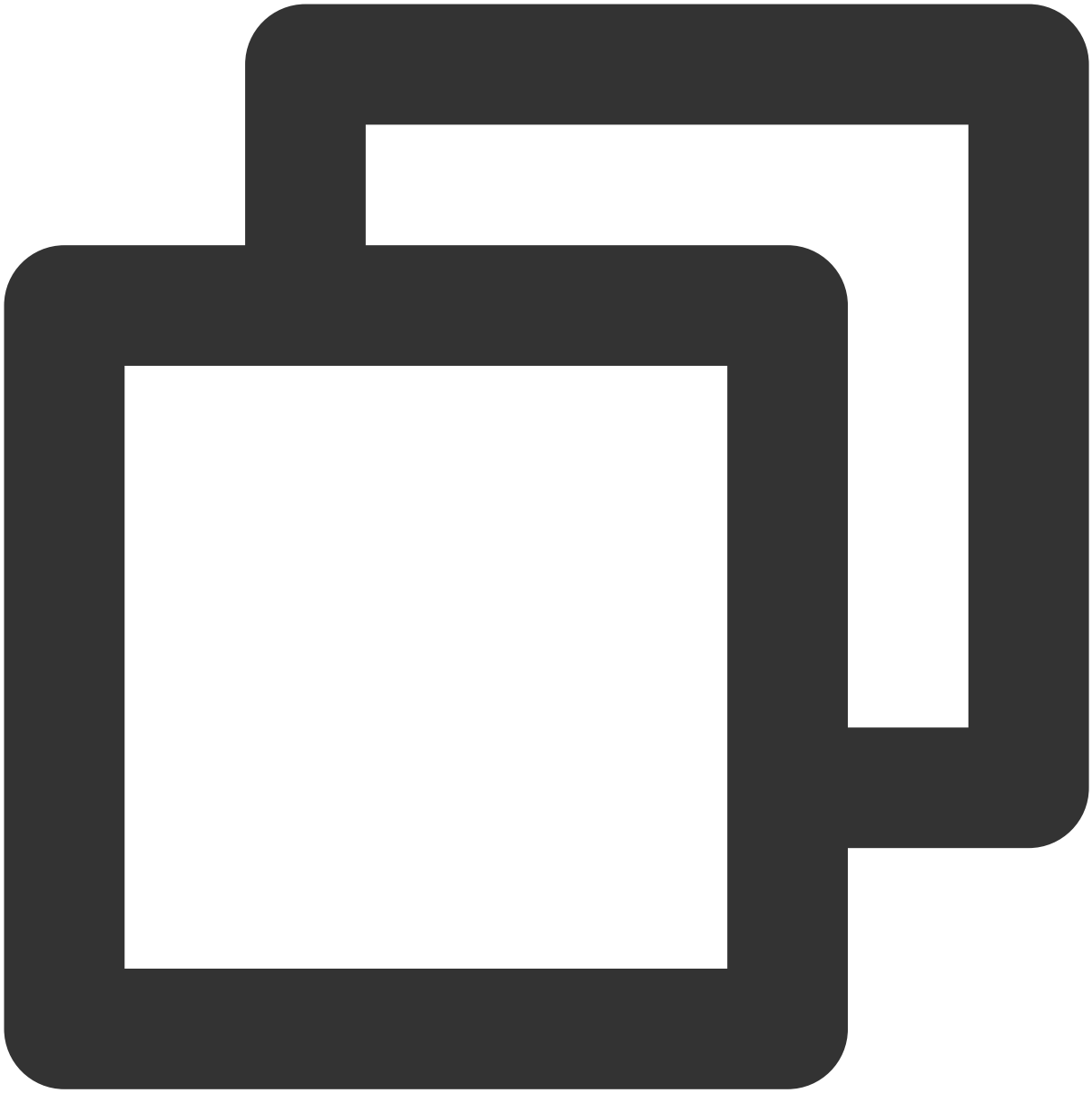


```
def ci_get_doc_bucket():
    # 查询文档预览开通状态
    response = client.ci_get_doc_bucket(
        Regions=region,
        # BucketName='demo',
        BucketNames=bucket_name,
        PageSize=1,
        PageNumber=1
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	string
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	string
BucketName	存储桶名称前缀，前缀搜索	string
PageNumber	第几页	int
pageSize	每页个数	int

返回结果说明



```
{
    'TotalCount': '1',
    'RequestId': 'NjMyMWE2ZjZfZWM0YTYyNjRxxxxxxxxxxxx',
    'PageNumber': '1', 'PageSize': '1',
    'DocBucketList': [{
        'Name': 'testpic-125xxxxxxx',
        'CreateTime': '2022-08-11T00:00:00+0800',
        'Region': 'ap-chongqing',
        'AliasBucketId': None,
        'BucketId': 'test-125xxxxxxxx'
    }]
```

```
}
```

参数名称	描述	类型
RequestId	请求的唯一 ID	string
TotalCount	文档预览 Bucket 总数	int
PageNumber	当前页数，同请求中的 pageNumber	int
PageSize	每页个数，同请求中的 pageSize	int
DocBucketList	文档预览 Bucket 列表	Container
BucketId	存储桶 ID	string
Name	存储桶名称，同 BucketId	string
Region	所在的地域	string
CreateTime	创建时间	string
AliasBucketId	存储桶别名	string

模板接口
语音识别

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于数据万象语音识别模板SDK使用方式。

API	操作名	操作描述
创建语音识别模板	创建模板	用于创建一个新的模板
删除语音识别模板	删除模板	删除一个模板
查询语音识别模板	查询模板	查询模板列表
更新语音识别模板	修改模板	修改一个模板

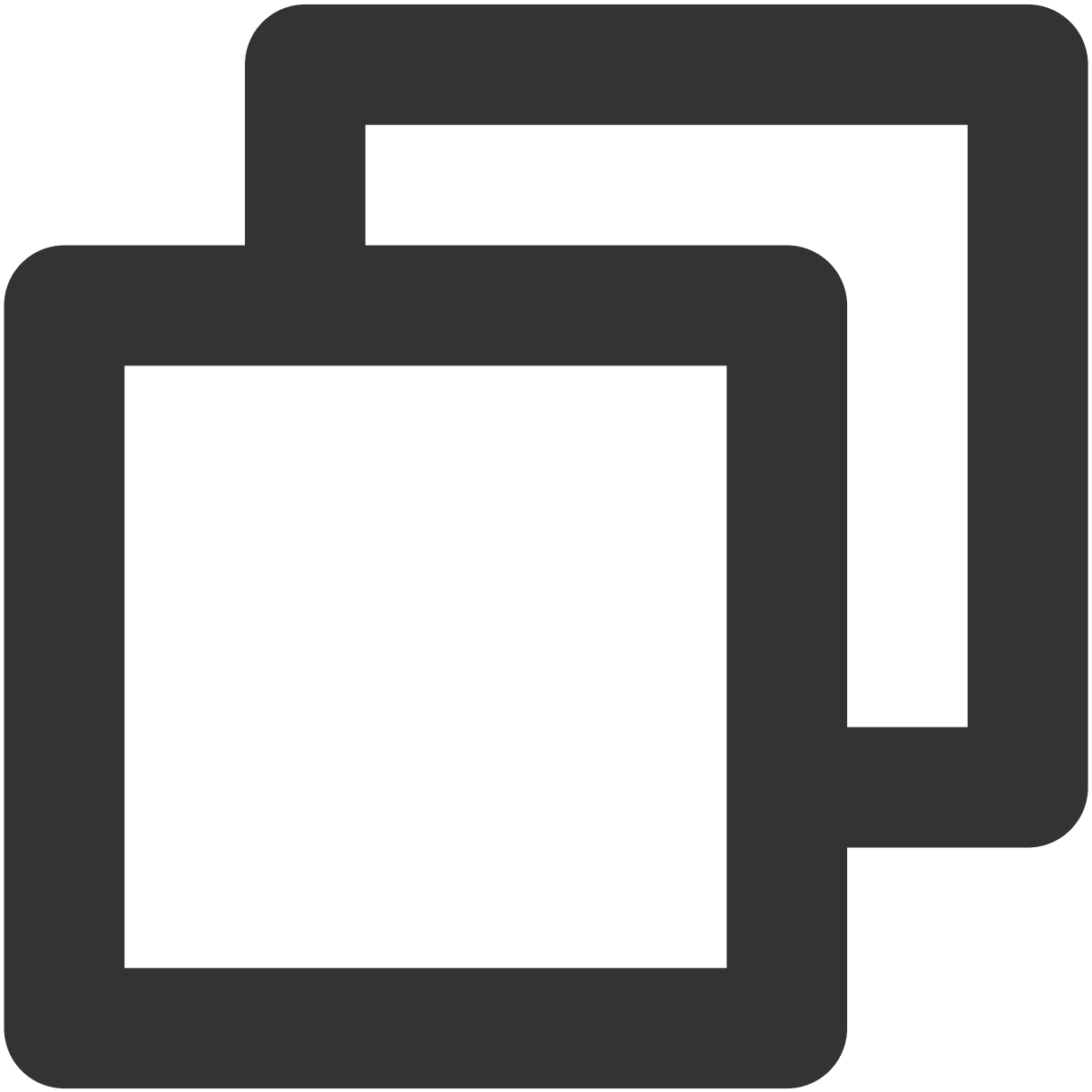
基本操作

创建模板

功能说明

用于创建一个新的模板。

方法原型



```
def ci_create_asr_template(self, Bucket, Name, EngineModelType, ChannelNum,
                           ResTextFormat, FilterDirty=0, FilterModal=0, ConvertNumM
                           SpeakerNumber=0, FilterPunc=0, OutputFileType='txt', **k
```

参数说明

Request 的具体参数描述如下：

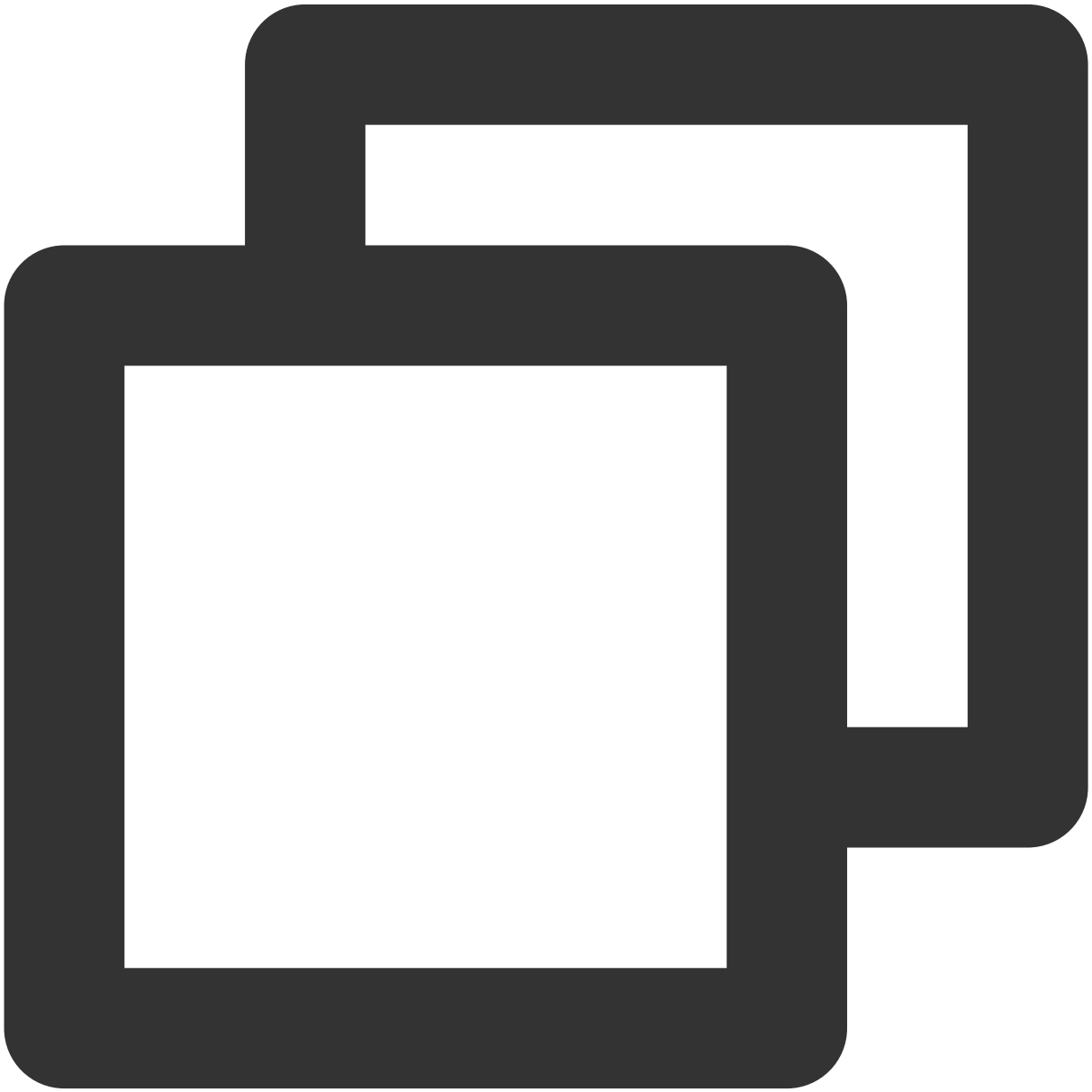
节点名称（关键字）	描述	类型	必选

Bucket	存储桶名称	String	是
Name	模板名称仅支持中文、英文、数字、_、-和*	String	是
EngineModelType	引擎模型类型，分为电话场景和非电话场景。 电话场景： 8k_zh：电话 8k 中文普通话通用（可用于双声道音频）； 8k_zh_s：电话 8k 中文普通话话者分离（仅适用于单声道音频）； 8k_en：电话 8k 英语； 非电话场景： 16k_zh：16k 中文普通话通用； 16k_zh_video：16k 音视频领域； 16k_en：16k 英语； 16k_ca：16k 粤语； 16k_ja：16k 日语； 16k_zh_edu：中文教育； 16k_en_edu：英文教育； 16k_zh_medical：医疗； 16k_th：泰语； 16k_zh_dialect：多方言，支持23种方言。	String	是
ChannelNum	语音声道数： 1 表示单声道.EngineModelType为非电话场景仅支持单声道； 2 表示双声道（仅支持 8k_zh 引擎模型 双声道应分别对应通话双方）。	int	否
ResTextFormat	识别结果返回形式： 0 表示识别结果文本（含分段时间戳）； 1 词级别粒度的详细识别结果,不含标点，含语速值.（词时间戳列表，一般用于生成字幕场景） 2 词级别粒度的详细识别结果（包含标点、语速值）。	int	是
FilterDirty	是否过滤脏词（目前支持中文普通话引擎）： 0 表示不过滤脏词； 1 表示过滤脏词； 2 表示将脏词替换为 *。 默认值为 0。	int	否
FilterModal	是否过语气词（目前支持中文普通话引擎）： 0 表示不过滤语气词； 1 表示部分过滤； 2 表示严格过滤 。 默认值为 0。	int	否
ConvertNumMode	是否进行阿拉伯数字智能转换（目前支持中文普通话引擎）： 0 表示不转换，直接输出中文数字；	int	否

版权所有：腾讯云计算（北京）有限责任公司

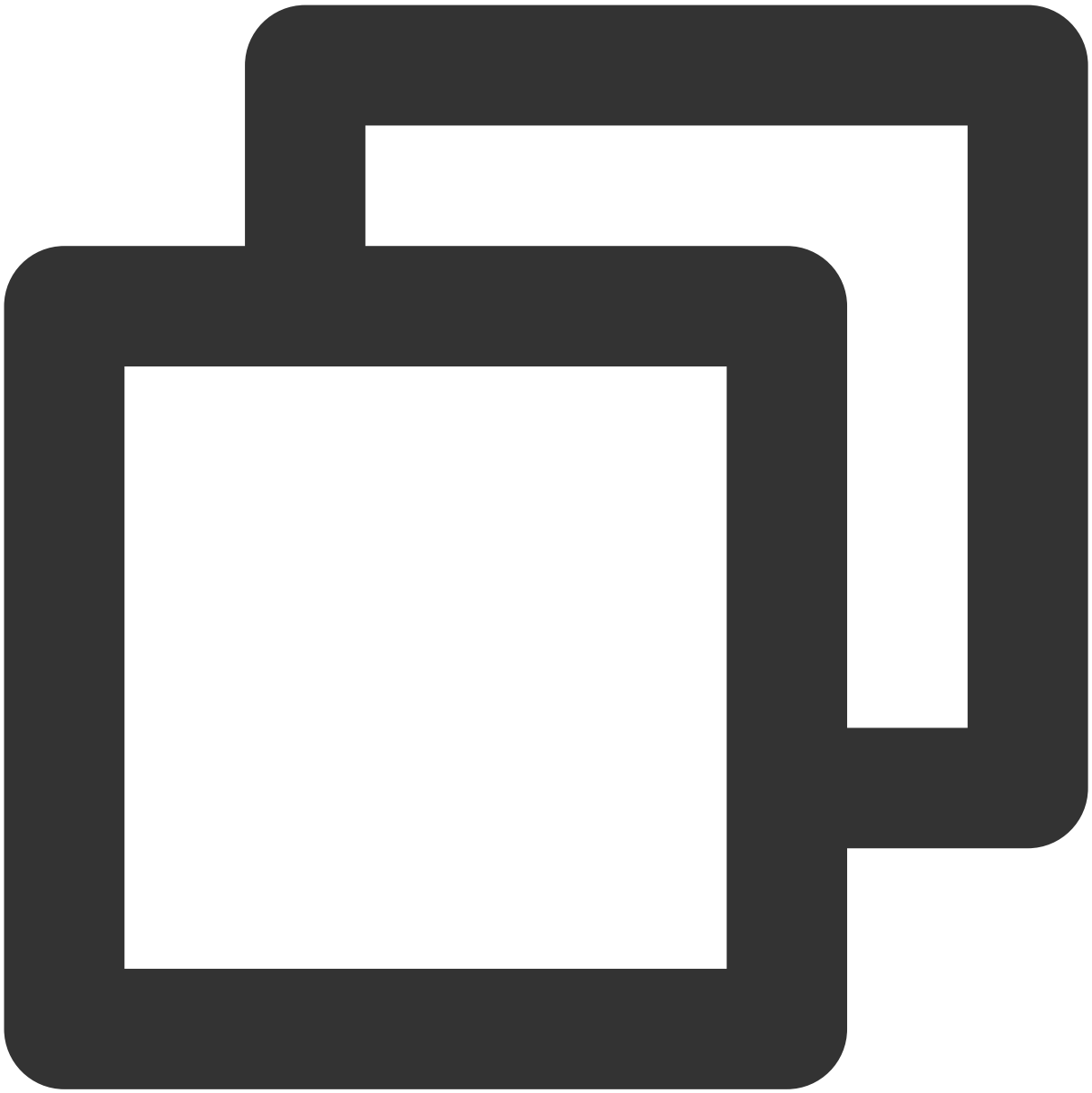
	1 表示根据场景智能转换为阿拉伯数字。 3 表示打开数学相关数字转换。 默认值为 0。		
SpeakerDiarization	是否开启说话人分离： 0 表示不开启； 1 表示开启(仅支持8k_zh，16k_zh，16k_zh_video，单声道音频)。 默认值为 0。 注意：8k电话场景建议使用双声道来区分通话双方，设置ChannelNum=2即可，不用开启说话人分离。	int	否
SpeakerNumber	说话人分离人数（需配合开启说话人分离使用），取值范围：0-10。 0代表自动分离（目前仅支持≤6个人），1-10代表指定说话人数分离。默认值为 0。	int	否
FilterPunc	是否过滤标点符号（目前支持中文普通话引擎）： 0 表示不过滤。 1 表示过滤句末标点。 2 表示过滤所有标点。 默认值为 0。	int	否
OutputFileType	输出文件类型，可选txt、srt。默认为txt	String	否

请求示例



```
def ci_create_asr_template():
    # 创建语音识别模板
    response = client.ci_create_asr_template(
        Bucket=bucket_name,
        Name='templateName',
        EngineModelType='16k_zh',
        ChannelNum=1,
        ResTextFormat=2,
    )
    print(response)
    return response
```

返回结果说明



```
{
    'RequestId': 'NjMyMjliMWZfZWM0YTYyNjRfNWNmNF8xMDBh',
    'Template': {
        'TemplateId': 't1c1287c04c147443da0b2cc7b8fbabf32',
        'Name': 'templateName',
        'State': 'Normal',
        'Tag': 'SpeechRecognition',
        'CreateTime': '2022-09-15T11:25:19+0800',
```

```
            'UpdateTime': '2022-09-15T11:25:19+0800',
            'BucketId': 'testpic-1253960454',
            'Category': 'Custom',
            'SpeechRecognition': {
                'EngineModelType': '16k_zh',
                'ChannelNum': '1',
                'ResTextFormat': '2',
                'FilterDirty': '0',
                'FilterModal': '0',
                'ConvertNumMode': '0',
                'SpeakerDiarization': '0',
                'SpeakerNumber': '0',
                'FilterPunc': '0',
                'OutputFileType': 'txt'
            }
        }
    }
```

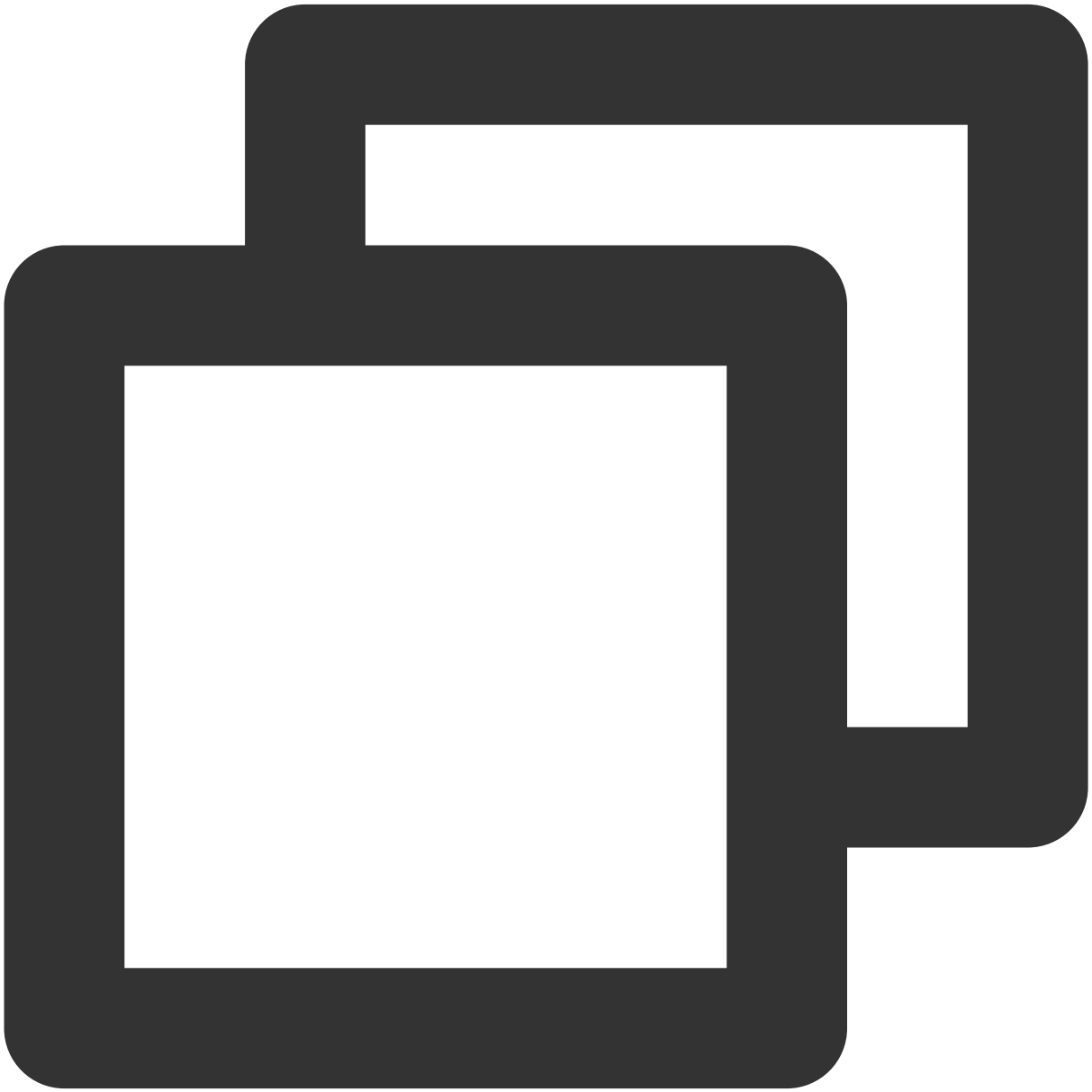
响应字段详情详见 创建语音识别模板响应信息。

删除模板

功能说明

删除一个模板

方法原型

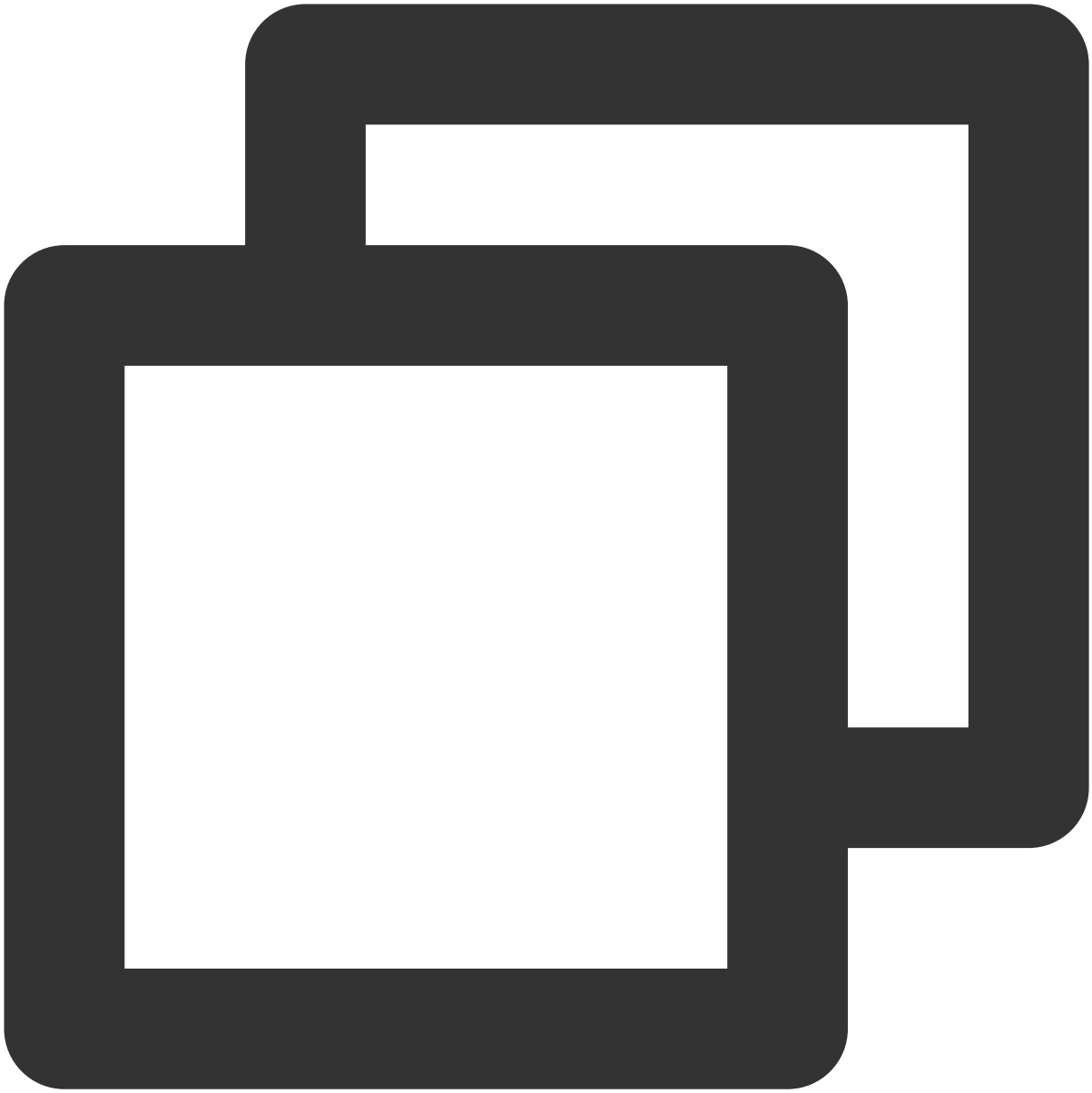


```
def ci_delete_asr_template(self, Bucket, TemplateId, **kwargs)
```

参数说明

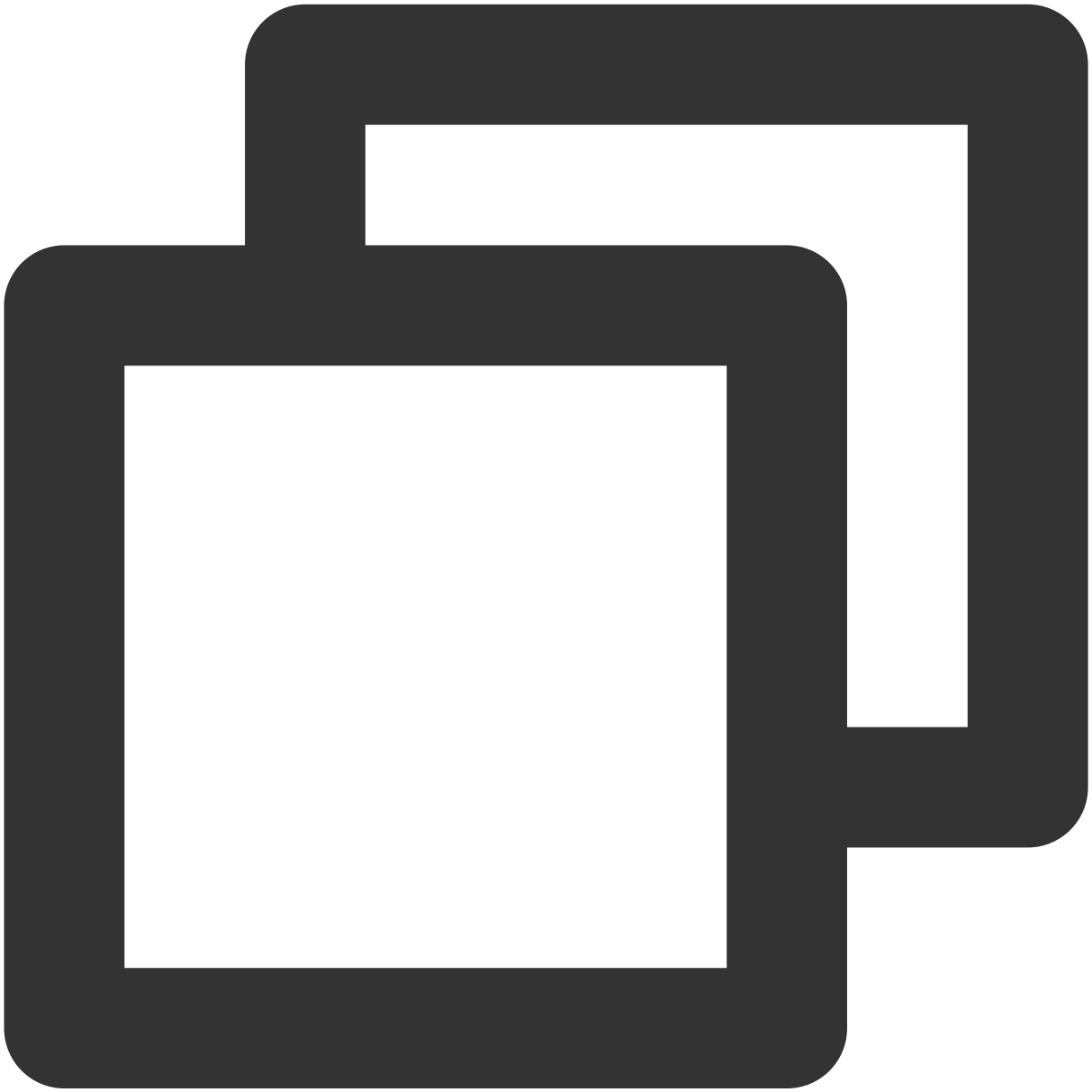
参数名称	描述	类型	必选
Bucket	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
TemplateId	要取消的模板 id	String	是

请求示例



```
def ci_delete_asr_template():
    # 删除指定语音识别模板
    response = client.ci_delete_asr_template(
        Bucket=bucket_name,
        TemplateId='t1bdxxxxxxxxxxxxxxxx94a9',
    )
    print(response)
    return response
```

返回结果说明



```
{
    'RequestId': 'NjMyMjlkZmRfZWM0YTYyNjRfNWNmNF8xMDBi',
    'TemplateId': 't1c1287c04c147443da0b2cc7b8fbabf32'
}
```

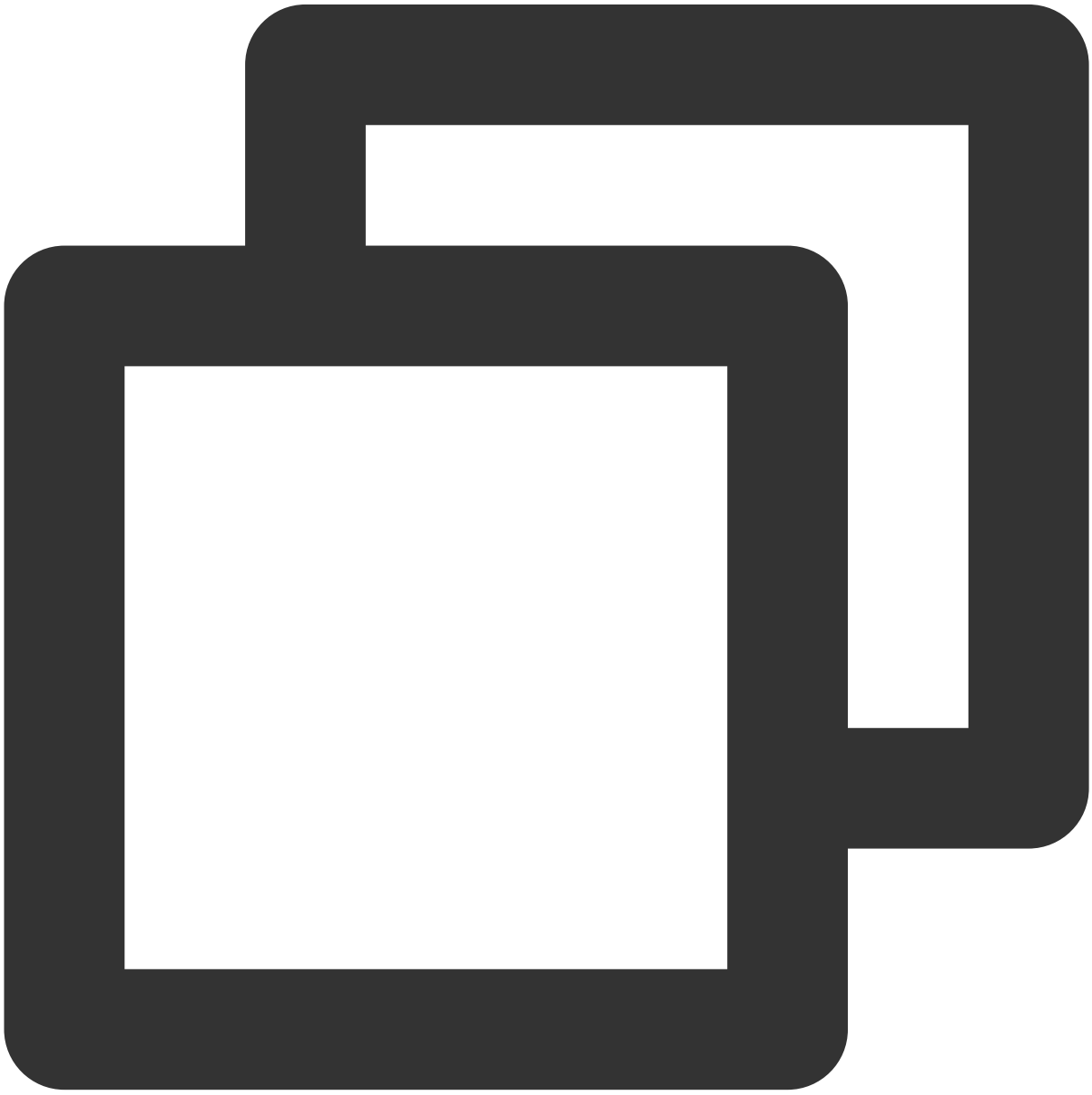
响应字段详情详见DeleteTemplate响应信息

查询模板列表

功能说明

查询模板列表

方法原型



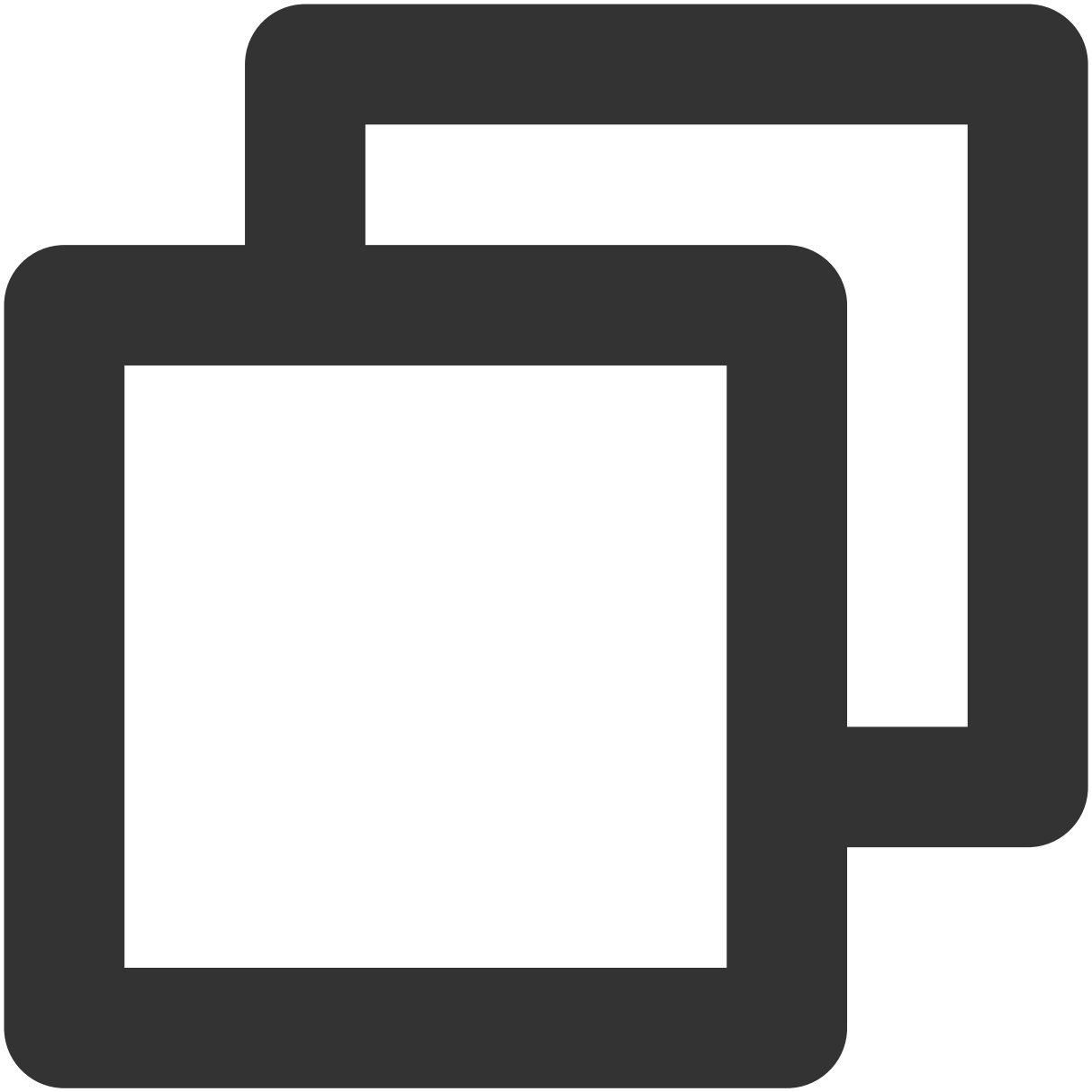
```
def ci_get_asr_template(self, Bucket, Category='Custom', Ids='', Name='', PageNumbe
```

参数说明

参数名称	描述	类型	必选

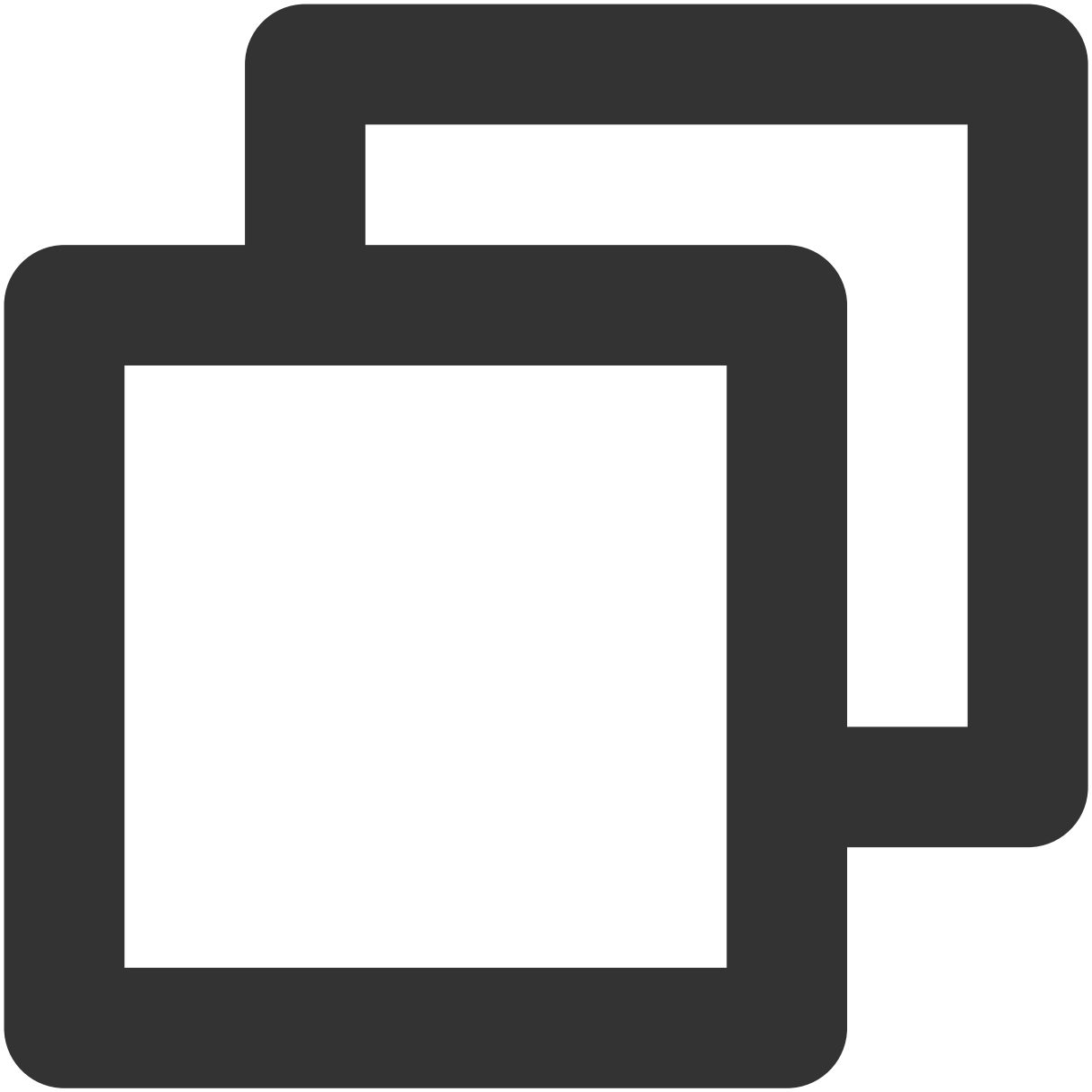
Bucket	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
Category	Official，Custom，默认值：Custom	String	否
Ids	模板 ID，以 `,` 符号分割字符串	String	否
Name	模板名称前缀	String	否
PageNumber	第几页	Integer	否
PageSize	每页个数	Integer	否

请求示例



```
def ci_get_asr_template():
    # 获取语音识别模板
    response = client.ci_get_asr_template(
        Bucket=bucket_name,
    )
    print(response)
    return response
```

返回结果说明




```
{
    'TotalCount': '1',
    'RequestId': 'NjMyMjljNTlfMTIwNjUzMDlfMmUzYV8xMWNh',
    'PageNumber': '1',
    'PageSize': '10',
    'TemplateList': [
        {
            'TemplateId': 't1c1287c04c147443da0b2cc7b8fbabf32',
            'Name': 'templateName',
            'State': 'Normal',
            'Tag': 'SpeechRecognition',
```

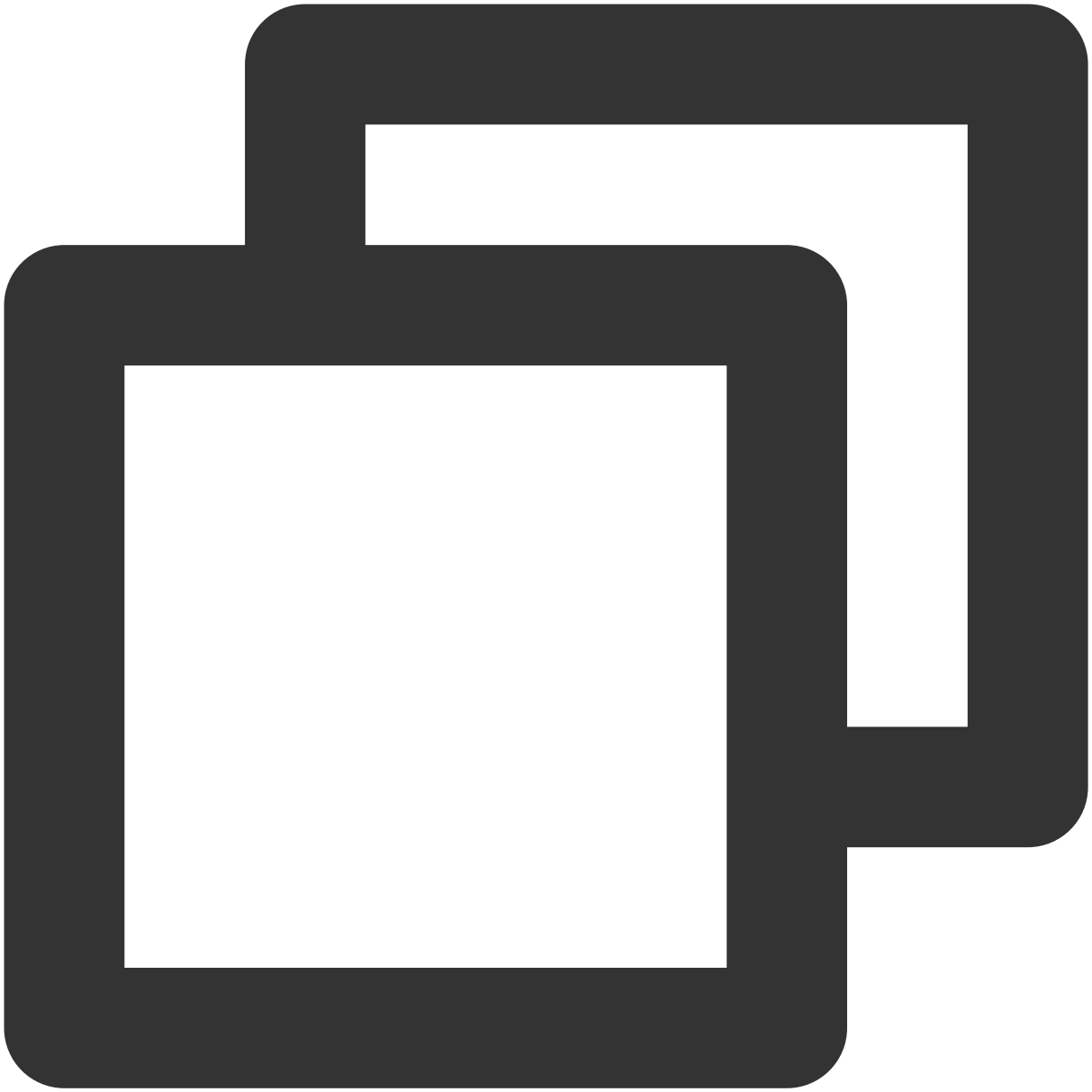
```
            'CreateTime': '2022-09-15T11:25:19+0800',
            'UpdateTime': '2022-09-15T11:25:19+0800',
            'BucketId': 'testpic-1253960454',
            'Category': 'Custom',
            'SpeechRecognition': {
                'EngineModelType': '16k_zh',
                'ChannelNum': '1',
                'ResTextFormat': '2',
                'FilterDirty': '0',
                'FilterModal': '0',
                'ConvertNumMode': '0',
                'SpeakerDiarization': '0',
                'SpeakerNumber': '0',
                'FilterPunc': '0',
                'OutputFileType': 'txt'
            }
        }
    ]
}
```

修改模板

功能说明

修改模板

方法原型



```
def ci_update_asr_template(self, Bucket, TemplateId, Name, EngineModelType, Channel
                           ResTextFormat, FilterDirty=0, FilterModal=0, ConvertNumM
                           SpeakerNumber=0, FilterPunc=0, OutputFileType='txt', **k
```

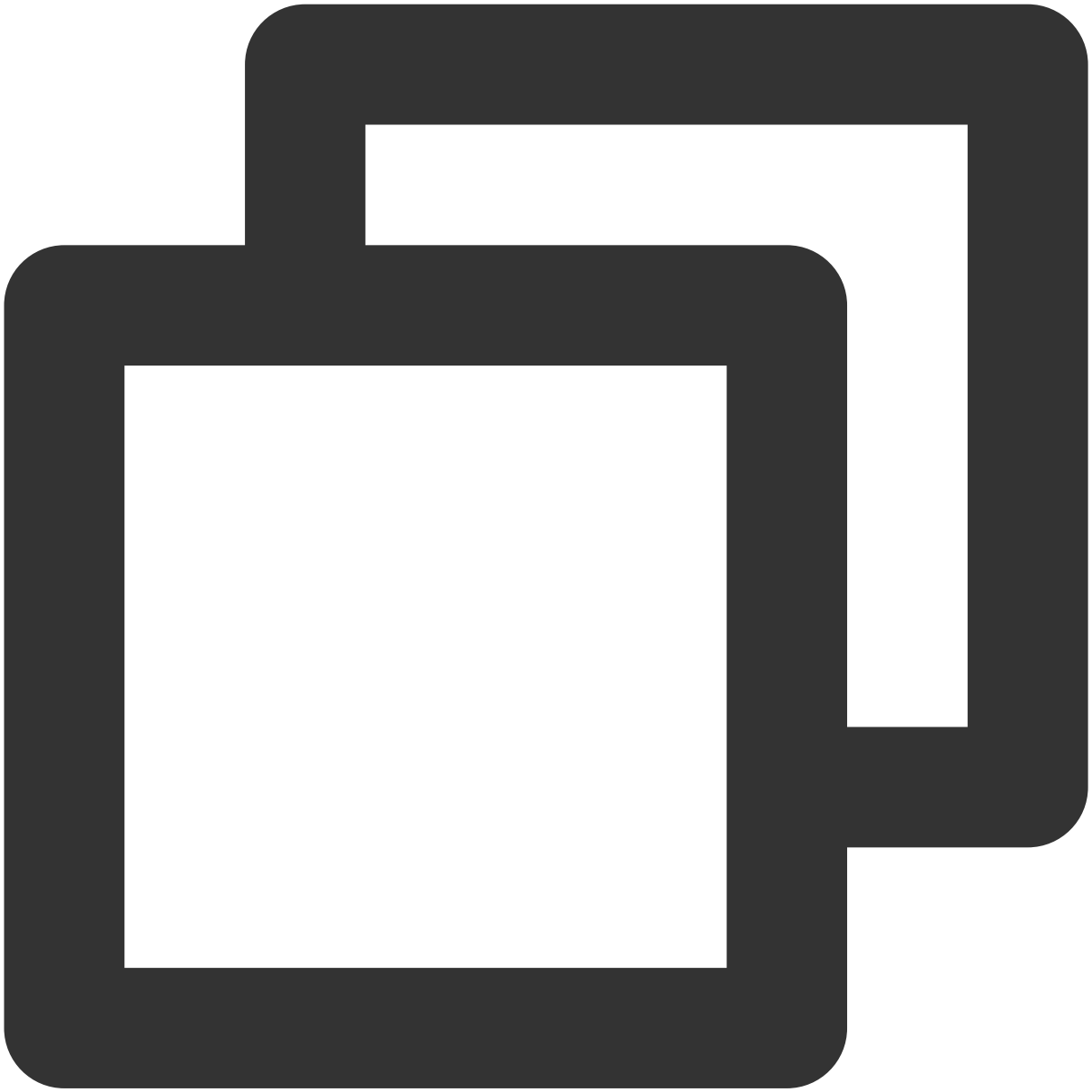
参数说明

节点名称（关键字）	描述	类型	必选
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概	String	是

	述		
templateId	修改的模板 ID	String	是

说明：
其余参数请参见 创建模板接口，与其参数一致。

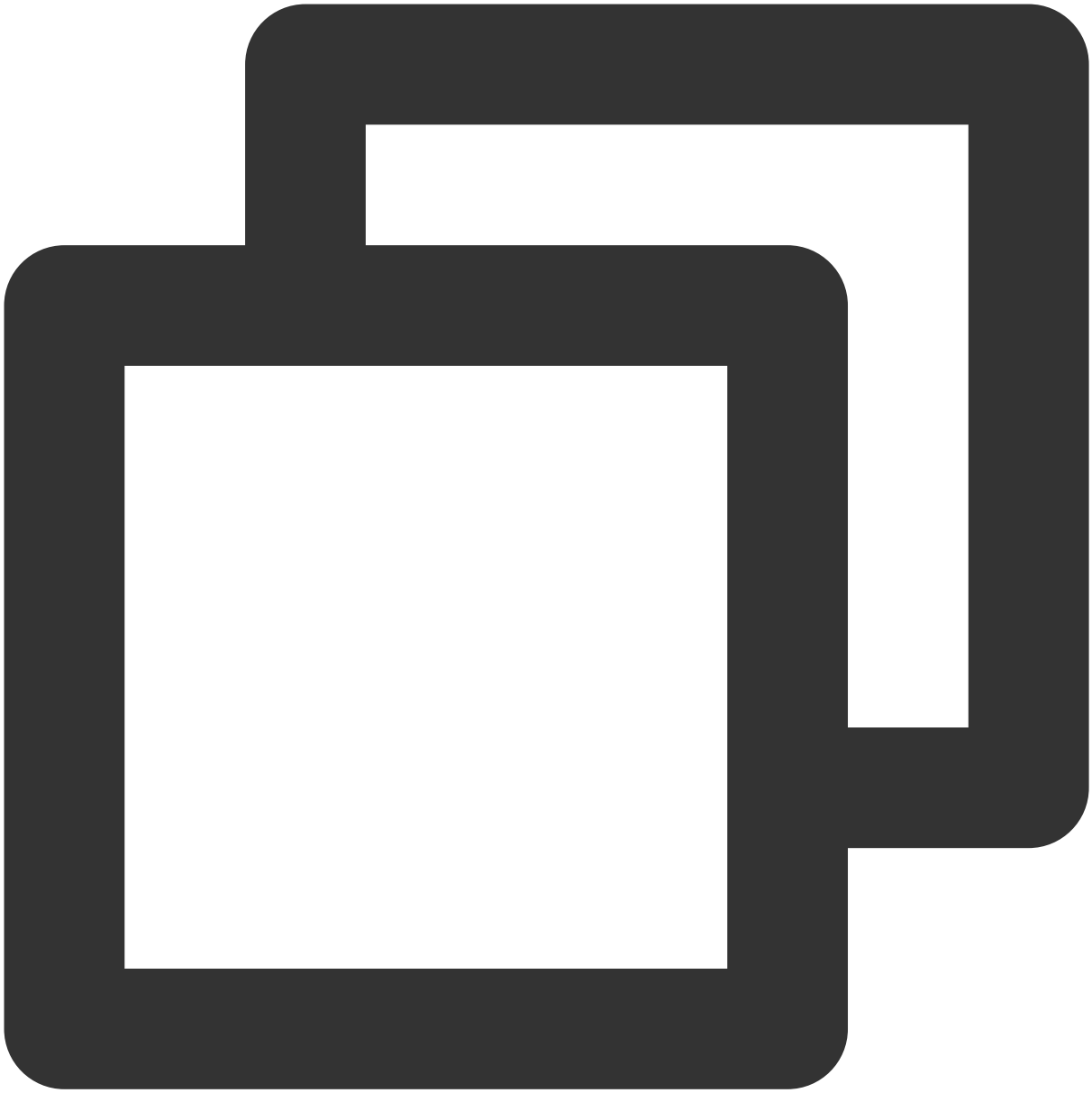
请求示例



```
def ci_update_asr_template():
```

```
    # 修改语音识别模板
    response = client.ci_update_asr_template(
        Bucket=bucket_name,
        TemplateId='t1bdxxxxxxxxxxxxxxxx94a9',
        Name='QueueId1',
        EngineModelType='16k_zh',
        ChannelNum=1,
        ResTextFormat=1,
    )
    print(response)
    return response
```

返回结果说明




```
{
    'RequestId': 'NjMyMjlkNzhfMTIwNjUzMDlfMmUxZF8xMGM4',
    'Template': {
        'TemplateId': 't1c1287c04c147443da0b2cc7b8fbabf32',
        'Name': 'QueueId1',
        'State': 'Normal',
        'Tag': 'SpeechRecognition',
        'CreateTime': '2022-09-15T11:25:19+0800',
        'UpdateTime': '2022-09-15T11:35:20+0800',
        'BucketId': 'testpic-1253960454',
        'Category': 'Custom',
```


版权所有：腾讯云计算（北京）有限责任公司

```
        'SpeechRecognition': {
            'EngineModelType': '16k_zh',
            'ChannelNum': '1',
            'ResTextFormat': '1',
            'FilterDirty': '0',
            'FilterModal': '0',
            'ConvertNumMode': '0',
            'SpeakerDiarization': '0',
            'SpeakerNumber': '0',
            'FilterPunc': '0',
            'OutputFileType': 'txt'
        }
    }
}
```

响应字段详情详见 更新语音识别模板响应信息。

文件处理
文件解压
查询文件解压结果

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文件处理查询文件解压结果请求的 API 概览以及 SDK 示例代码。

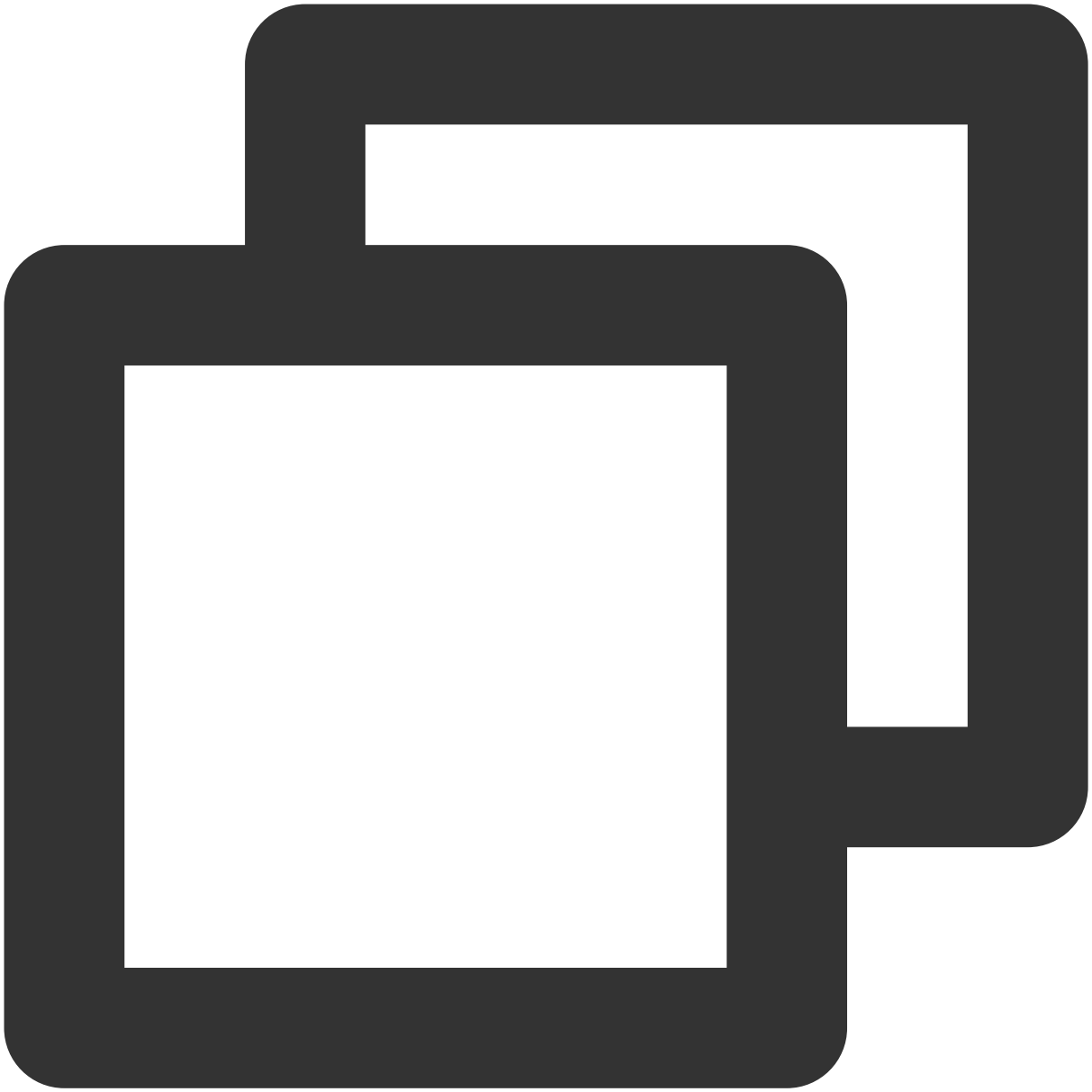
API	说明
查询文件解压结果请求接口	用于查询异步查询文件解压结果

异步请求接口

功能说明

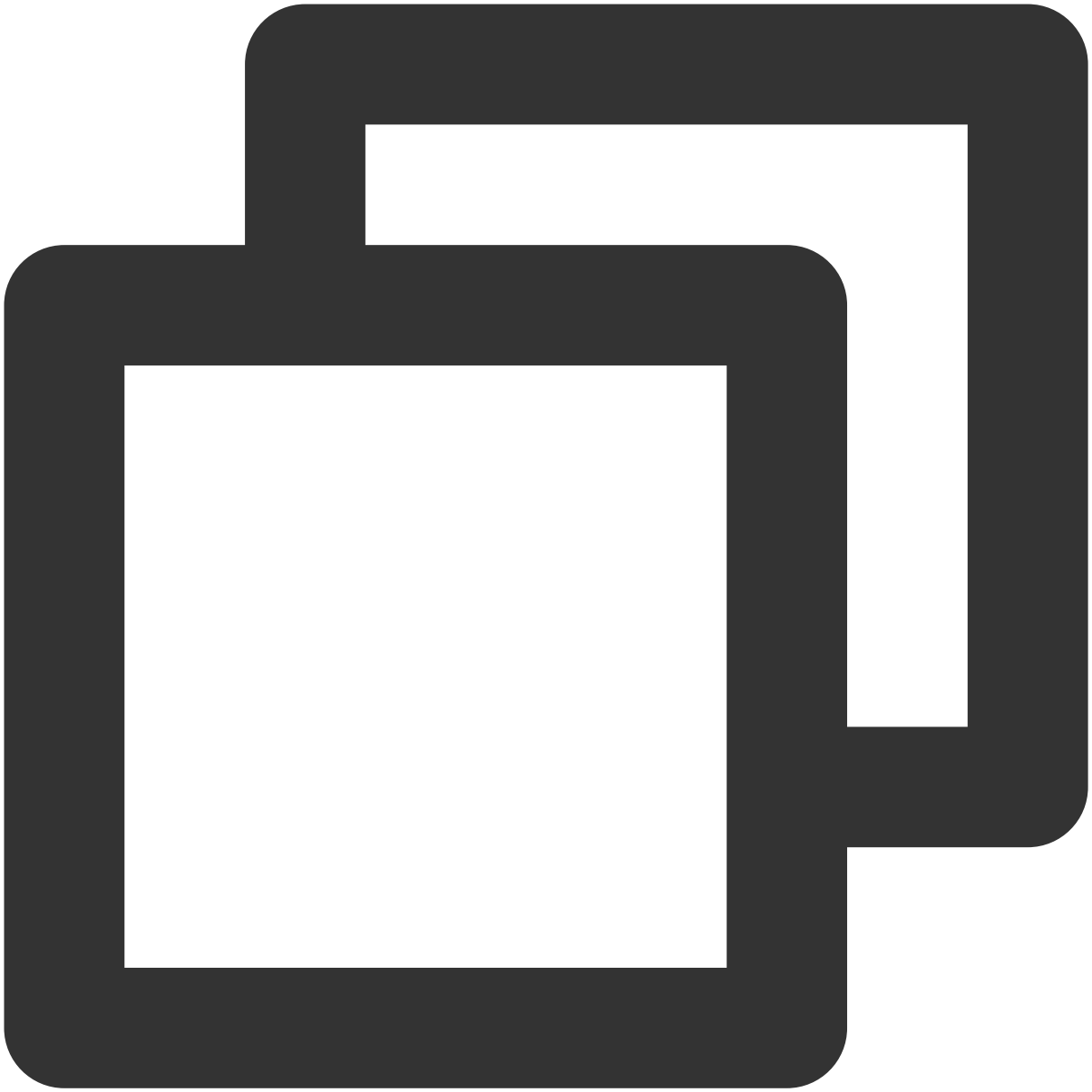
查询异步文件处理查询文件解压结果请求。

方法原型



```
def ci_get_file_process_jobs(self, Bucket, JobIDs, **kwargs)
```

请求示例

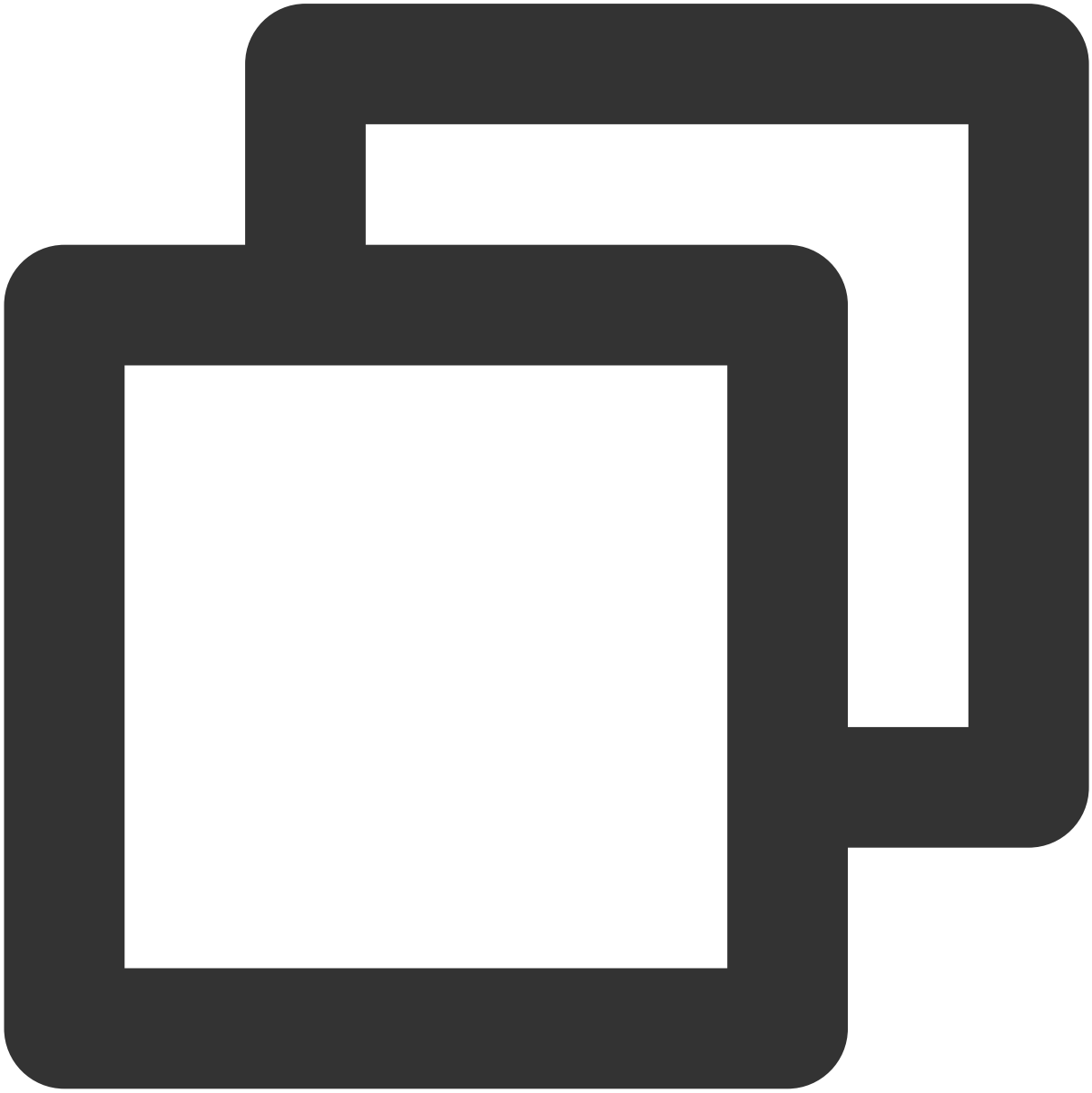


```
    def ci_get_file_process_jobs():
        response = client.ci_get_file_process_jobs(
            Bucket=bucket_name,
            JobIDs='fcb9a9f90e57611ed9************',
        )
        print(response)
        return response
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
JobIDs	任务ID，以分割多个任务ID	String

返回结果说明



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

```
        "Code": "Success",
        "Message": "",
        "JobId": "f93984788066911ed89ed352d4d9d2084",
        "State": "Submitted",
        "CreationTime": "2022-07-18T15:16:43+0800",
        "EndTime": "-",
        "StartTime": "-",
        "QueueId": "p2911917386e148639319e13c285cc774",
        "Tag": "FileUncompress",
        "Input": {
            "BucketId": "test-1234567890",
            "Object": "input/test.zip",
            "Region": "ap-chongqing"
        },
        "Operation": {
            "FileUncompressConfig": {
                "Prefix": "output/",
                "PrefixReplaced": 1
            },
            "Output": {
                "Region": "ap-chongqing",
                "Bucket": "test-1234567890"
            },
            "UserData": "This is my data."
        }
        }
    }
}
```


参数名称	描述	类型
Response	结果的详情，详见 Response 详情	dict

提交文件解压任务

最近更新时间：2024-03-13 11:32:21

简介

本文档提供关于文件处理文件解压异步请求的 API 概览以及 SDK 示例代码。

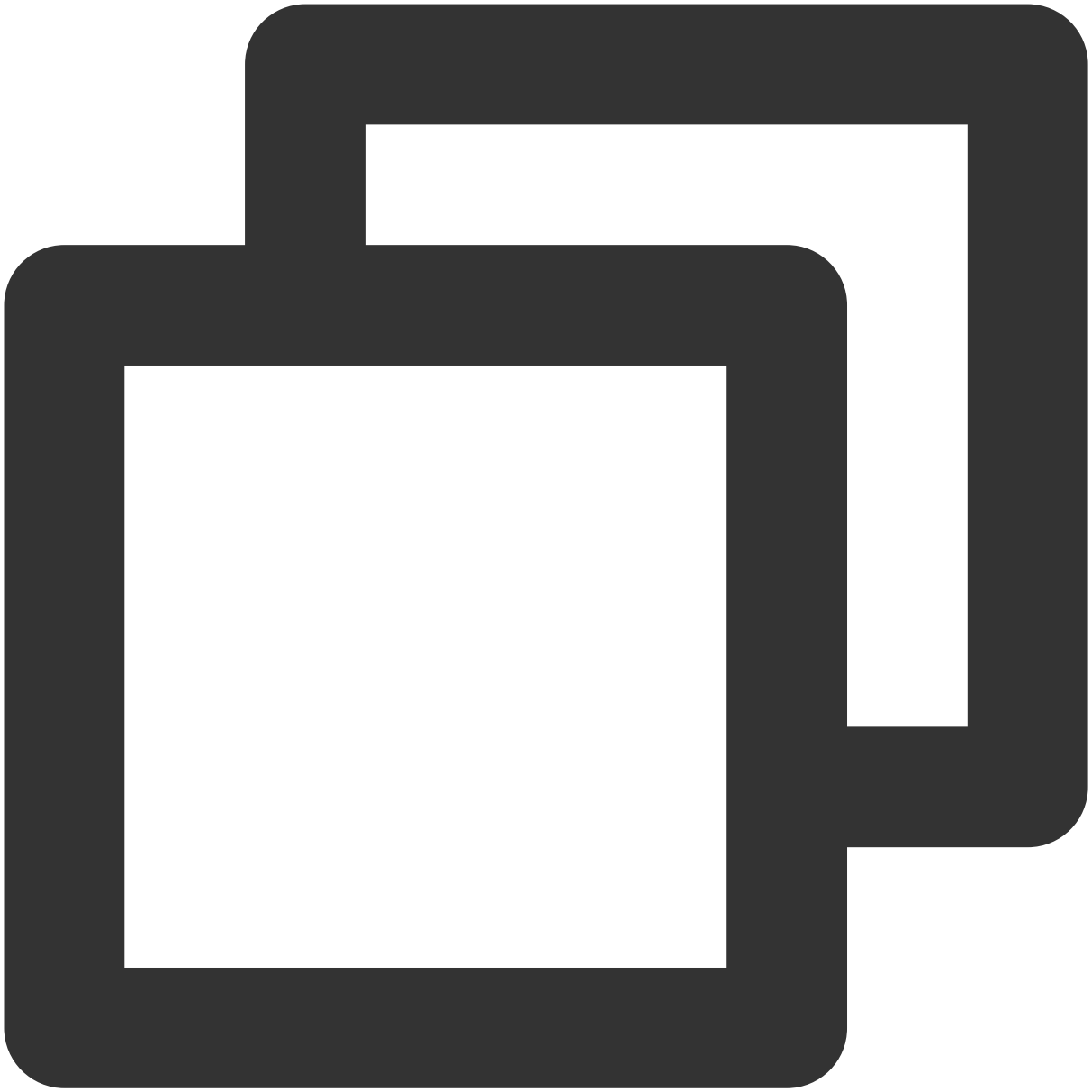
API	说明
提交文件解压任务	用于异步提交文件解压任务

异步请求接口

功能说明

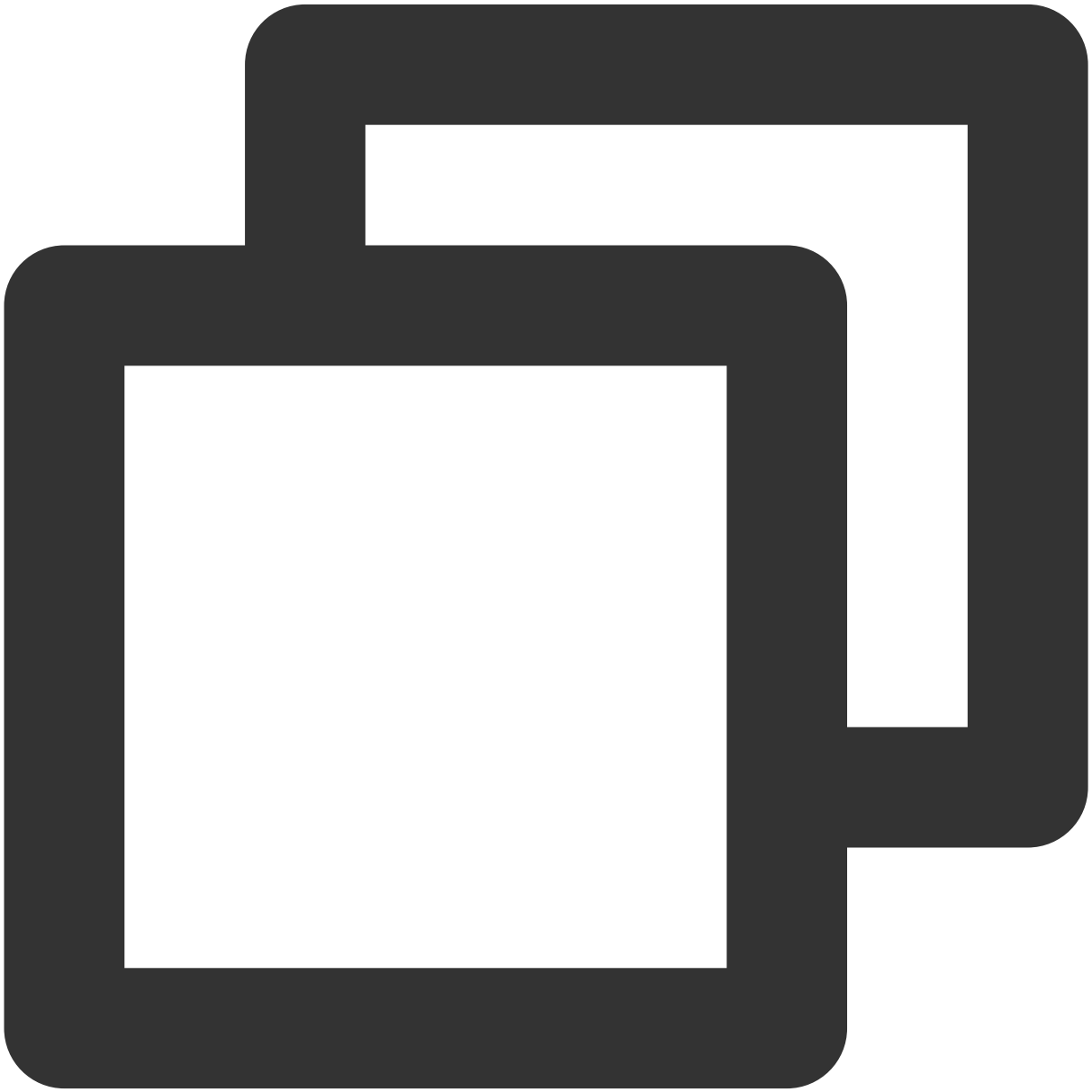
文件处理文件解压异步请求。

方法原型



```
def ci_create_file_uncompress_job(self, Bucket, InputObject,
    FileUncompressConfig, QueueId=None, CallBack=None, CallBackFormat=None,
    CallBackType=None, CallBackMqConfig=None, UserData=None, **kwargs)
```

请求示例

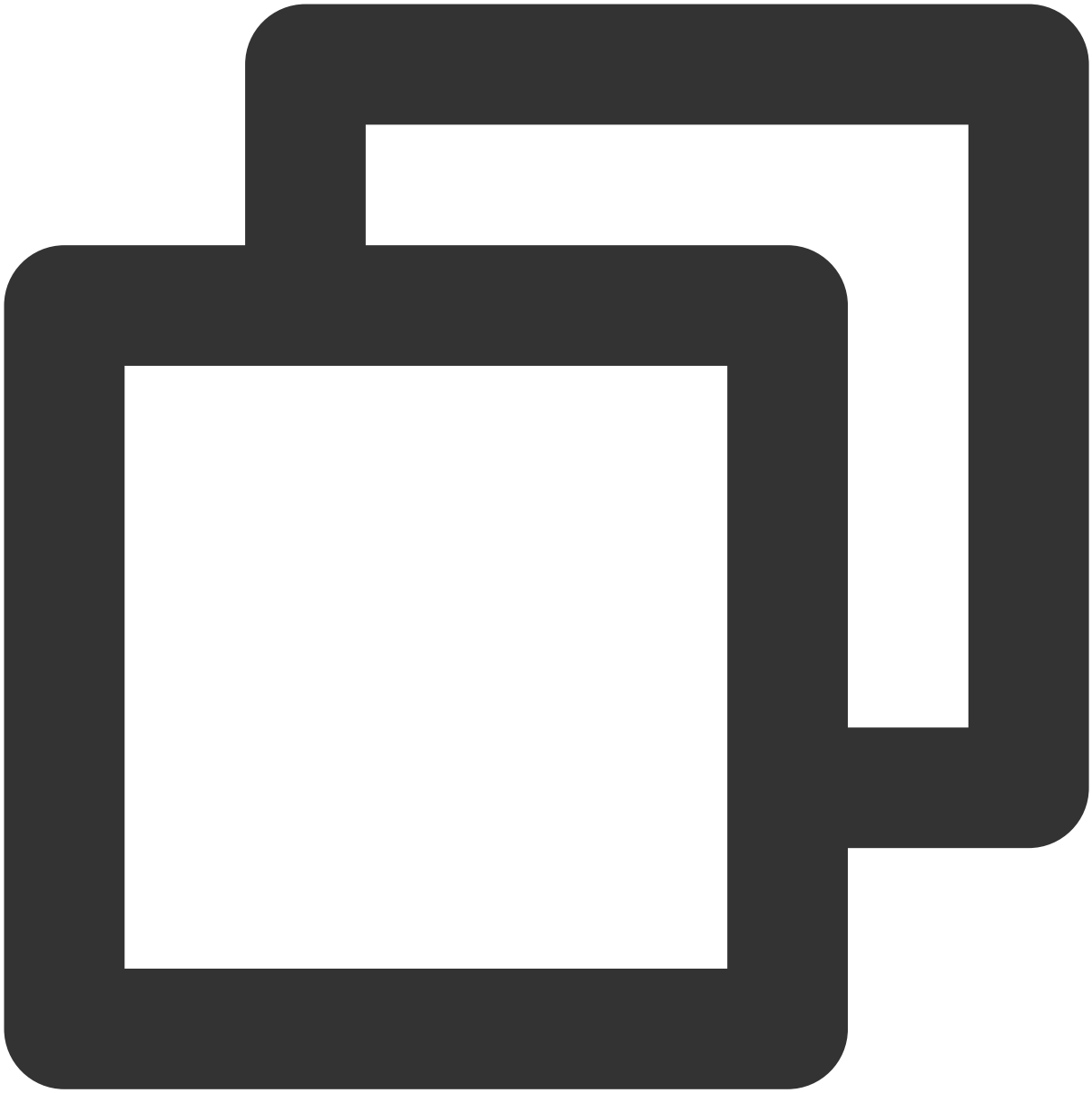


```
def ci_create_file_uncompress_job():
    body = {
        'Prefix': 'output/',
    }
    response = client.ci_create_file_uncompress_job(
        Bucket=bucket_name,
        InputObject='test.zip',
        FileUncompressConfig=body
    )
    print(response)
    return response
```

参数说明

参数名称	描述	类型
Bucket	对象所在存储桶	String
InputObject	对象文件名，例如 folder/document.pdf	String
FileUncompressConfig	指定文件解压缩的处理规则，详见 FileUncompressConfig 详情	dict
QueueId	任务所在的队列 ID	String
CallBack	任务回调地址，优先级高于队列的回调地址，设置为 no 时，表示队列的回调地址不产生回调	string
CallBackFormat	任务回调格式，JSON 或 XML，默认 XML，优先级高于队列的回调格式	string
CallBackType	任务回调类型，Url 或 TDMQ，默认 Url，优先级高于队列的回调类型	string
CallBackMqConfig	任务回调TDMQ配置，当 CallBackType 为 TDMQ 时必填，详见 CallBackMqConfig 详情	dict
UserData	透传用户信息，可打印的 ASCII 码，长度不超过1024	String

返回结果说明




```
{
    "Response": {
        "JobsDetail": {
        "Code": "Success",
        "Message": "",
        "JobId": "f93984788066911ed89ed352d4d9d2084",
        "State": "Submitted",
        "CreationTime": "2022-07-18T15:16:43+0800",
        "EndTime": "-",
        "StartTime": "-",
        "QueueId": "p2911917386e148639319e13c285cc774",
```

```
        "Tag": "FileUncompress",
        "Input": {
            "BucketId": "test-1234567890",
            "Object": "input/test.zip",
            "Region": "ap-chongqing"
        },
        "Operation": {
            "FileUncompressConfig": {
                "Prefix": "output/",
                "PrefixReplaced": 1
            },
            "Output": {
                "Region": "ap-chongqing",
                "Bucket": "test-1234567890"
            },
            "UserData": "This is my data."
        }
        }
    }
}
```

参数名称	描述	类型
Response	结果的详情，详见 Response 详情	dict

小程序 SDK
快速入门

最近更新时间：2024-03-13 11:41:12

下载与安装

相关资源

对象存储 COS 的 XML 小程序 SDK 源码下载地址：XML 小程序 SDK。
SDK 快速下载地址：XML 小程序 SDK。
演示示例 Demo 下载地址：XML 小程序 SDK Demo。
SDK 文档中的所有示例代码请参见 SDK 代码示例。
SDK 更新日志请参见 ChangeLog。
SDK 常见问题请参见：小程序 SDK 常见问题。
说明：
如果您在使用 SDK 时遇到函数或方法不存在等错误，请先将 SDK 升级到最新版再重试。

环境依赖

1. 该 SDK 只适用于微信小程序环境。
2. 登录 对象存储控制台 创建存储桶后，获取存储桶名称和 地域信息。
3. 登录 访问管理控制台 获取您的项目 SecretId 和 SecretKey。

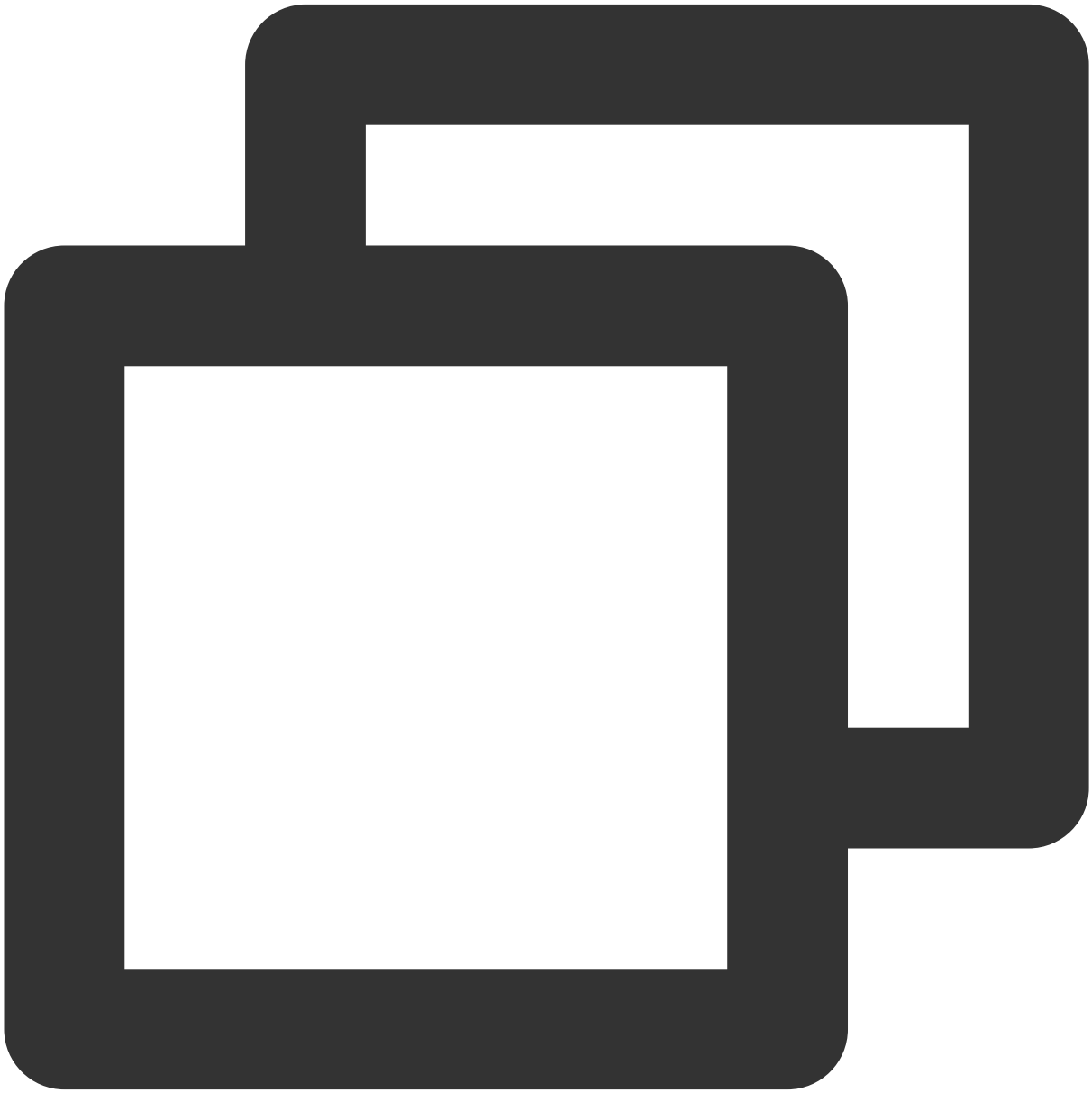
说明：
关于本文中出现的 SecretId、SecretKey、Bucket 等名称的含义和获取方式请参见 COS 术语信息。
关于跨端框架（例如 uni-app）的使用说明，使用小程序 SDK 开发后无法打包成正常使用的移动应用，例如安卓 App、iOS App，需要使用对应的安卓 SDK、iOS SDK。
XCosSecurityToken 字段说明：v1.2.0之前版本的 SDK 只支持 XCosSecurityToken，v1.2.0及之后的版本请使用 SecurityToken 代替。

安装 SDK

安装小程序 SDK 有两种方式：手动安装和 npm 安装，具体安装方法如下。

手动安装

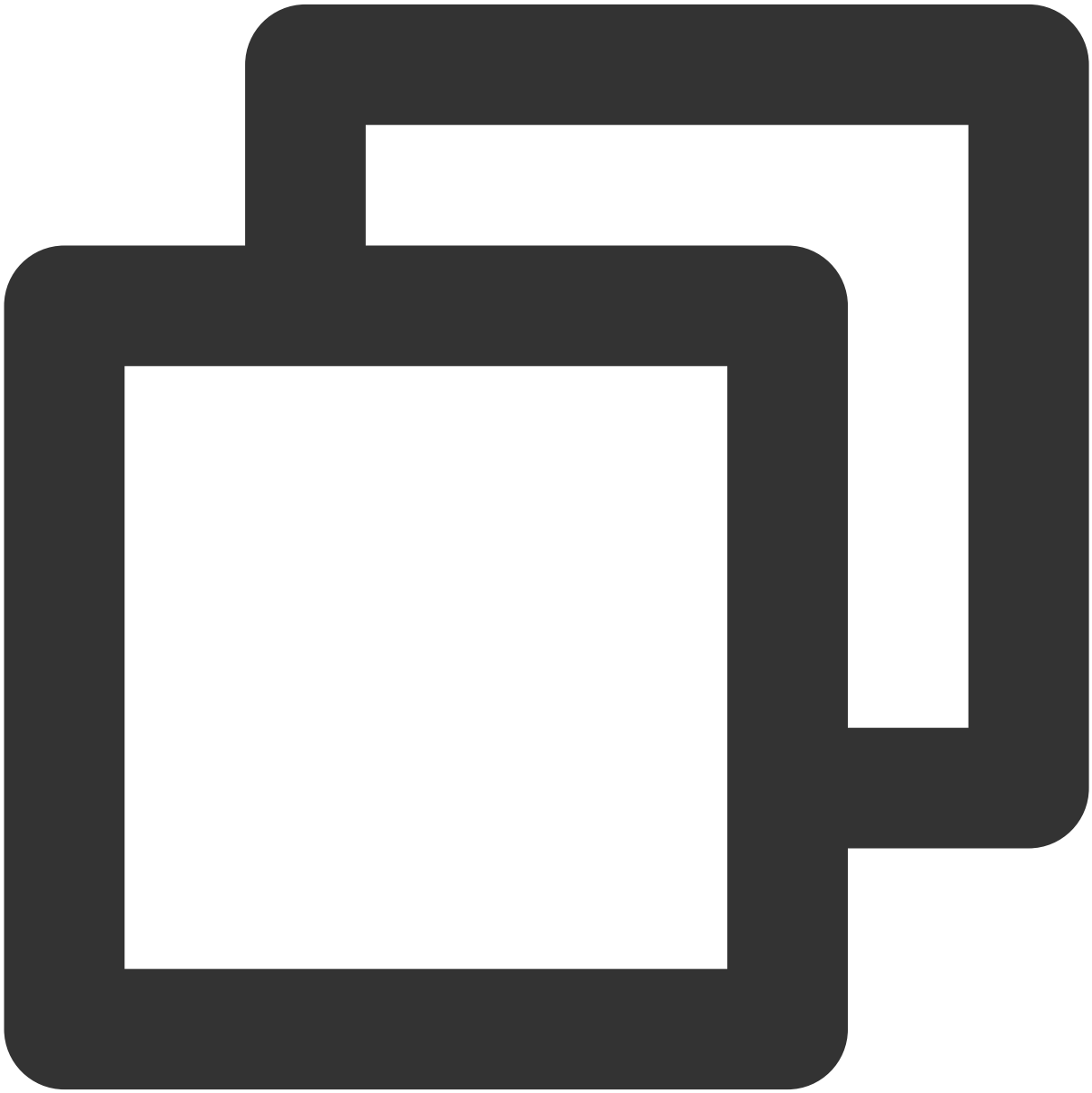
复制源码文件中的 cos-wx-sdk-v5.js 到自己小程序代码根目录下任意路径，并用相对路径引用：



```
var COS = require('./lib/cos-wx-sdk-v5.js')
```

npm 安装

如果小程序代码使用了 webpack 打包，则通过 npm 安装依赖即可：



```
npm install cos-wx-sdk-v5
```

其中，程序代码使用 `var COS = require('cos-wx-sdk-v5');` 进行引用。更多详情请参见 npm 支持。

开始使用

小程序域名白名单配置

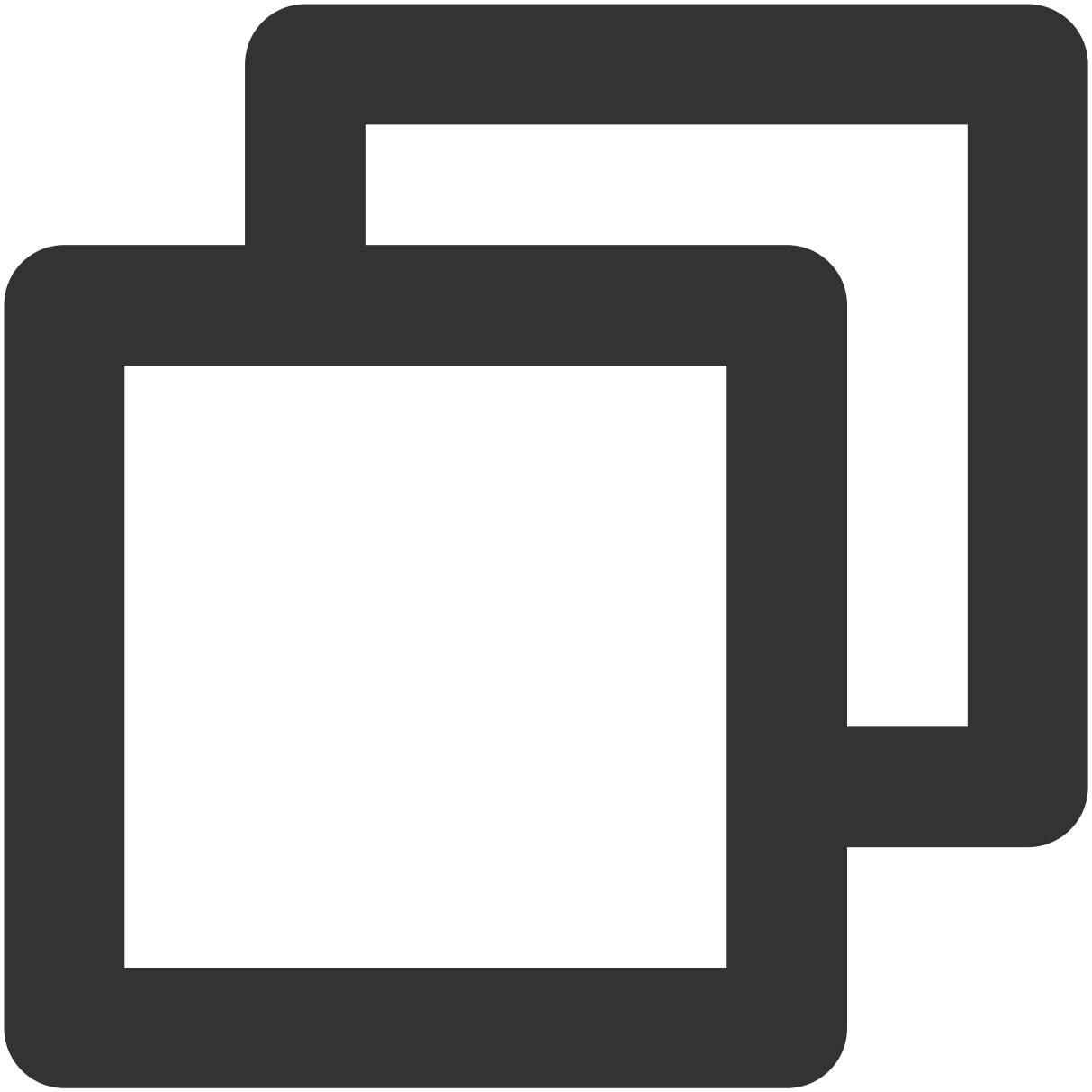
小程序里请求 COS 需要登录到 微信公众平台，选择**开发 > 开发设置 > 服务器域名**，配置域名白名单。SDK 使用到了两个接口：

1. cos.postObject 使用 wx.uploadFile 方法。
2. 其他方法使用 wx.request 方法。

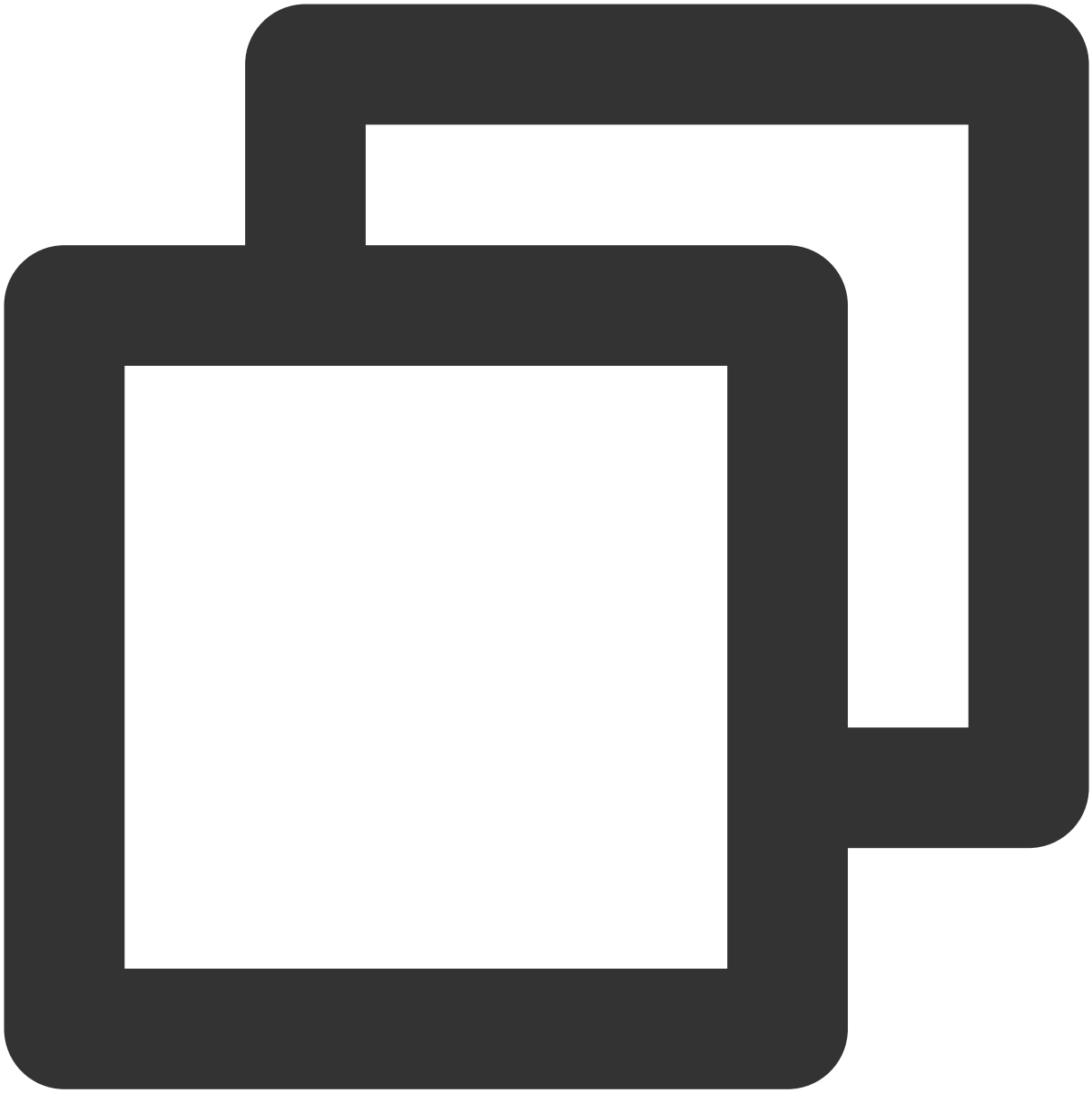
需要在对应白名单里，配置 COS 域名，白名单域名格式有两种：

1. 如果是标准请求，可以配置存储桶域名作为白名单域名，例如：`examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com`。
2. 如果小程序使用的存储桶多，可以选择后缀式请求 COS，只需要在 SDK 实例化时传入 `ForcePathStyle: true`，这种方式需要配置地域域名作为白名单，例如：`cos.ap-guangzhou.myqcloud.com`。

初始化



```
var COS = require('./lib/cos-wx-sdk-v5.js')
```



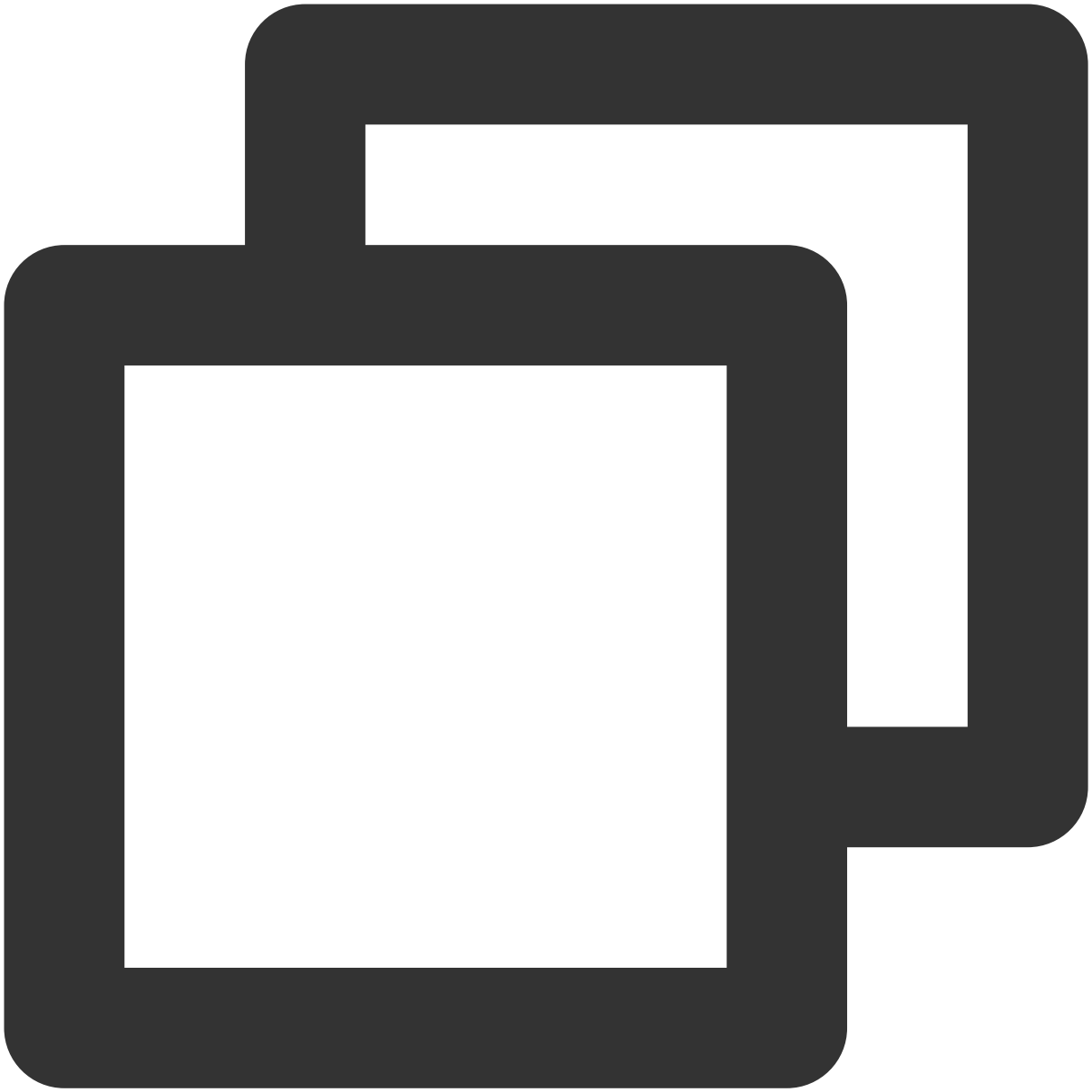
```
var cos = new COS({
    // ForcePathStyle: true, // 如果使用了很多存储桶，可以通过打开后缀式，减少配置白名单域名数
    SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时
    getAuthorization: function (options, callback) {
        // 初始化时不会调用，只有调用cos方法（比如cos.putObject）时才会进入
        // 异步获取临时密钥
        wx.request({
            url: 'https://example.com/server/sts.php',
            data: {
                bucket: options.Bucket,
                region: options.Region,
```

```
            },
            dataType: 'json',
            success: function (result) {
                var data = result.data;
                var credentials = data && data.credentials;
                if (!data || !credentials) return console.error('credentials invali
                callback({
                    TmpSecretId: credentials.tmpSecretId,
                    TmpSecretKey: credentials.tmpSecretKey,
                    // v1.2.0之前版本的sdk使用XCosSecurityToken而不是SecurityToken
                    SecurityToken: credentials.sessionToken,
                    // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致
                    StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
                    ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000900
                });
            }
        });
    }
});

// 接下来可以通过 cos 实例调用 COS 请求。
// TODO
```

配置项

使用示例

创建一个 COS SDK 实例，COS SDK 支持以下几种格式创建：
格式一（推荐）：后端通过获取临时密钥给到前端，前端计算签名。



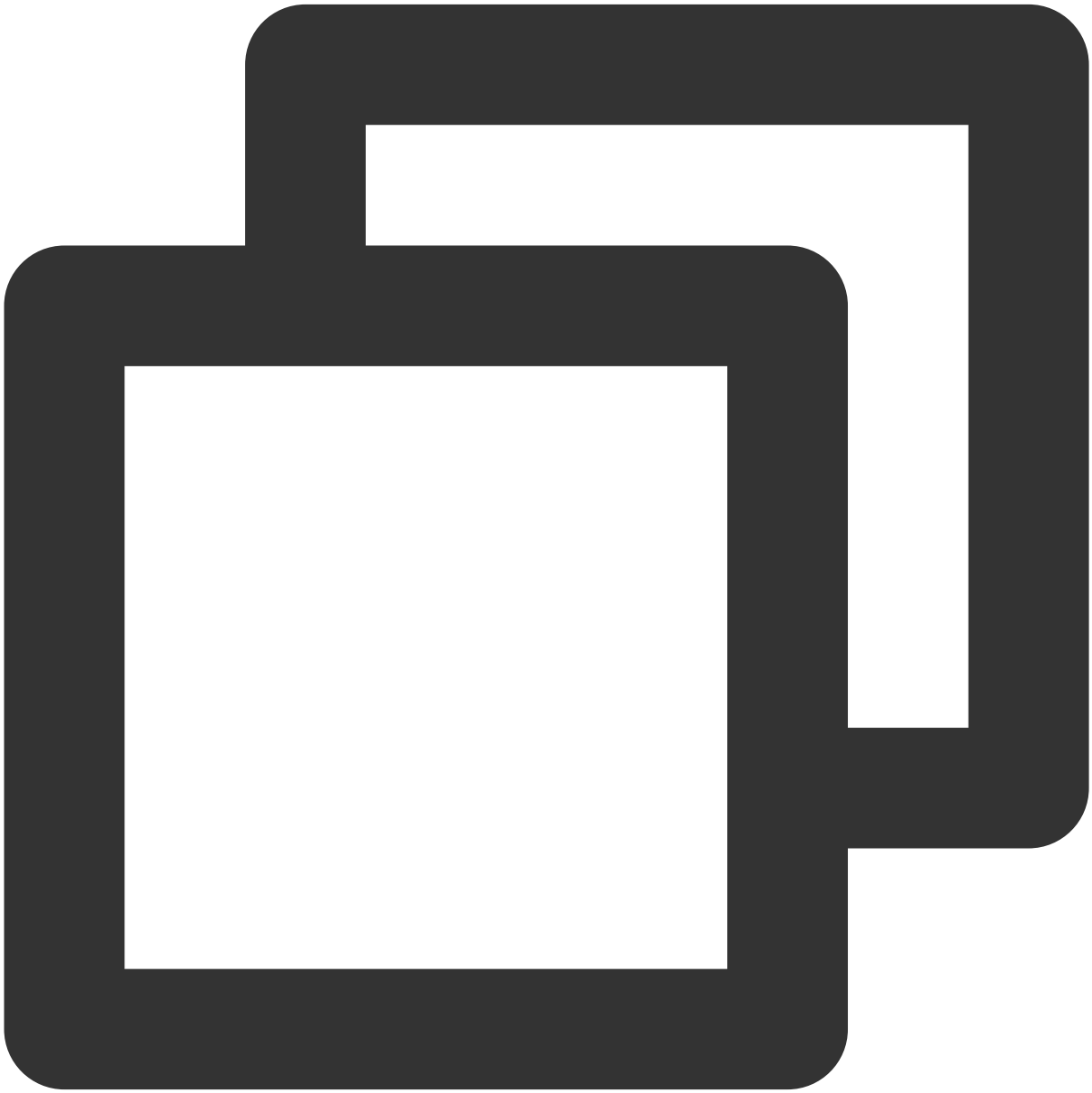
```
var cos = new COS({
    SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时
    // 必选参数
    getAuthorization: function (options, callback) {
        // 服务端 JS 和 PHP 示例：https://github.com/tencentyun/cos-js-sdk-v5/blob/mas
        // 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-st
        // STS 详细文档指引看：https://www.tencentcloud.com/document/product/436/14048
        wx.request({
            url: 'https://example.com/server/sts.php',
            data: {
                // 可从 options 取需要的参数
```

```
            },
            success: function (result) {
                var data = result.data;
                var credentials = data && data.credentials;
                if (!data || !credentials) return console.error('credentials invali
                callback({
                    TmpSecretId: credentials.tmpSecretId,
                    TmpSecretKey: credentials.tmpSecretKey,
                    // v1.2.0之前版本的sdk使用XCosSecurityToken而不是SecurityToken
                    SecurityToken: credentials.sessionToken,
                    // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致
                    StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
                    ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000900
                });
            }
        });
    }
});
```

说明：

临时密钥生成和使用可参见 临时密钥生成及使用指引。

格式二（推荐）：细粒度控制权限，后端通过获取临时密钥给到前端，前端只有相同请求才重复使用临时密钥，后端可以通过 Scope 细粒度控制权限。



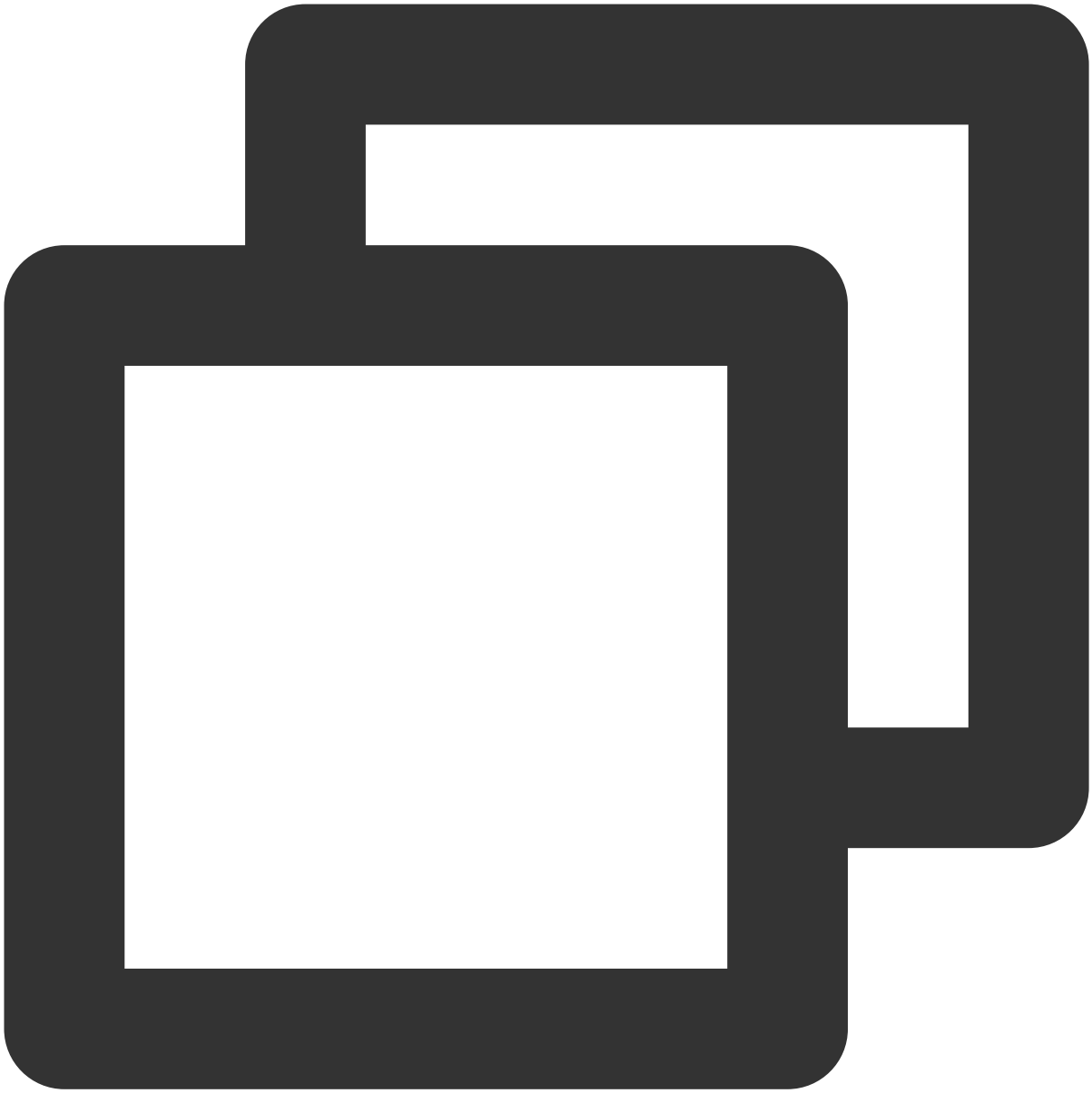
```
var cos = new COS({
    SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时
    // 必选参数
    getAuthorization: function (options, callback) {
        // 服务端示例：https://github.com/tencentyun/qcloud-cos-sts-sdk/edit/master/s
        wx.request({
            url: 'https://example.com/server/sts-scope.php',
            data: JSON.stringify(options.Scope),
            success: function (result) {
                var data = result.data;
                var credentials = data && data.credentials;
```

```
                if (!data || !credentials) return console.error('credentials invali
                callback({
                    TmpSecretId: credentials.tmpSecretId,
                    TmpSecretKey: credentials.tmpSecretKey,
                    // v1.2.0之前版本的sdk使用XCosSecurityToken而不是SecurityToken
                    SecurityToken: credentials.sessionToken,
                    // 建议返回服务器时间作为签名的开始时间，避免用户浏览器本地时间偏差过大导致
                    StartTime: data.startTime, // 时间戳，单位秒，如：1580000000
                    ExpiredTime: data.expiredTime, // 时间戳，单位秒，如：1580000900
                    ScopeLimit: true, // 细粒度控制权限需要设为 true，会限制密钥只在相同请
                });
            }
        });
    }
});
```

说明：

临时密钥生成和使用可参见 临时密钥生成及使用指引。

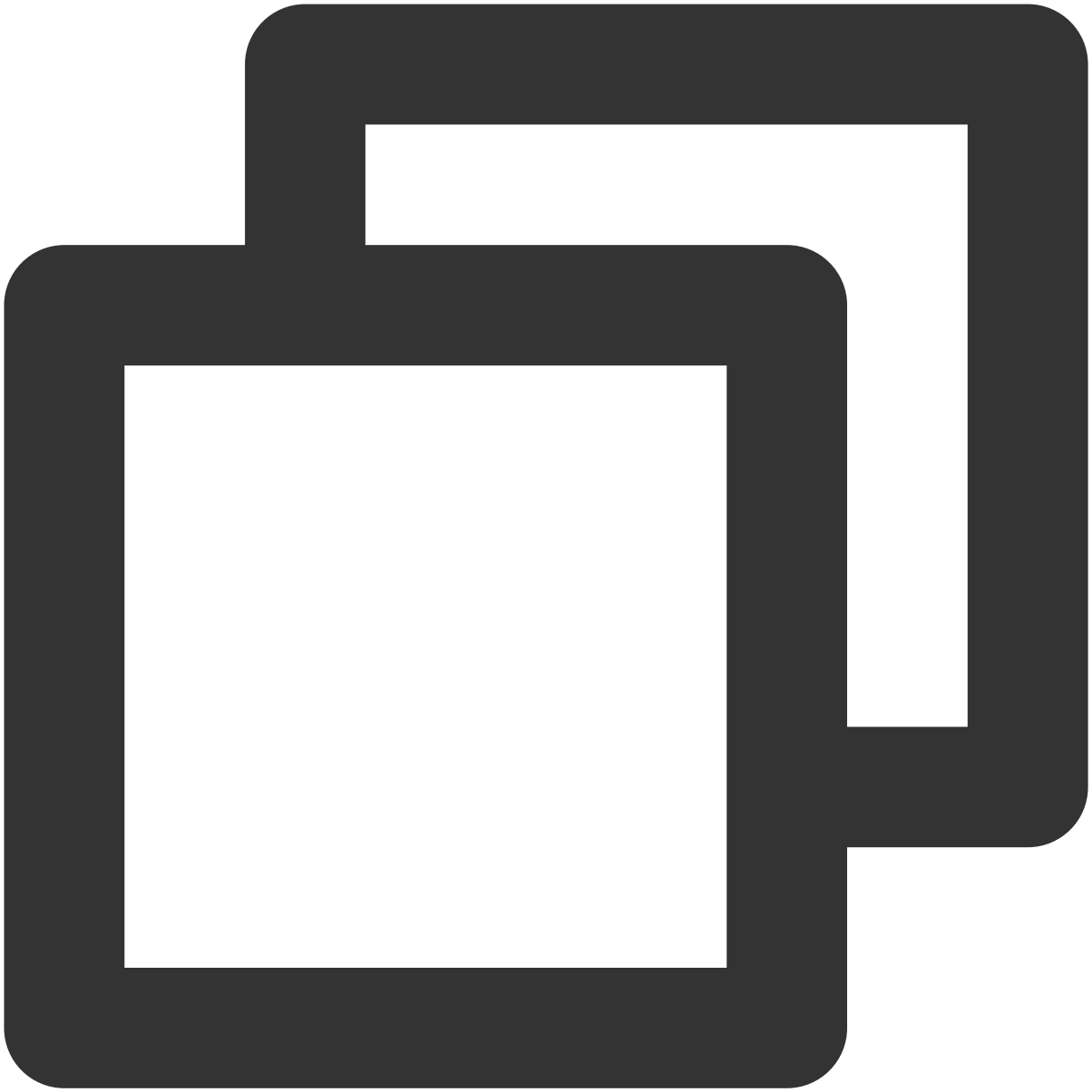
格式三（不推荐）：前端每次请求前都需要通过 getAuthorization 获取签名，后端使用固定密钥或临时密钥计算签名返回给前端。该格式分块上传权限不易控制，不推荐您使用此格式。



```
var cos = new COS({
    SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时
    // 必选参数
    getAuthorization: function (options, callback) {
        // 服务端获取签名，请参考对应语言的 COS SDK：https://www.tencentcloud.com/documer
        // 注意：这种有安全风险，后端需要通过 method、pathname 严格控制好权限，例如不允许 put
        wx.request({
            url: 'https://example.com/server/auth.php',
            data: JSON.stringify(options.Scope),
            success: function (result) {
                var data = result.data;
```

```
                if (!data || !data.authorization) return console.error('authorizati
                callback({
                    Authorization: data.authorization,
                    // v1.2.0之前版本的sdk使用XCosSecurityToken而不是SecurityToken
                    // SecurityToken: data.sessionToken, // 如果使用临时密钥，需要把 se
                });
            }
        });
    },
});
```

格式四（不推荐）：前端使用固定密钥计算签名，该格式适用于前端调试，若使用此格式，请避免泄露密钥。



```
// SECRETID 和 SECRETKEY请登录 https://console.tencentcloud.com/cam/capi 进行查看和管理
var cos = new COS({
    SecretId: 'SECRETID',
    SecretKey: 'SECRETKEY',
    SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时
});
```

构造函数参数说明

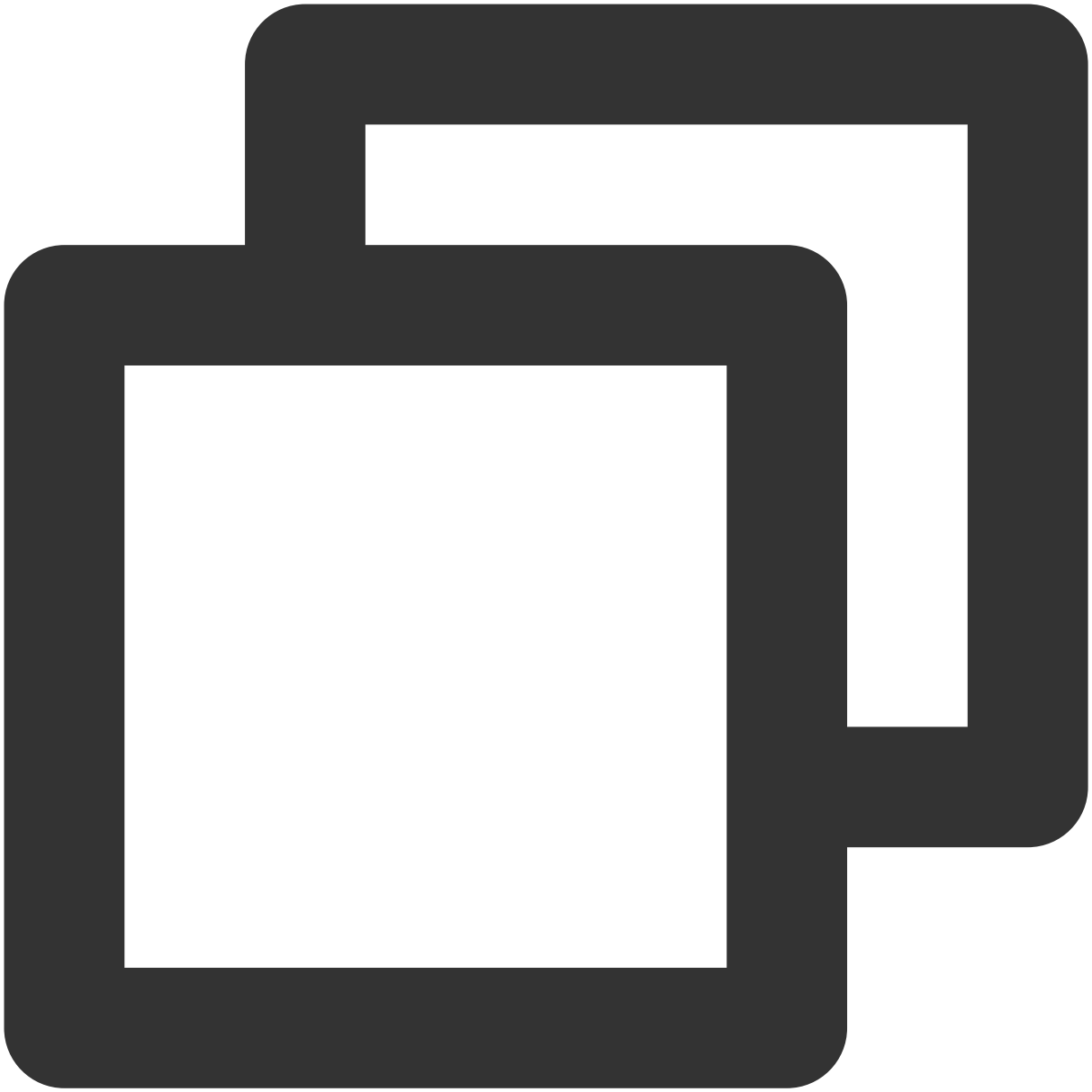
参数名	参数描述	类型	是否必

			填
SecretId	用户的 SecretId	String	否
SecretKey	用户的 SecretKey，建议只在前端调试时使用，避免暴露密钥	String	否
FileParallelLimit	同一个实例下上传的文件并发数，默认值3	Number	否
ChunkParallelLimit	同一个上传文件的分块并发数，默认值3	Number	否
ChunkRetryTimes	分块上传及分块复制时，出错重试次数，默认值2（加第一次，请求共3次）	Number	否
ChunkSize	分块上传时，每块的字节数大小，默认值1048576（1MB）	Number	否
SliceSize	使用 uploadFiles 批量上传时，文件大小大于该数值就使用分块上传 sliceUploadFile，否则使用简单上传 putObject，默认值1048576（1MB）	Number	否
CopyChunkParallelLimit	进行分块复制操作中复制分块上传的并发数，默认值20	Number	否
CopyChunkSize	使用 sliceCopyFile 分块复制文件时，每块的大小字节数，默认值10485760（10MB）	Number	否
CopySliceSize	使用 sliceCopyFile 分块复制文件时，文件大小大于该数值就会使用分块复制 sliceCopyFile，否则使用简单复制 putObjectCopy，默认值10485760（10MB）	Number	否
ProgressInterval	上传进度的回调方法 onProgress 的回调频率，单位 ms，默认值1000	Number	否
Protocol	发请求时用的协议，可选项 `https:`、`http:`，默认判断当前页面是 `http:` 时使用 `http:`，否则使用 `https:`	String	否
ServiceDomain	调用 getService 方法时，请求的域名，例如 `service.cos.myqcloud.com`	String	否
Domain	调用操作存储桶和对象的 API 时自定义请求域名。可以使用模板，例如 `"{Bucket}.cos.{Region}.myqcloud.com"`，即在调用 API 时会使用参数中传入的 Bucket 和 Region 进行替换	String	否
UploadQueueSize	上传队列最长大小，超出的任务如果状态不是 waiting、checking、uploading 会被清理，默认10000	Number	否

版权所有：腾讯云计算（北京）有限责任公司

ForcePathStyle	强制使用后缀式模式发请求，后缀式模式中 Bucket 会放在域名后的 pathname 里，并且 Bucket 会加入签名 pathname 计算，默认为 false	Boolean	否
UploadCheckContentMd5	强制上传文件校验 Content-MD5，会对文件请求 Body 计算 md5 放在 header 的 Content-MD5 字段里，默认为 false	Boolean	否
getAuthorization	获取签名的回调方法，如果没有 SecretId、SecretKey 时，这个参数必选。 **注意: 该回调方法在初始化实例时传入，在使用实例调用接口时才会执行并获取签名。**	Function	否
UseAccelerate	是否启用全球加速域名，默认为 false。若改为 true，需要存储桶开启全球加速功能，详情请参见 开启全球加速。	Boolean	否
SimpleUploadMethod	高级上传、批量上传内部对小文件做简单上传的方法名，可选'postObject'或'putObject'，默认为'postObject'（该参数v1.3.0版本开始支持）	String	否

getAuthorization 回调函数说明的函数说明（使用格式一）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的回调参数说明：

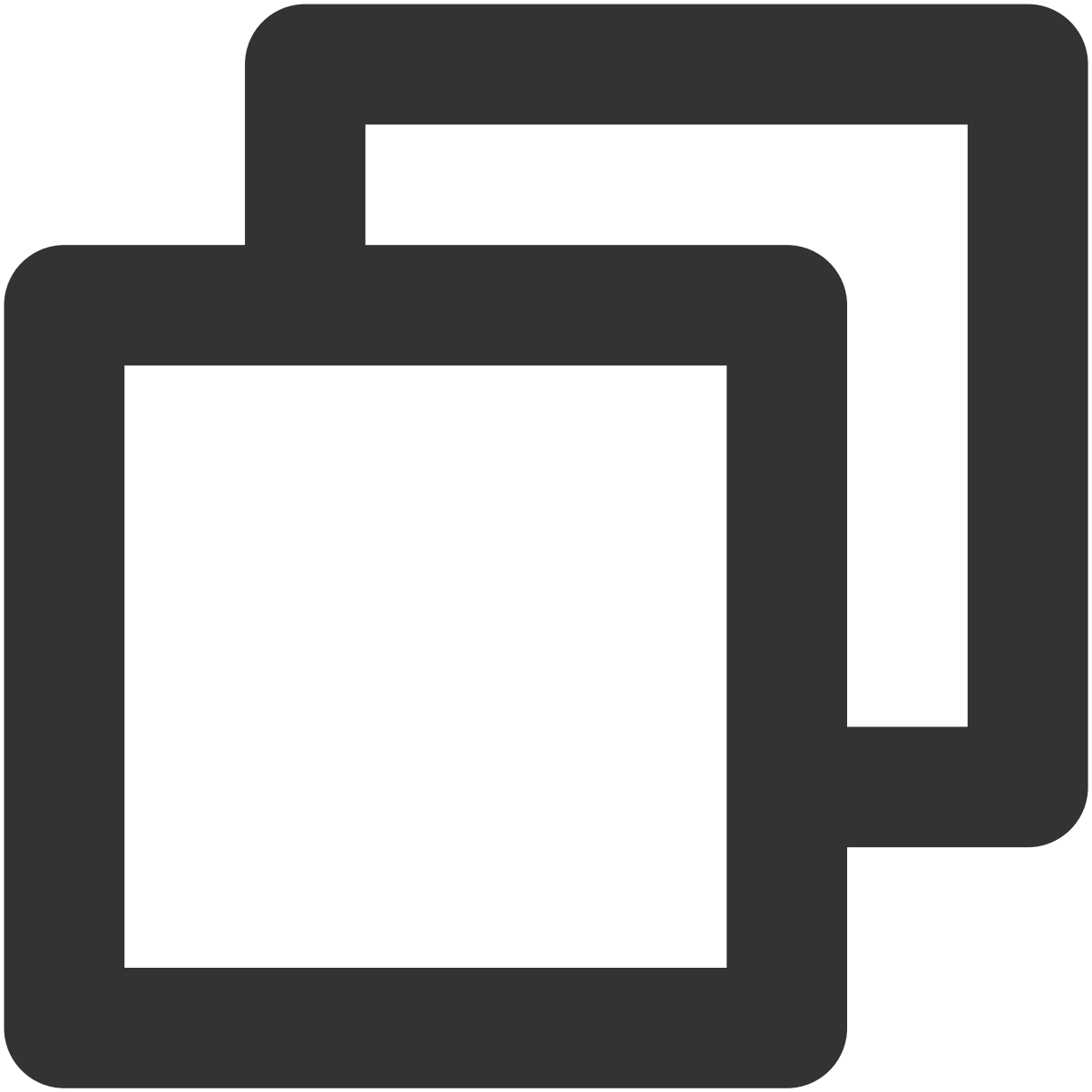
参数名	参数描述	类型
options	获取临时密钥需要的参数对象	Object
- Bucket	存储桶的名称，命名规则为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String

- Region	存储桶所在地域，枚举值请参见 地域和访问域名	String
callback	临时密钥获取完成后的回传方法	Function

获取完临时密钥后，callback 回传一个对象，回传对象的属性列表如下：

属性名	参数描述	类型	是否必填
TmpSecretId	获取回来的临时密钥的 tmpSecretId	String	是
TmpSecretKey	获取回来的临时密钥的 tmpSecretKey	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段。v1.2.0之前版本的 SDK 使用 XCosSecurityToken 而不是 SecurityToken	String	是
StartTime	密钥获取的开始时间，即获取时刻的时间戳，单位秒，startTime，如：1580000000，用于签名开始时间，传入该参数可避免前端时间偏差签名过期问题	String	否
ExpiredTime	获取回来的临时密钥的 expiredTime，超时时刻的时间戳，单位秒，如：1580000900	String	是

getAuthorization 回调函数说明（使用格式二）



```
getAuthorization: function(options, callback) { ... }
```

getAuthorization 的函数说明回调参数说明：

参数名	参数描述	类型
options	获取签名需要的参数对象	Object
- Method	当前请求的 Method	Object
- Pathname	请求路径，用于签名计算	String

- Key	对象键（Object 的名称），对象在存储桶中的唯一标识，了解更多可参见 对象概述。 **注意:当使用实例请求的接口不是对象操作相关接口时，该参数为空。**	String
- Query	当前请求的 query 参数对象，{key: 'val'} 的格式	Object
- Headers	当前请求的 header 参数对象，{key: 'val'} 的格式	Object
callback	临时密钥获取完成后的回调	Function

getAuthorization 计算完成后，callback 回传参数支持两种格式：
格式一：回传鉴权凭证字符串 Authorization。
格式二：回传一个对象，对象属性列表如下：

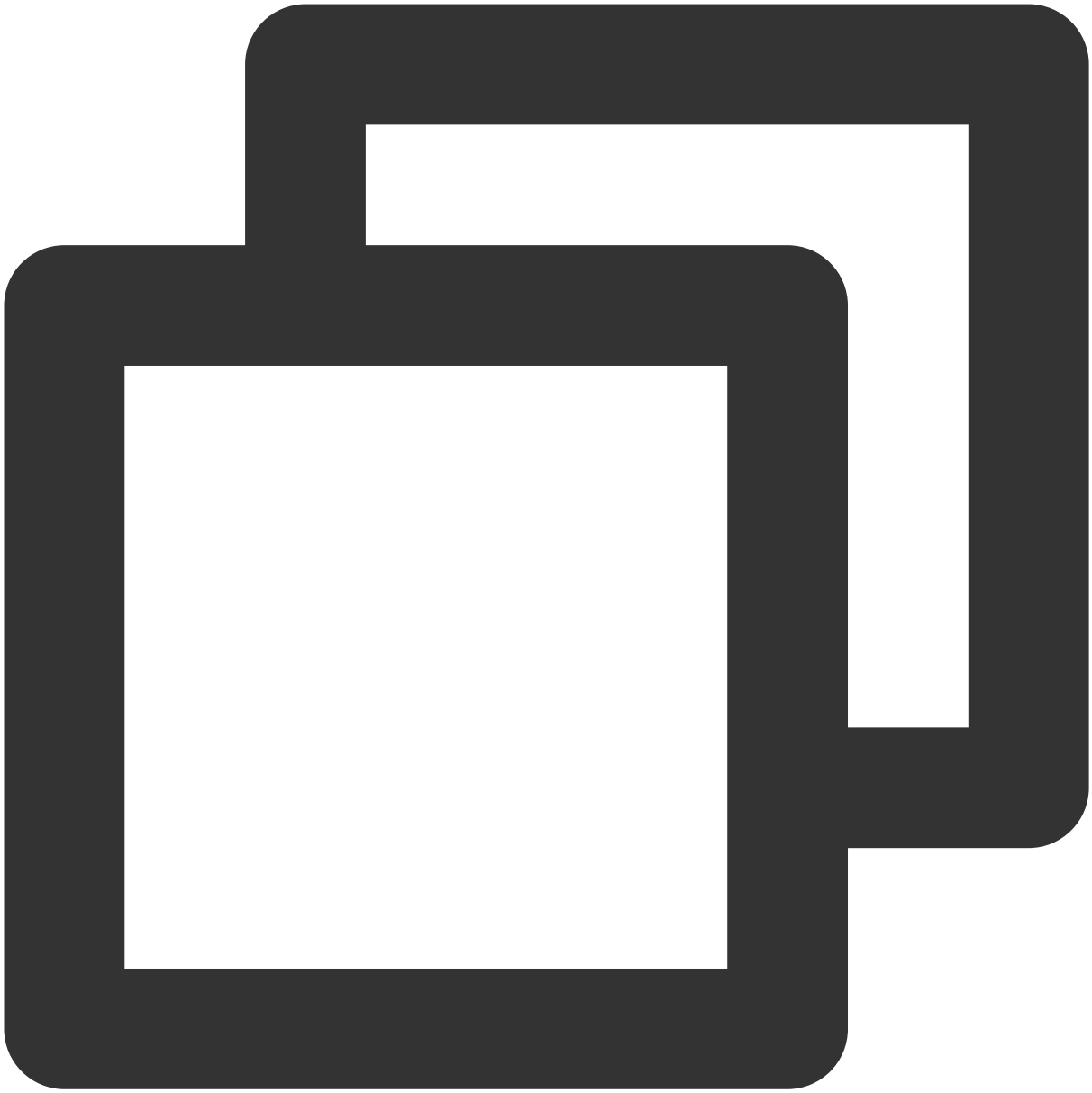
属性名	参数描述	类型	是否必填
Authorization	计算得到的签名字符串	String	是
SecurityToken	获取回来的临时密钥的 sessionToken，对应 header 的 x-cos-security-token 字段。v1.2.0之前版本的 SDK 使用 XCosSecurityToken 而不是 SecurityToken	String	否

获取鉴权凭证

实例本身鉴权凭证可以通过实例化时传入的参数控制如何获取，有三种获取方式：
1. 实例化时，传入 SecretId、SecretKey，每次需要签名都由实例内部计算。
2. 实例化时，传入 getAuthorization 回调，每次需要签名通过这个回调计算完返回签名给实例。
3. 实例化时，传入 getSTS 回调，每次需要临时密钥通过这个回调回去完返回给实例，在每次请求时实例内部使用临时密钥计算得到签名。

开启 beacon 上报

为了持续跟踪和优化 SDK 的质量，给您带来更好的使用体验，我们在 SDK 中引入了 腾讯灯塔 SDK。
说明：
腾讯灯塔只对 COS 侧的请求性能进行监控，不会上报业务侧数据。
若是想开启该功能，请先确保 SDK 版本升级到1.4.0及以上，然后在初始化中指定 EnableTracker 为 true。
并在小程序域名白名单中 request 合法域名里添加：`https://h.trace.qq.com;https://oth.str.beacon.qq.com;https://otheve.beacon.qq.com;` 。

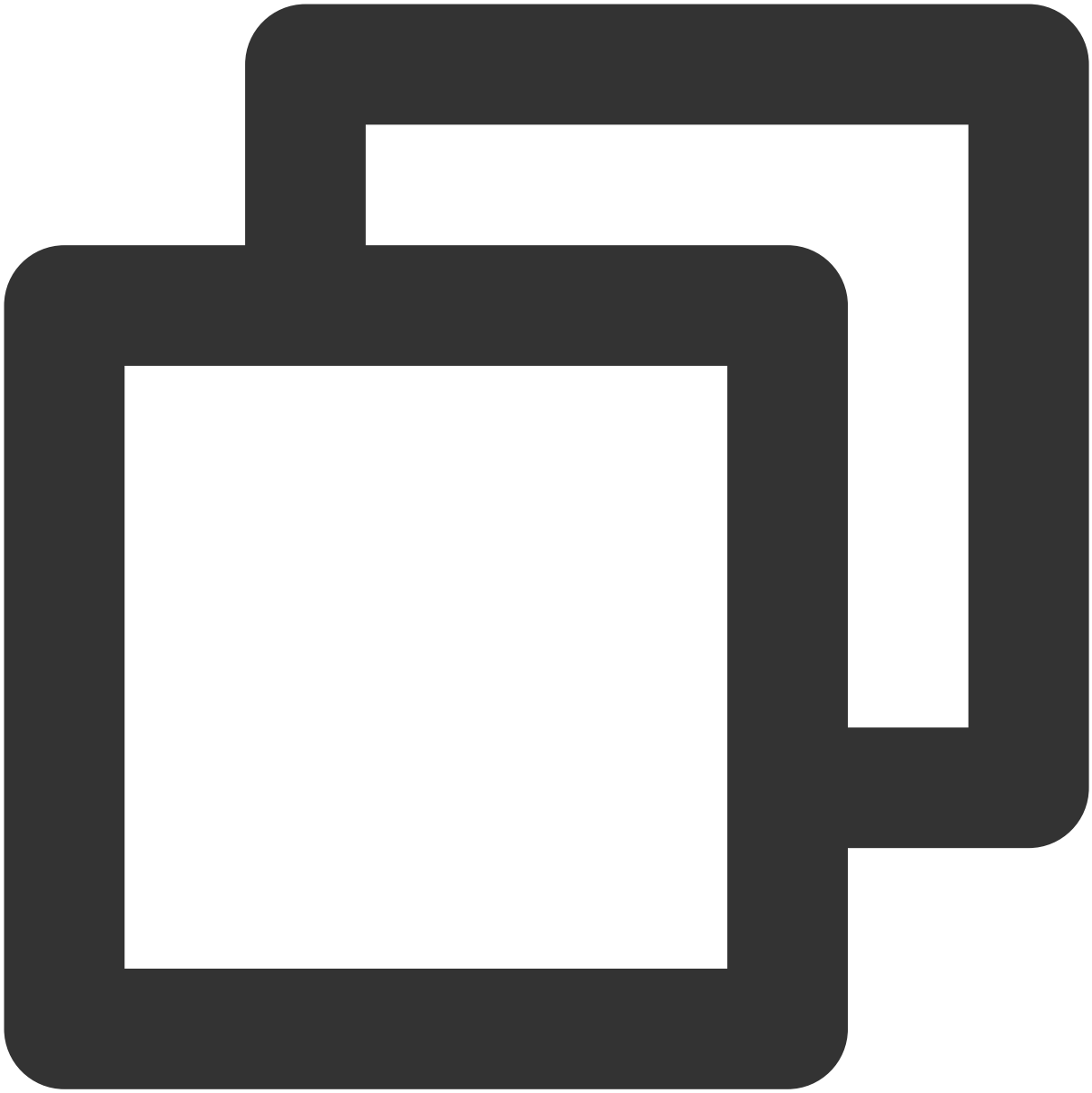


```
new COS({
  EnableTracker: true,
})
```

使用方式

回调方式

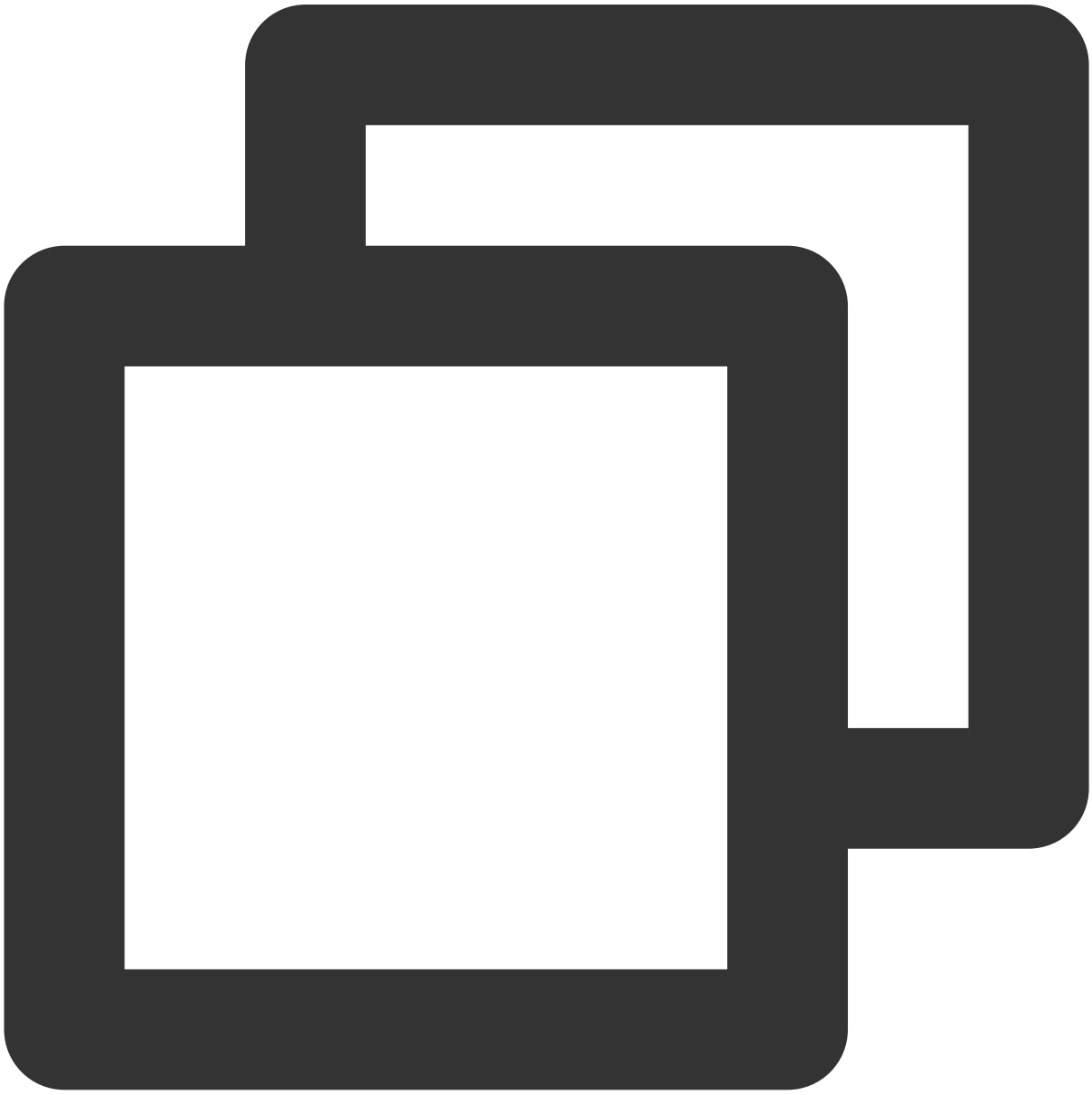
文档里默认使用回调方式，使用代码如下:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }, function(err, data) {
  if (err) {
    console.log('上传出错', err);
  } else {
    console.log('上传成功', data);
  }
});
```

Promise

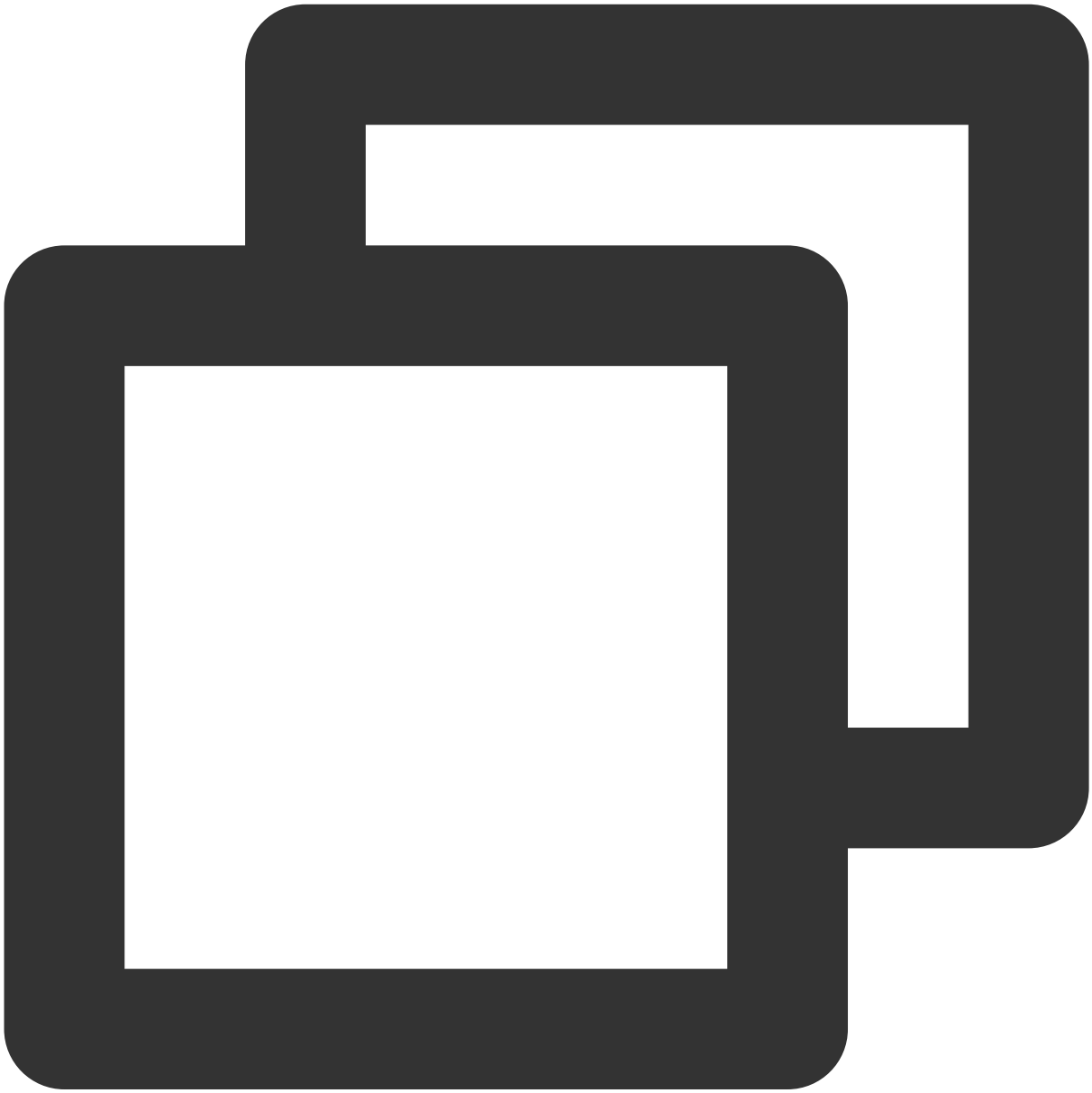
sdk同样支持 Promise 方式调用，比如上述回调方式的代码等同于以下代码:



```
// 这里省略初始化过程和上传参数
var cos = new COS({ ... });
cos.uploadFile({ ... }).then(data => {
  console.log('上传成功', data);
}).catch(err => {
  console.log('上传出错', err);
});
```

同步方式

同步方式基于 JavaScript 的 async 和 await，上述回调方式的代码等同于以下代码:



```
async function upload() {
  // 这里省略初始化过程和上传参数
  var cos = new COS({ ... });
  try {
    var data = await cos.uploadFile({ ... });
    return { err: null, data: data }
  } catch (err) {
    return { err: err, data: null };
```

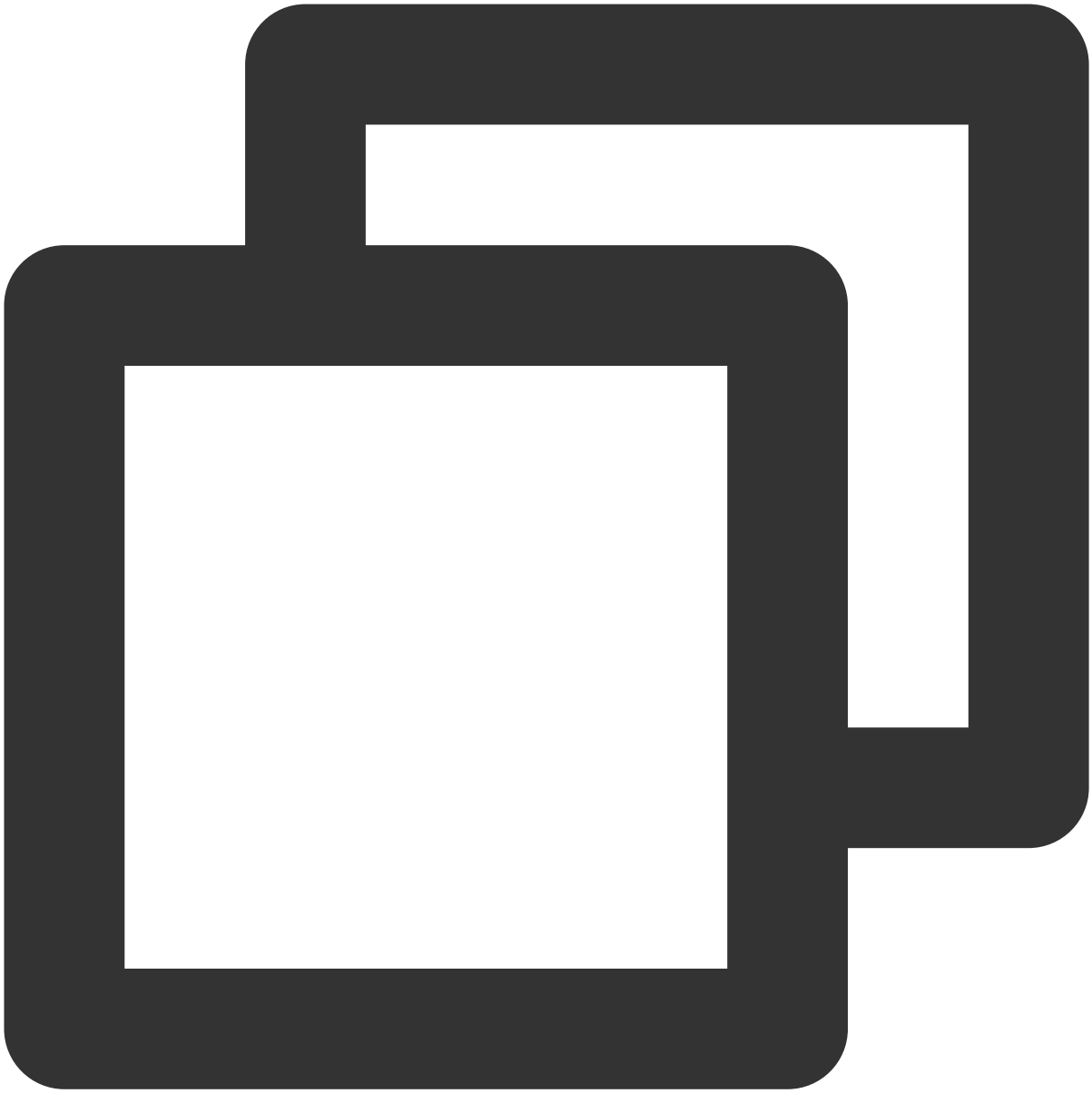
```
    }
  }
  // 可以同步拿到请求的返回值,这里举例说明,实际返回的数据格式可以自定义
  var uploadResult = await upload();
  if (uploadResult.err) {
    console.log('上传出错', uploadResult.err);
  } else {
    console.log('上传成功', uploadResult.data);
  }
```

注意：

cos.getObjectUrl 目前只支持回调方式。

使用技巧

通常情况下我们只需要创建一个 COS SDK 实例，然后在需要调用SDK方法的地方直接使用这个实例即可，示例代码如下：

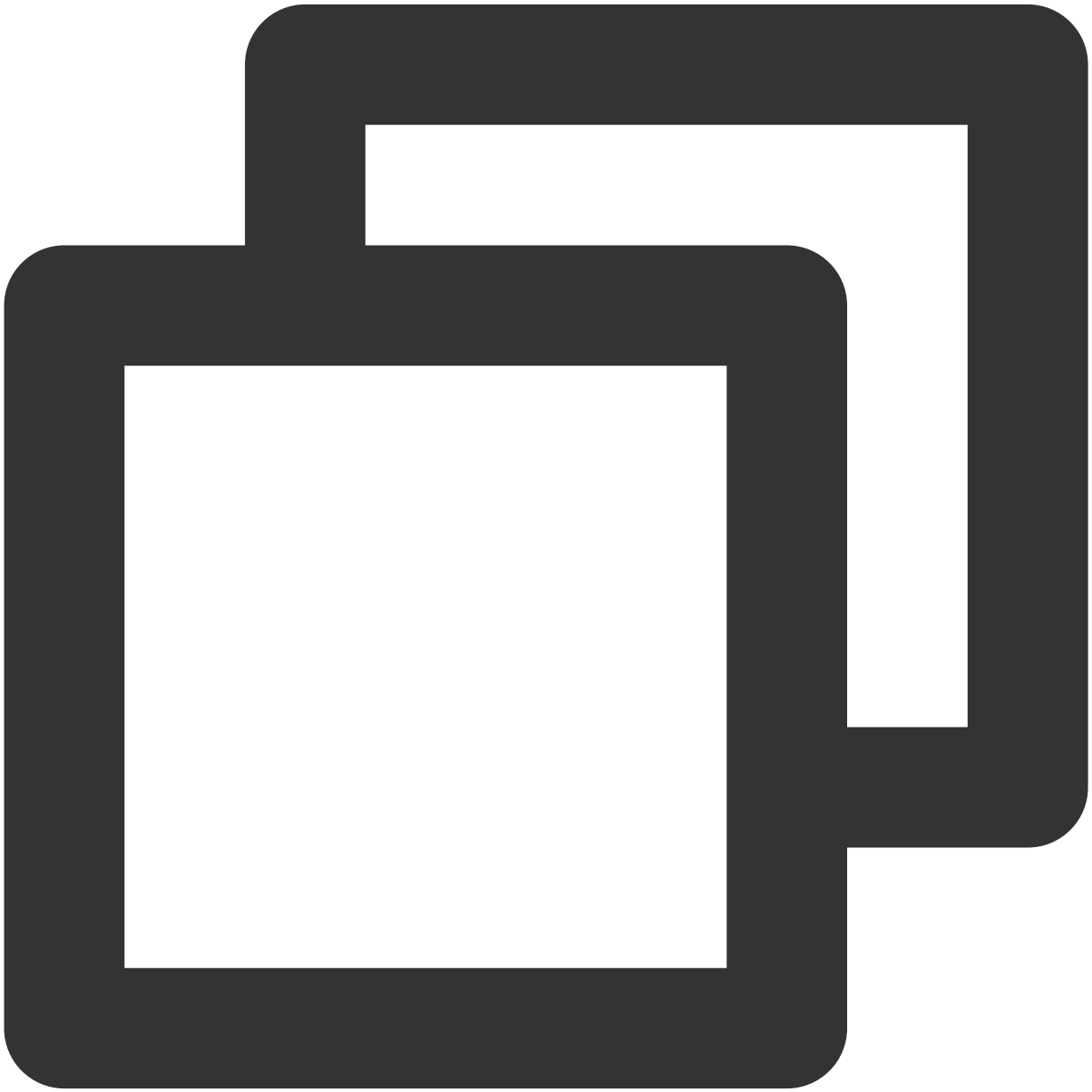


```
var cos = new COS({
  ....
});

/* 自己封装的上传方法 */
function myUpload() {
  // 不需要在每个方法里创建一个 COS SDK 实例
  // var cos = new COS({
  //   ...
  // });
  cos.putObject({
```

```
    ....
  });
}

/* 自己封装的删除方法 */
function myDelete() {
  // 不需要在每个方法里创建一个 COS SDK 实例
  // var cos = new COS({
  //   ...
  // });
  cos.deleteObject({
    ....
  });
}
```

以下是部分常用接口示例，更详细的初始化方法请参见 demo 示例。

创建存储桶

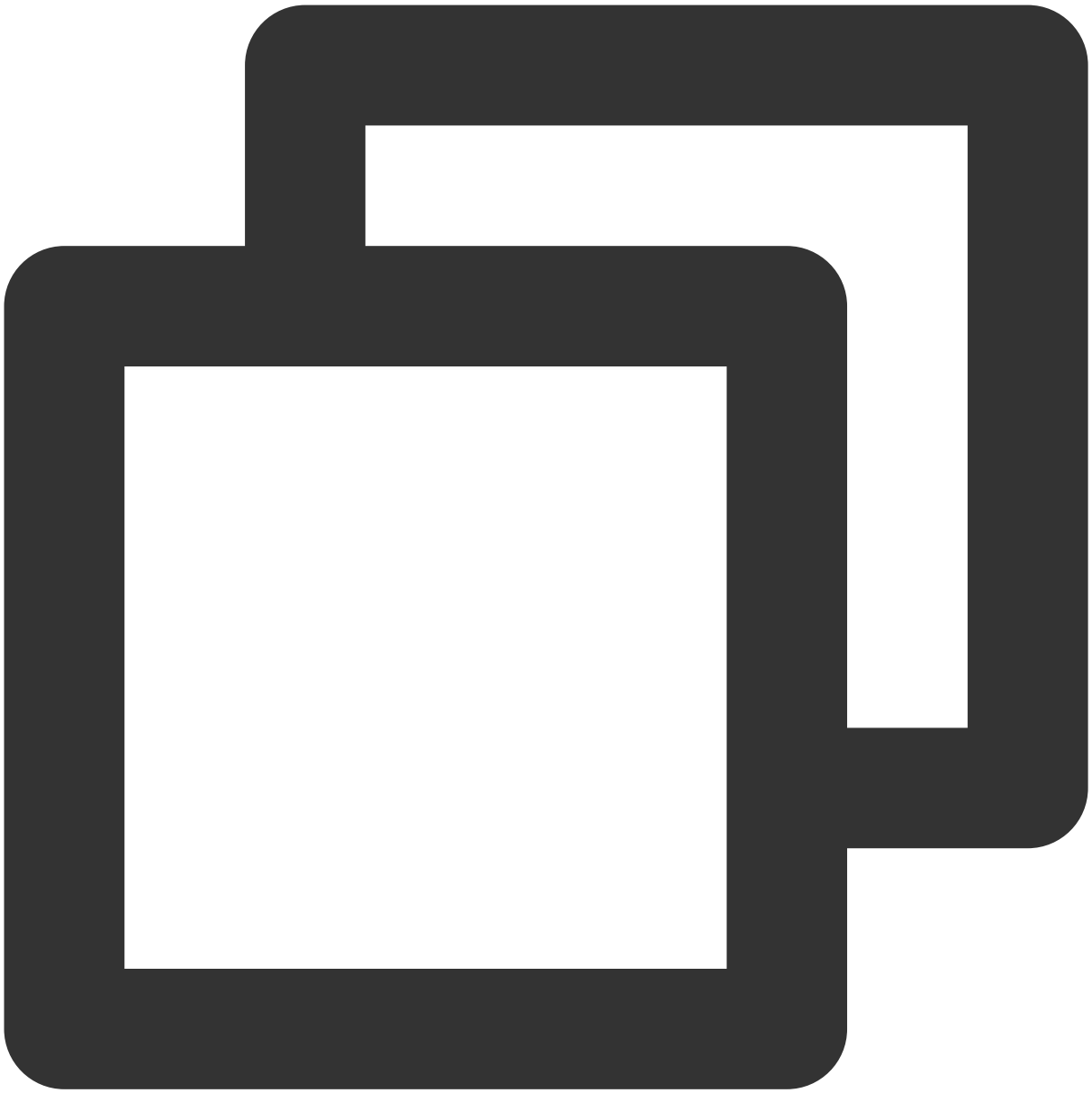


```
cos.putBucket({
    Bucket: 'examplebucket-1250000000',
    Region: 'ap-beijing',
}, function (err, data) {
    console.log(err || data);
});
```

注意：

如果需要在小程序创建存储桶，但存储桶名称未知时，无法将存储桶名称配置为域名白名单，可以使用后缀式调用，相关处理措施请参见 常见问题。

查询存储桶列表



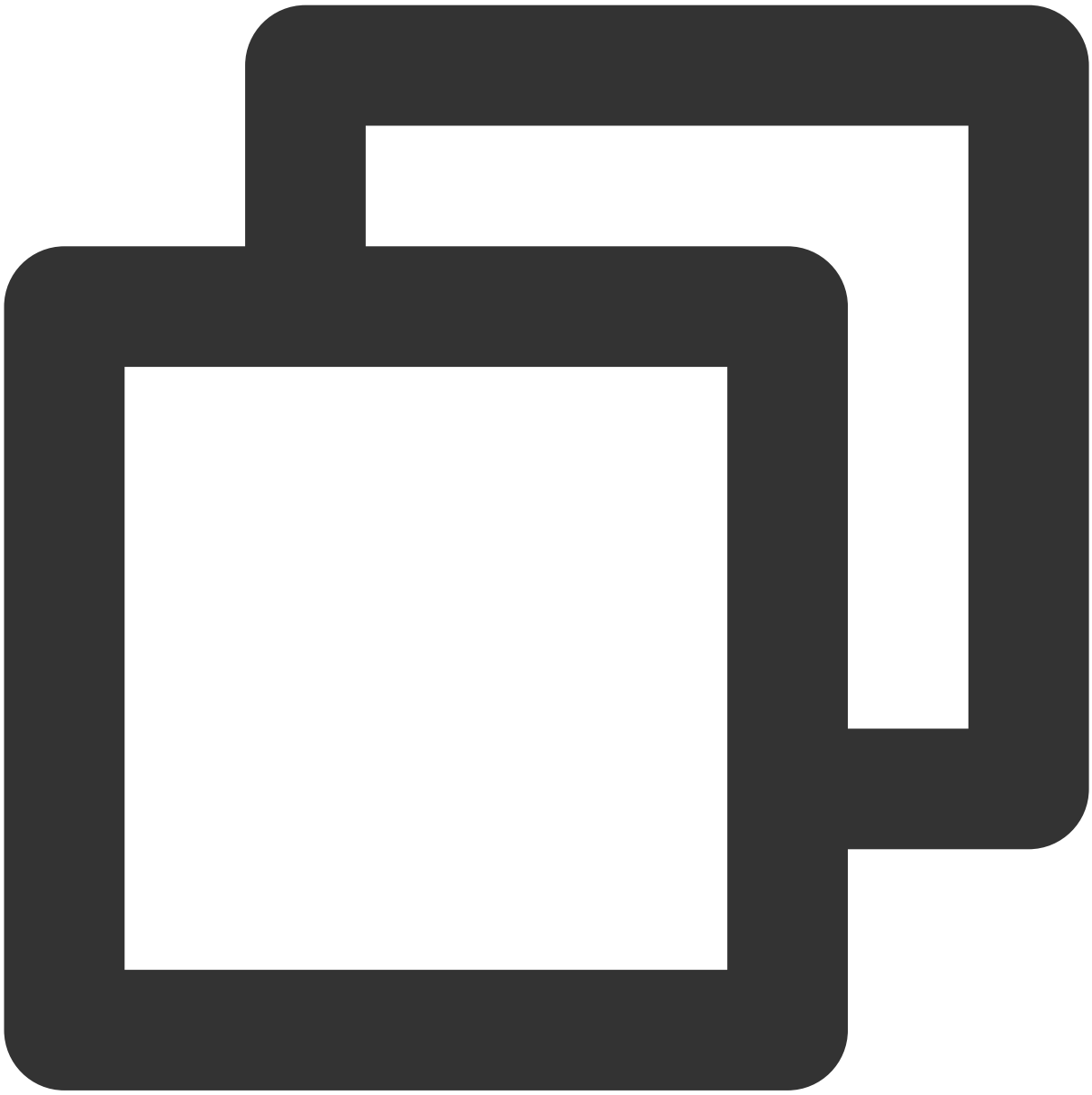
```
cos.getService(function (err, data) {
    console.log(data && data.Buckets);
});
```

上传对象

强烈推荐使用高级上传接口uploadFile，自动针对小文件使用简单上传，大文件使用分块上传，性能更好。详情请参见 高级上传 文档。

若使用临时密钥方式，需同时授权 `简单上传对象` 和 `分块上传` 的权限。请参考授权指引。

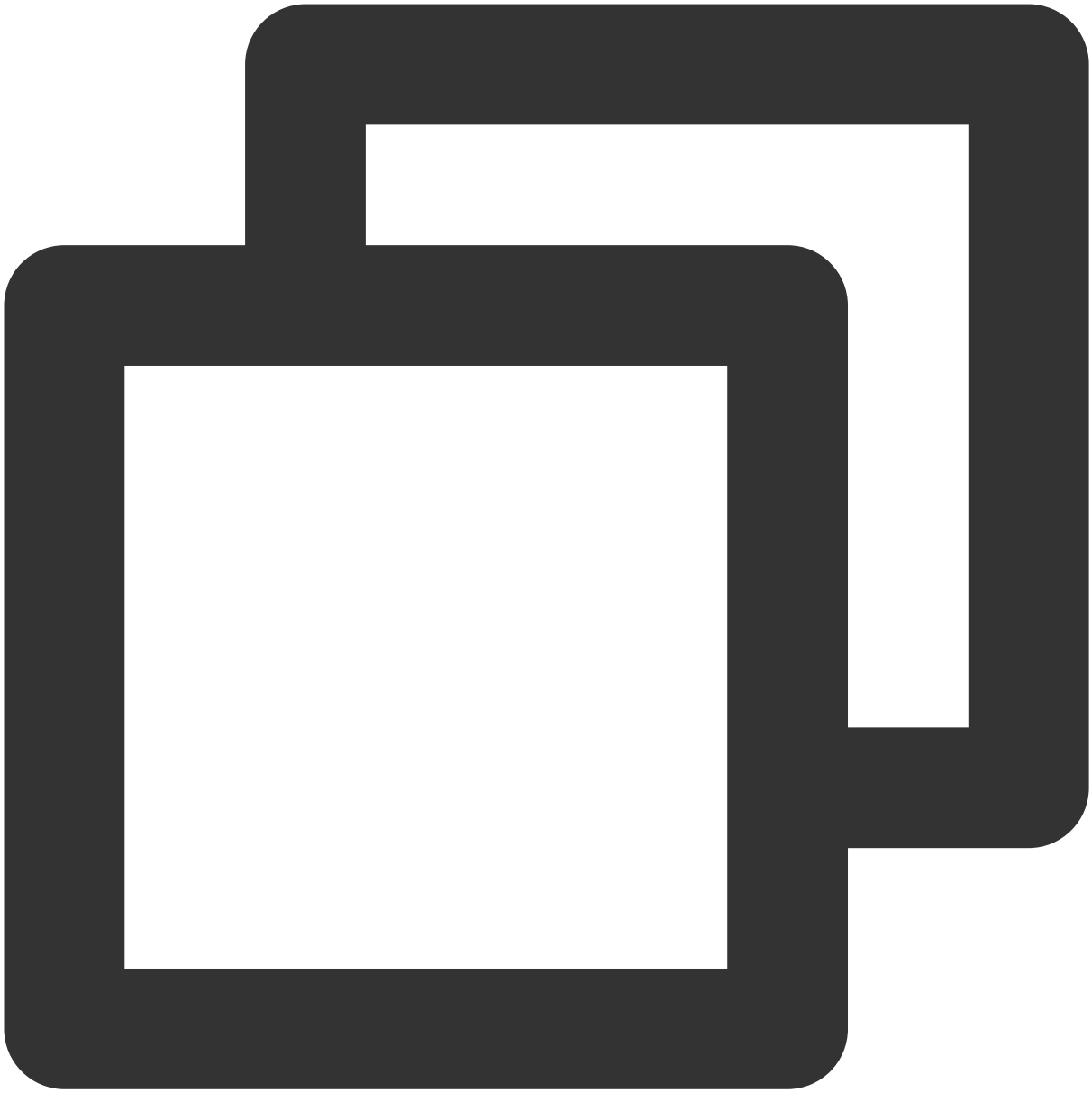
常见上传错误排查，请参考 常见问题。



```
/* 初始化cos */
var cos = new COS({
  // getAuthorization: funciton() {}, // 参考上方初始化
  SimpleUploadMethod: 'putObject', // 强烈建议，高级上传、批量上传内部对小文件做简单上传时使
});

function handleFileInUploading(fileName, filePath) {
  cos.uploadFile({
```

```
        Bucket: 'examplebucket-1250000000', /* 填写自己的bucket，必须字段 */
        Region: 'COS_REGION',      /* 存储桶所在地域，必须字段 */
        Key: fileName,              /* 存储在桶里的对象键（例如:1.jpg，a/b/test.txt，图片.
        FilePath: filePath, /* 上传文件路径，必须字段 */
        SliceSize: 1024 * 1024 * 5,      /* 触发分块上传的阈值，超过5MB使用分块上传，小于5MB使
        onProgress: function(progressData) {
            console.log(JSON.stringify(progressData));
        }
    }, function(err, data) {
        if (err) {
          console.log('上传失败', err);
        } else {
          console.log('上传成功');
        }
    });
}

/* 选择文件,得到临时路径(以选择图片为例，选择其他文件请参考小程序官方api) */
wx.chooseImage({
    count: 1, // 默认9
    sizeType: ['original'], // 可以指定是原图还是压缩图，默认用原图
    sourceType: ['album', 'camera'], // 可以指定来源是相册还是相机，默认二者都有
    success: function (res) {
        var filePath = res.tempFiles[0].path;
        var fileName = filePath.substr(filePath.lastIndexOf('/') + 1);
        handleFileInUploading(fileName, filePath);
    }
});
```

查询对象列表

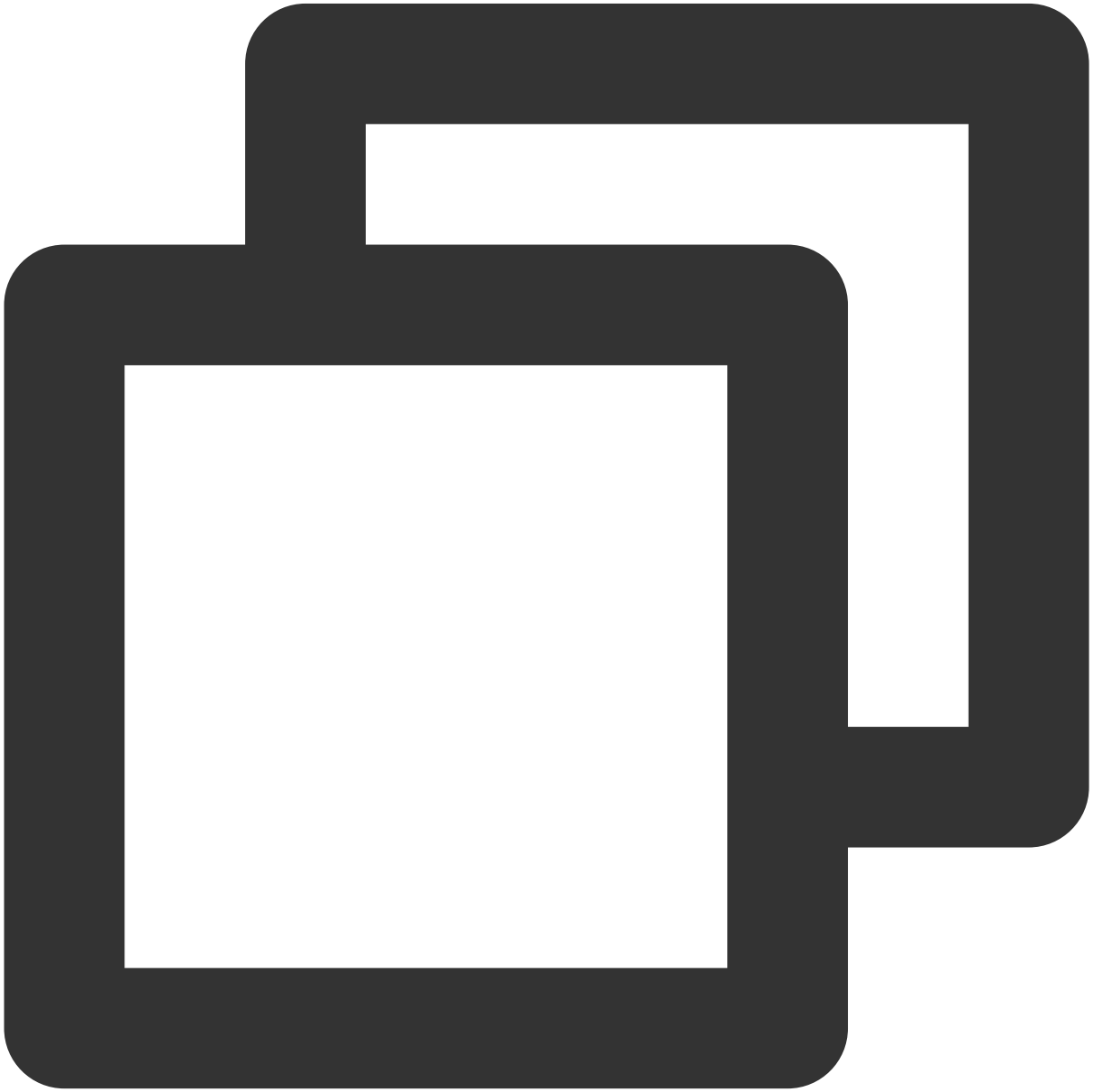


```
cos.getBucket({
    Bucket: 'examplebucket-1250000000',
    Region: 'ap-beijing',
    Prefix: 'exampledir/', // 这里传入列出的文件前缀
}, function (err, data) {
    console.log(err || data.Contents);
});
```

下载对象

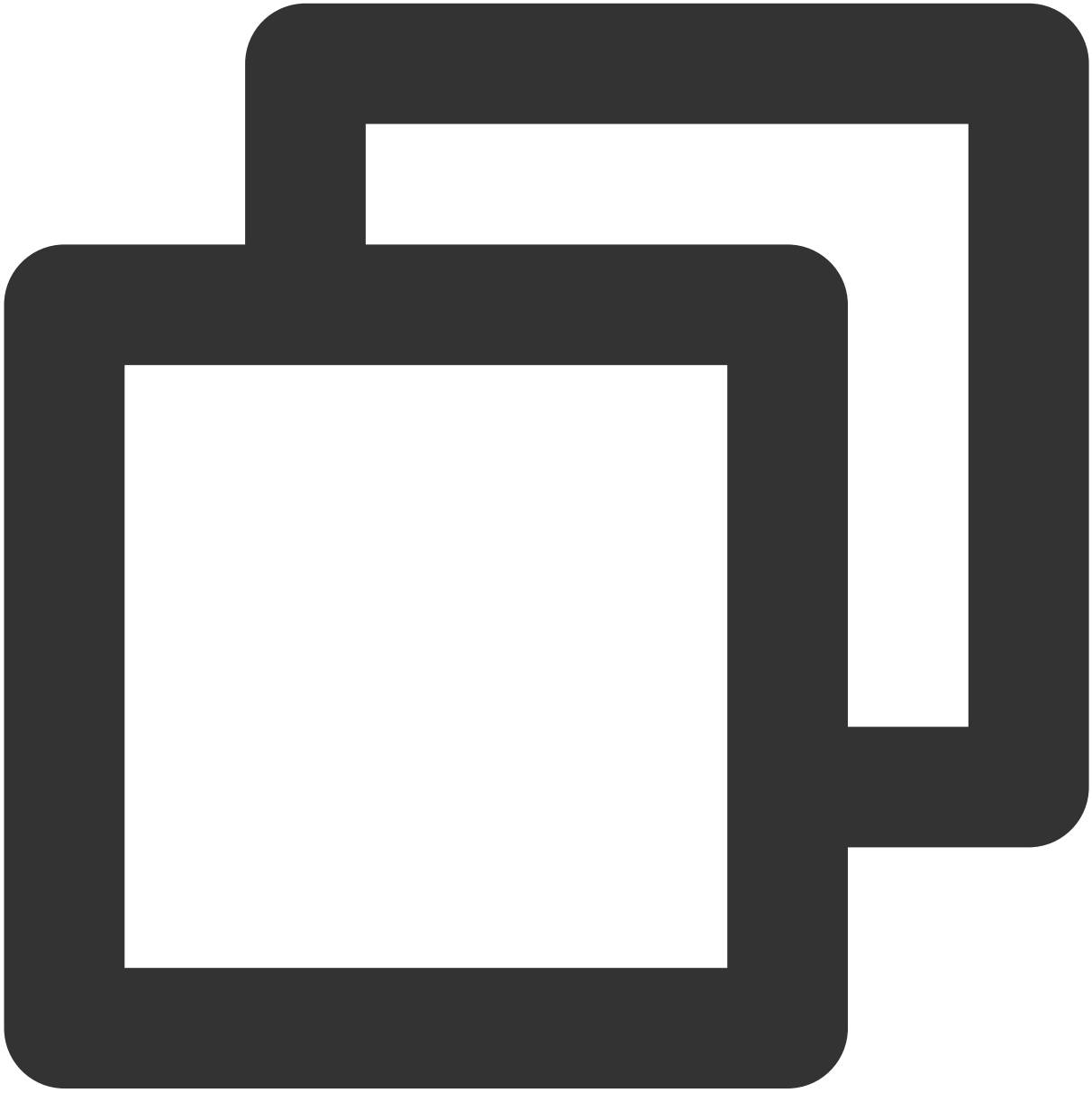
注意：

该接口用于读取对象内容，如果需要发起浏览器下载文件，可以通过 cos.getObjectUrl 获取 url 再触发浏览器下载，具体参见 预签名 URL 文档。



```
cos.getObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'ap-beijing',
    Key: 'exampleobject.txt',
}, function (err, data) {
    console.log(err || data.Body);
});
```

删除对象



```
cos.deleteObject({
    Bucket: 'examplebucket-1250000000',
    Region: 'ap-beijing',
    Key: 'picture.jpg',
}, function (err, data) {
    console.log(err || data);
});
```

版权所有：腾讯云计算（北京）有限责任公司

图片处理
基础图片处理

最近更新时间：2024-03-13 11:41:11

简介

本文档提供关于基础图片处理的相关的 API 概览以及 SDK 示例代码。

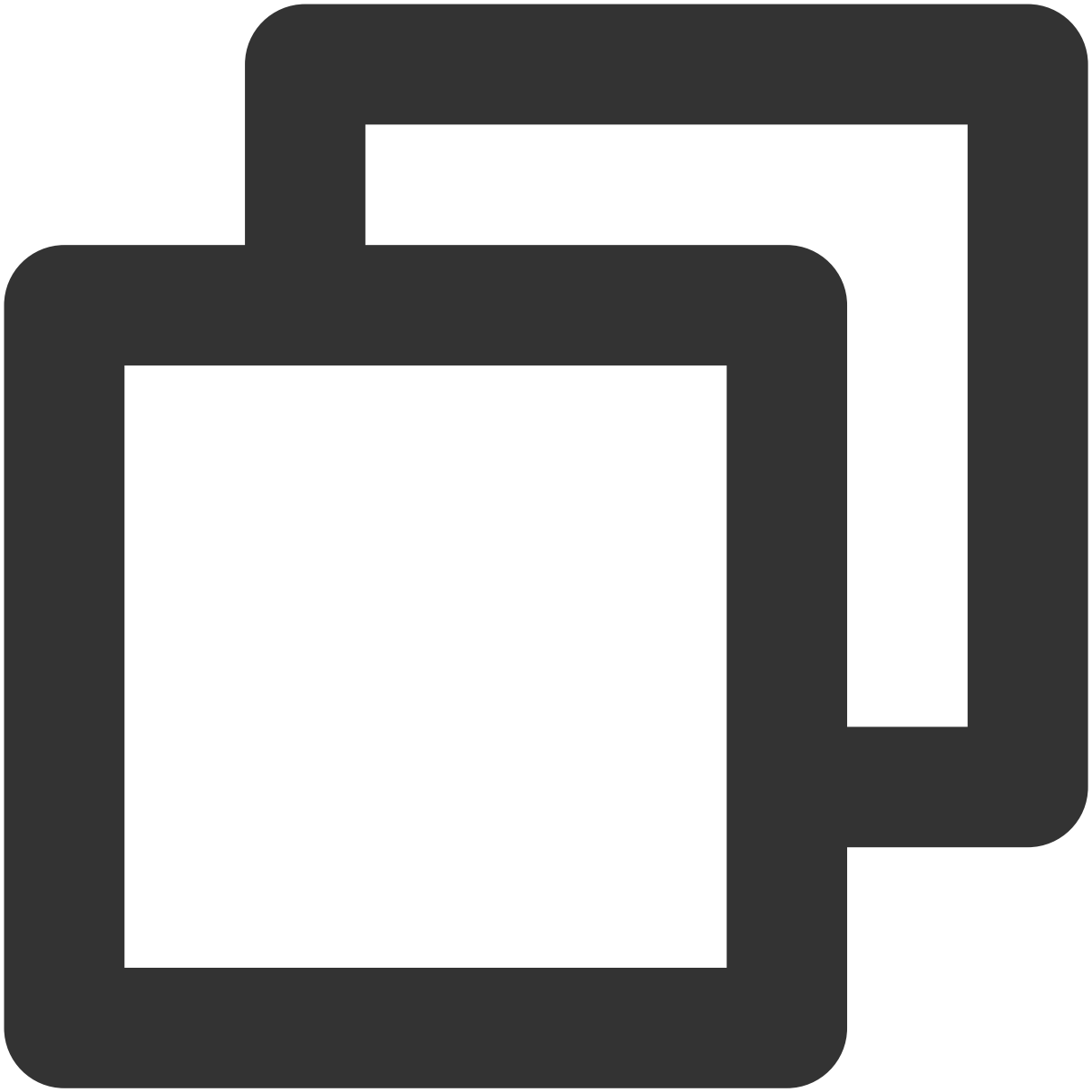
服务	功能	说明
基础图片处理服务	缩放	等比缩放、设定目标宽高缩放等多种方式
	裁剪	普通裁剪、缩放裁剪、内切圆、人脸智能裁剪
	旋转	自适应旋转、普通旋转
	格式转换	格式转换、GIF 格式优化、渐进显示
	质量变换	针对 JPG 和 WEBP 图片进行质量变换
	高斯模糊	对图片进行模糊处理
	锐化	对图片进行锐化处理
	添加水印	图片水印、文字水印
	获取图片信息	基本信息、EXIF 信息、主色调
	去除元信息	包括 EXIF 信息
	快速缩略模板	快速实现图片格式转换、缩略、剪裁等功能，生成缩略图

上传时使用图片处理

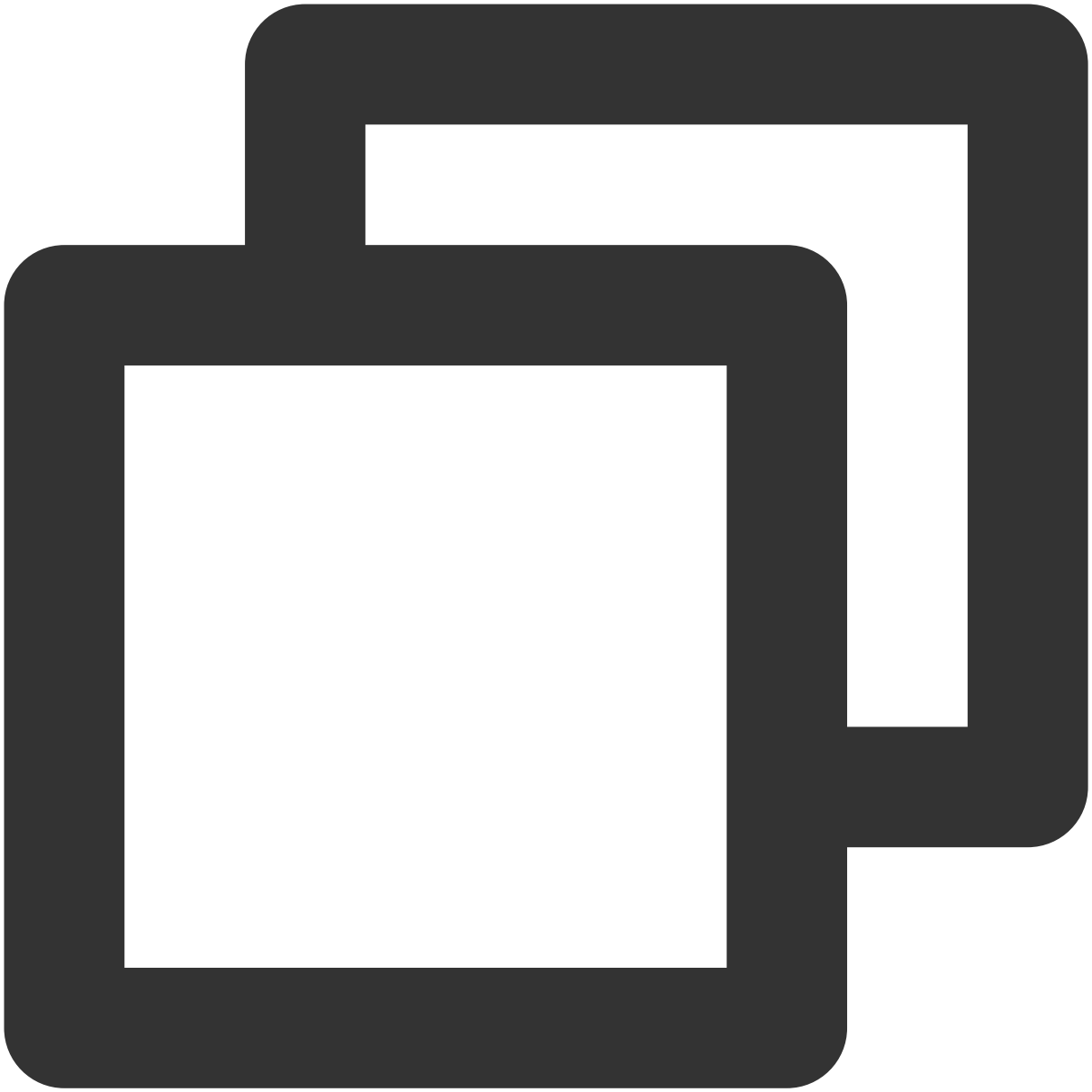
下面示例展示了如何在上传图片时自动实现图片处理。
图片上传完成后，COS 会存储原始图片和已处理过的图片。后续用户可以通过普通的下载请求获取处理结果。

示例代码

版权所有：腾讯云计算（北京）有限责任公司



```
<view>
  <button type="primary" bindtap="button">上传时使用图片处理</button>
</view>
```



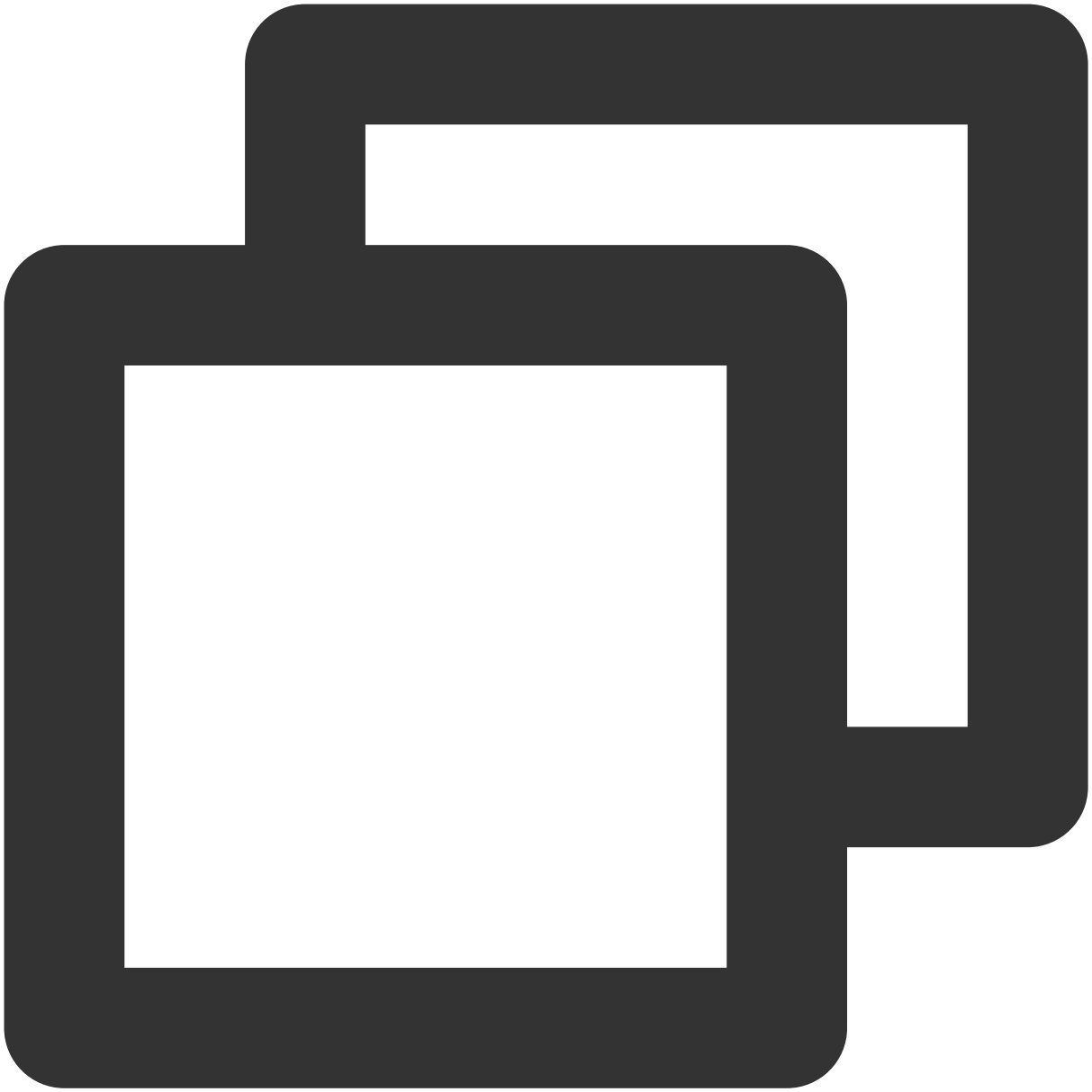
```
Page({
  button: function () {
    wx.chooseMessageFile({
      count: 10,
      type: 'all',
      success: function (res) {
        var file = res.tempFiles[0];
        wxfs.readFile({
          filePath: file.path,
          success: function (res) {
            cos.putObject(
```

```
            {
              Bucket: 'examplebucket-1250000000',
              Region: 'COS_REGION',
              Key: file.name,
              Body: res.data,
              Headers: {
                // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
                'Pic-Operations':
                  '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jpg",
              },
            },
            (err, data) => {
              console.log(err || data);
            },
          );
        },
        fail: (err) => console.error(err),
      });
    },
    fail: (err) => console.error(err),
  });
},
});
```

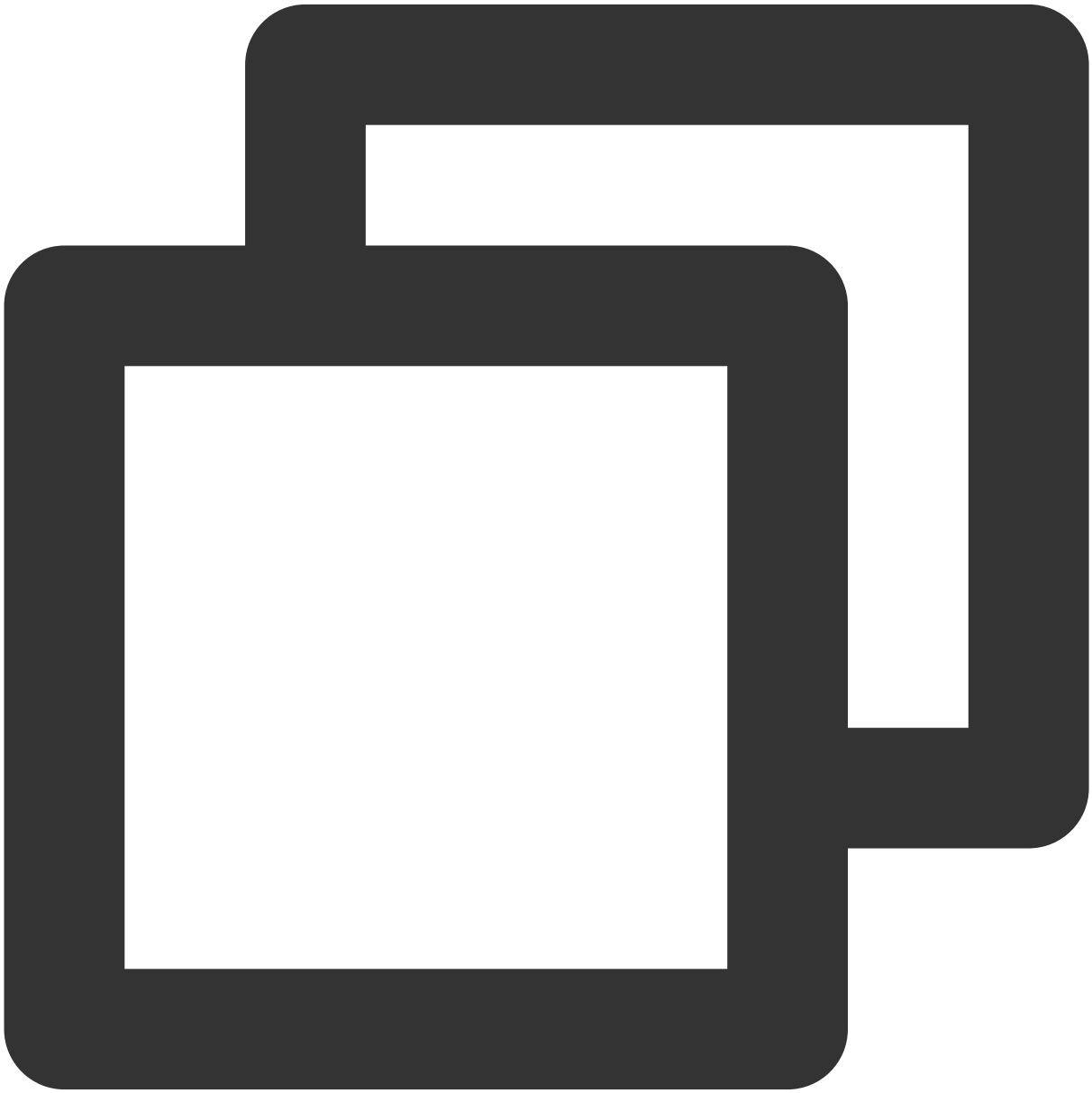
对云上数据进行图片处理

下面示例展示了如何对已存储在 COS 的图片进行相应处理操作，并将结果存入到 COS。

示例代码



```
<view>
  <button type="primary" bindtap="button">对云上数据进行图片处理</button>
</view>
```



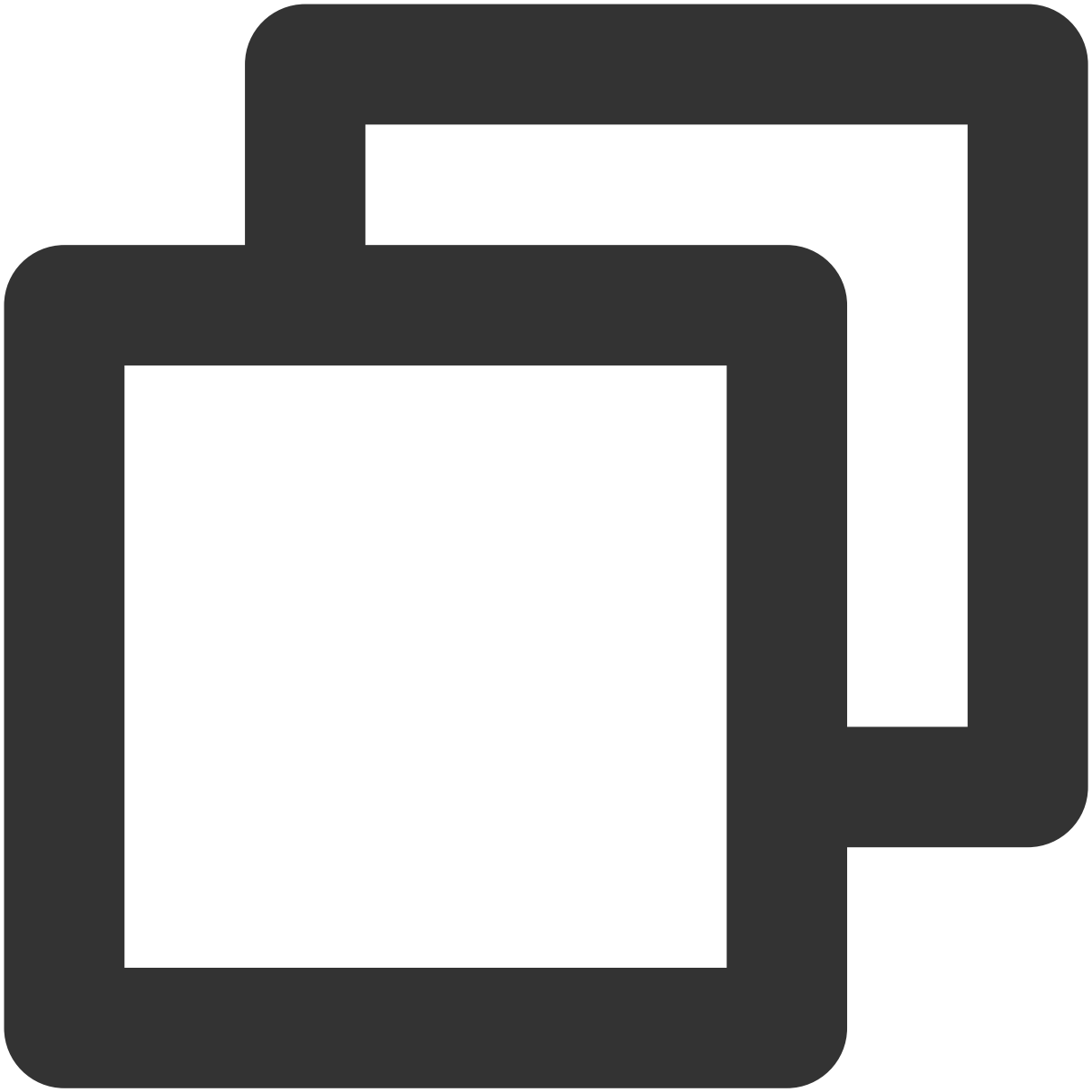
```
Page({
  button: function () {
    cos.request(
      {
        Bucket: 'examplebucket-1250000000',
        Region: 'COS_REGION',
        Key: 'exampleImage',
        Method: 'POST',
        Action: 'image_process',
        Headers: {
          // 通过 imageMogr2 接口使用图片缩放功能：指定图片宽度为 200，宽度等比压缩
```

```
            'Pic-Operations':
              '{"is_pic_info": 1, "rules": [{"fileid": "desample_photo.jpg", "rule":
          },
        },
        (err, data) => {
          console.log(err || data);
        },
      );
    },
  });
```

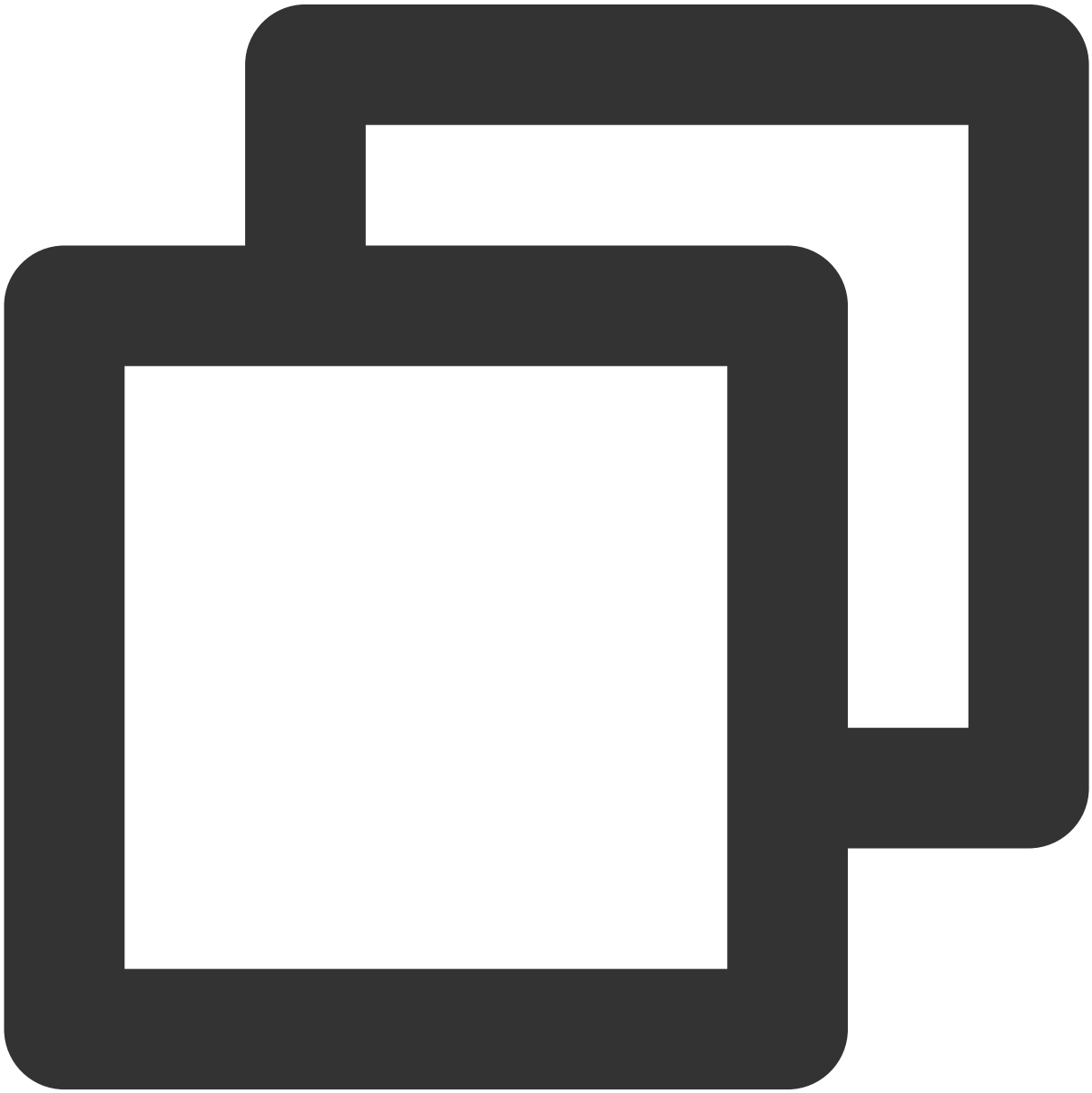
下载时使用图片处理

下面示例展示了如何在下载图片时实现图片处理。

示例代码



```
<view>
  <button type="primary" bindtap="button">下载时使用图片处理</button>
</view>
```

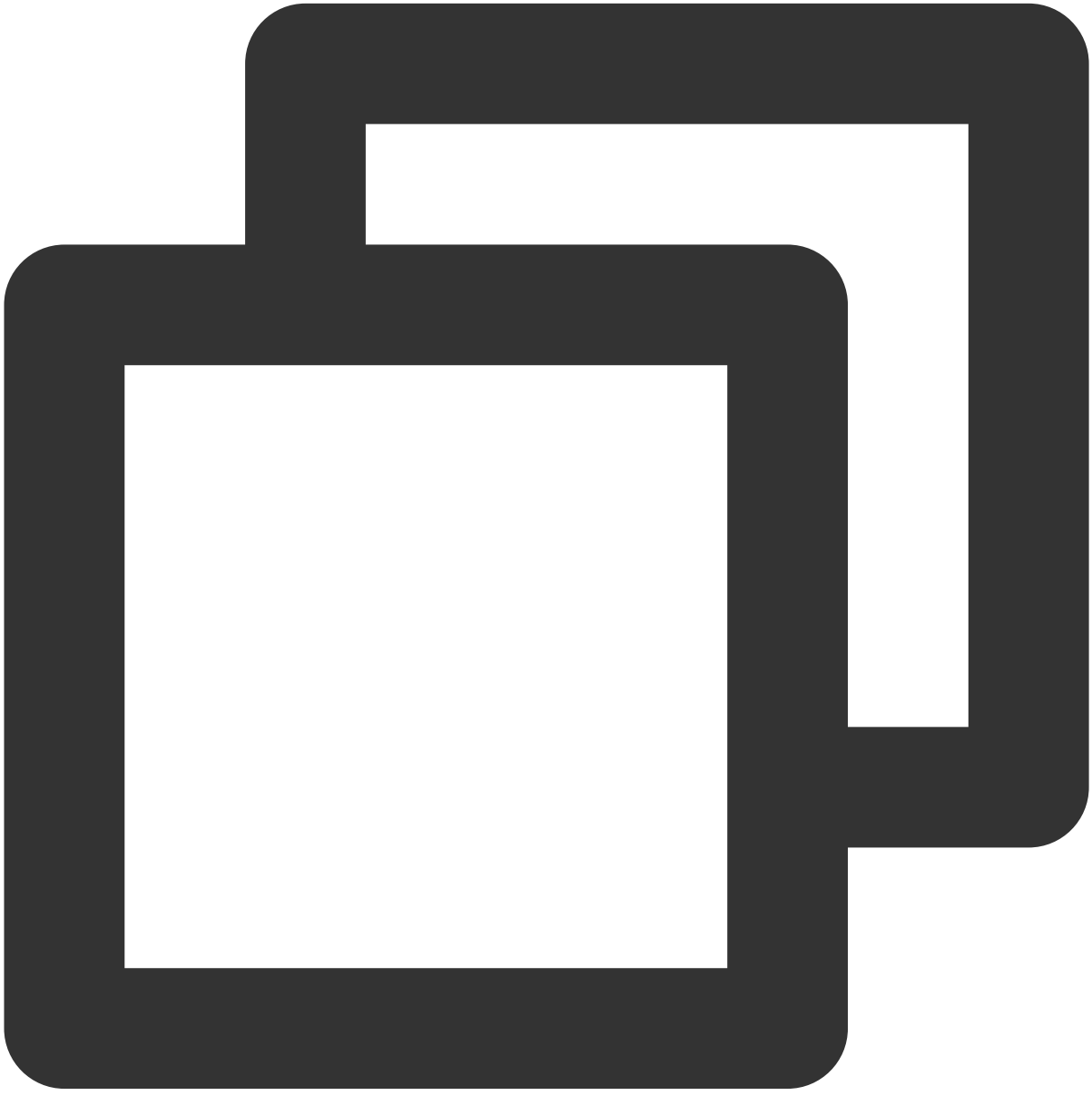


```
Page({
  button: function () {
    cos.getObject(
      {
        Bucket: 'examplebucket-1250000000',
        Region: 'COS_REGION',
        Key: 'exampleImage',
        QueryString: `imageMogr2/thumbnail/200x/`,
      },
      (err, data) => {
        console.log(err || data);
```

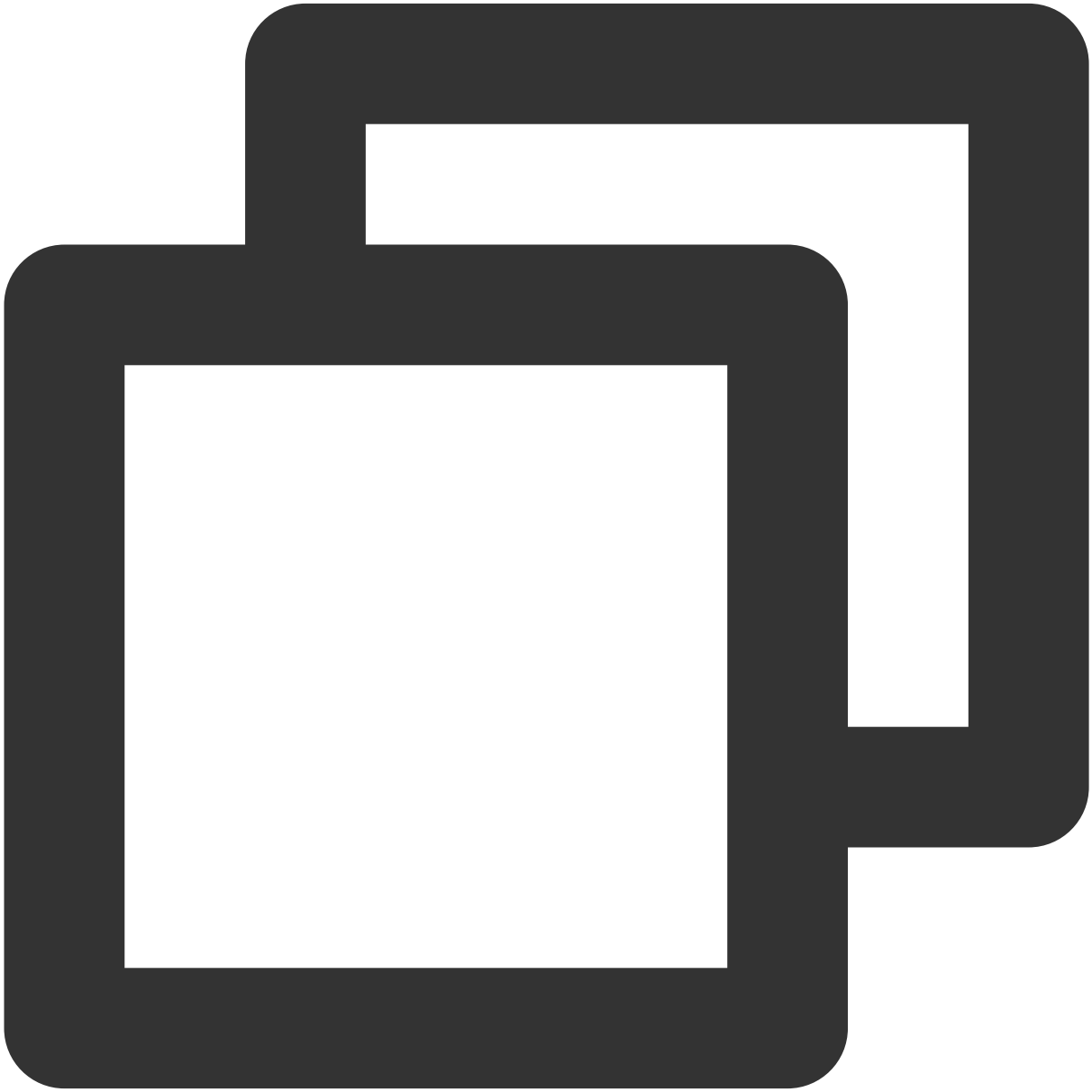
```
        },
      );
    },
});
```

生成带图片处理参数的签名 URL

示例代码



```
<view>
  <button type="primary" bindtap="sign">生成带图片处理参数的 URL，带签名</button>
  <button type="primary" bindtap="unsign">生成带图片处理参数的 URL，不带签名</button>
</view>
```



```
Page({
  sign: function () {
    // 生成带图片处理参数的文件签名 URL，过期时间设置为 30 分钟。
    cos.getObjectUrl(
      {
```

```
        Bucket: 'examplebucket-1250000000',
        Region: 'COS_REGION',
        Key: 'exampleobject',
        Query: {
          `imageMogr2/thumbnail/200x/`: ''
        },
        Expires: 1800,
        Sign: true,
      },
      (err, data) => {
        if (data) {
          console.log(data.URL);
        }
      },
    );
  },
  unsign: function () {
    // 生成带图片处理参数的文件 URL，不带签名。
    cos.getObjectUrl(
      {
        Bucket: 'examplebucket-1250000000',
        Region: 'COS_REGION',
        Key: 'exampleobject',
        Query: {
          `imageMogr2/thumbnail/200x/`: ''
        },
        Sign: false,
      },
      (err, data) => {
        if (data) {
          console.log(data.URL);
        }
      },
    );
  },
});
```

盲水印

最近更新时间：2024-03-13 11:41:12

简介

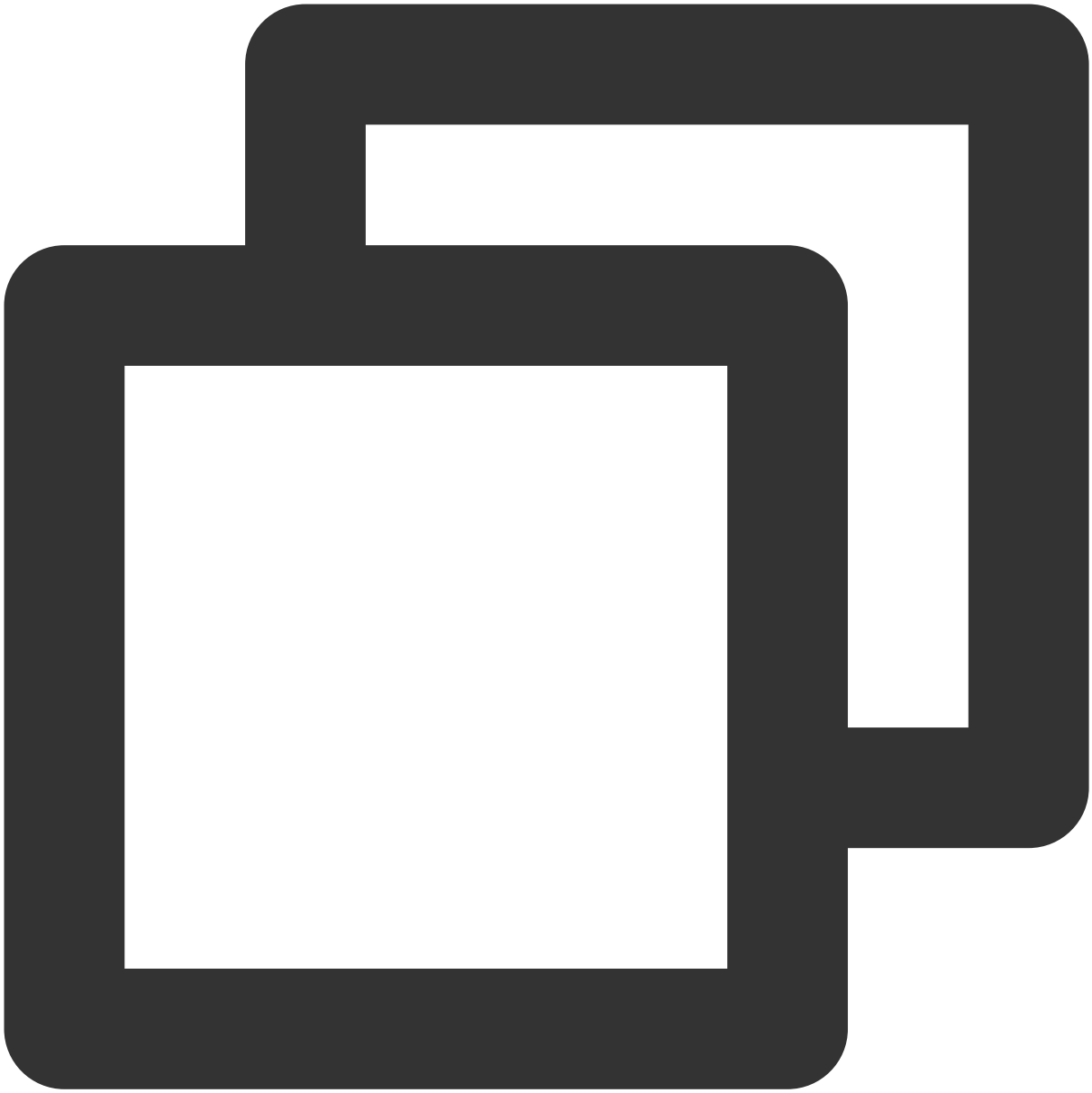
本文档提供关于盲水印的相关的 API 概览以及 SDK 示例代码。
关于盲水印 API 文档请参见盲水印。

添加盲水印

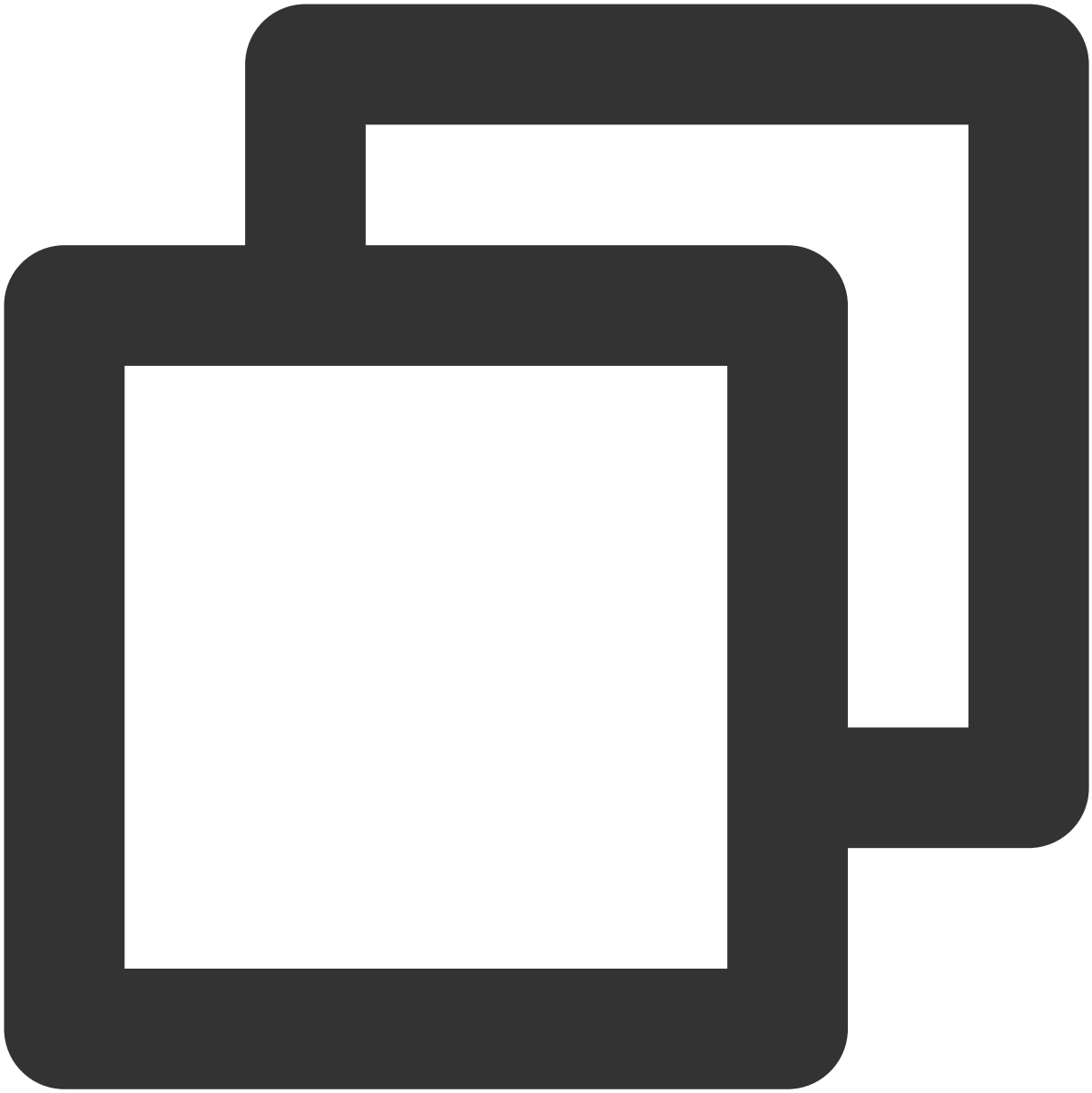
功能说明

盲水印支持在上传时添加和下载时添加。

示例代码一：上传时添加盲水印



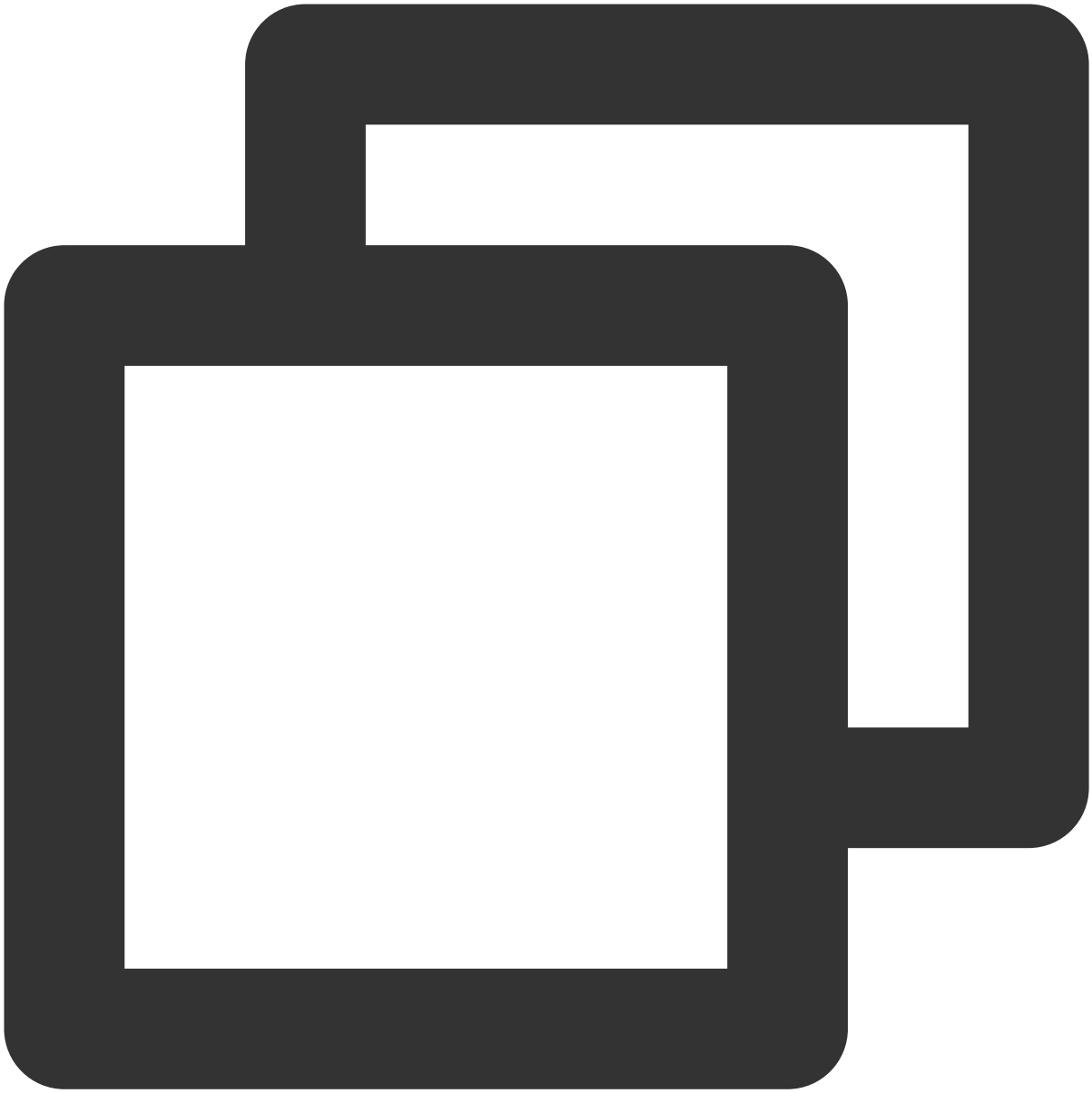
```
<view>
  <button type="primary" bindtap="button">上传时添加盲水印</button>
</view>
```



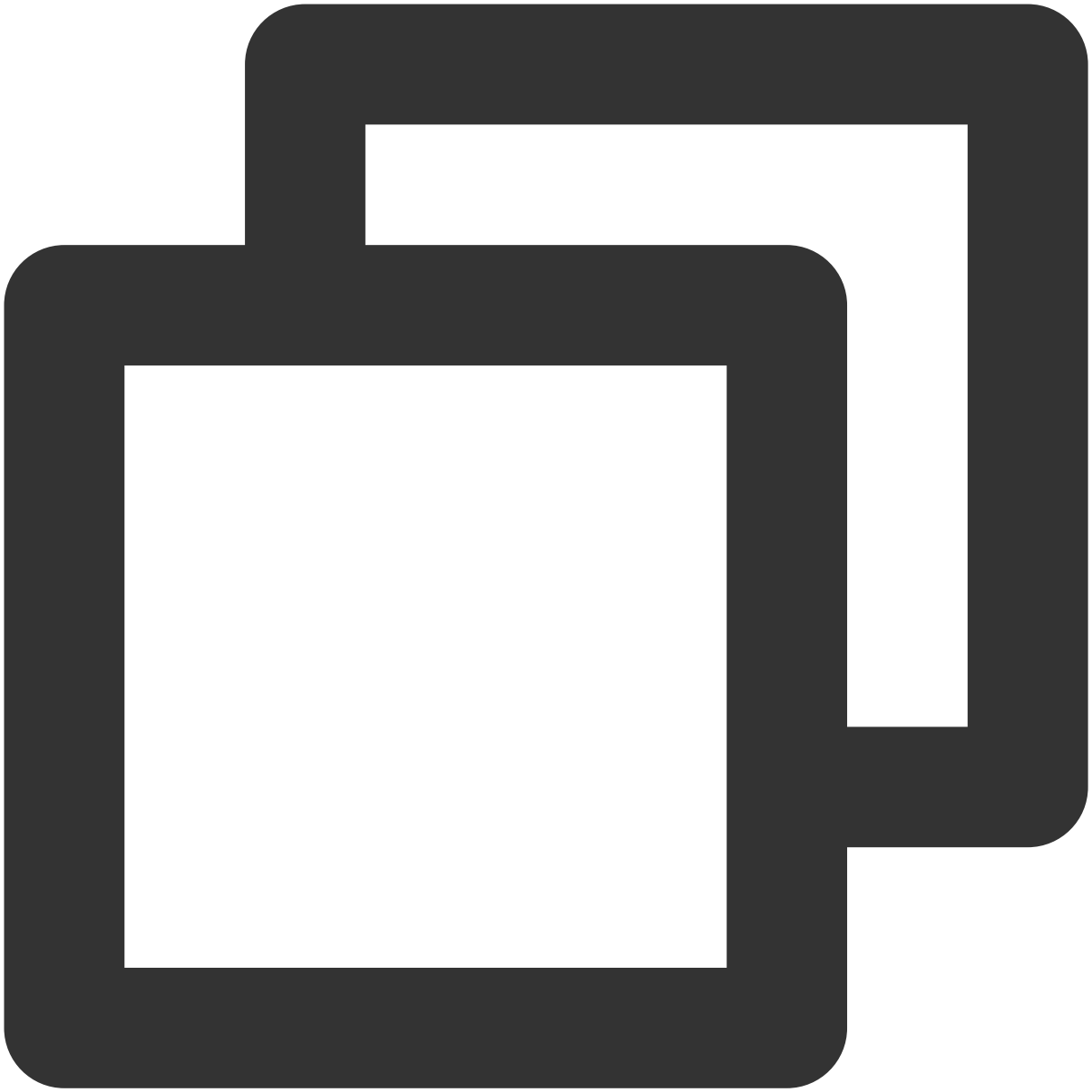
```
Page({
  button: function () {
    wx.chooseMessageFile({
      count: 10,
      type: 'all',
      success: function (res) {
        var file = res.tempFiles[0];
        wxfs.readFile({
          filePath: file.path,
          success: function (res) {
            cos.putObject(
```

```
                {
                  Bucket: 'examplebucket-1250000000',
                  Region: 'COS_REGION',
                  Key: file.name,
                  Body: res.data,
                  Headers: {
                    // 万象持久化接口，上传时持久化
                    'Pic-Operations': '{"is_pic_info": 1, "rules": [{"fileid": "desam
                  },
                },
                (err, data) => {
                  console.log(err || data);
                },
              );
            },
            fail: (err) => console.error(err),
          });
        },
        fail: (err) => console.error(err),
      });
    },
});
```

示例代码二：下载时添加盲水印



```
<view>
  <button type="primary" bindtap="button">下载时添加盲水印</button>
</view>
```



```
Page({
  button: function () {
    cos.getObject(
      {
        Bucket: 'examplebucket-1250000000',
        Region: 'COS_REGION',
        Key: 'exampleImage',
        QueryString: `watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQu
      },
      (err, data) => {
        console.log(err || data);
```

```
        },
      );
    },
  });
```

版权所有：腾讯云计算（北京）有限责任公司

媒体处理
媒体 bucket 接口

最近更新时间：2024-03-13 11:41:11

简介

本文档提供关于媒体 bucket 接口的 API 概览和 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体处理开通情况	用于查询已经开通媒体处理功能的存储桶

查询媒体处理开通情况

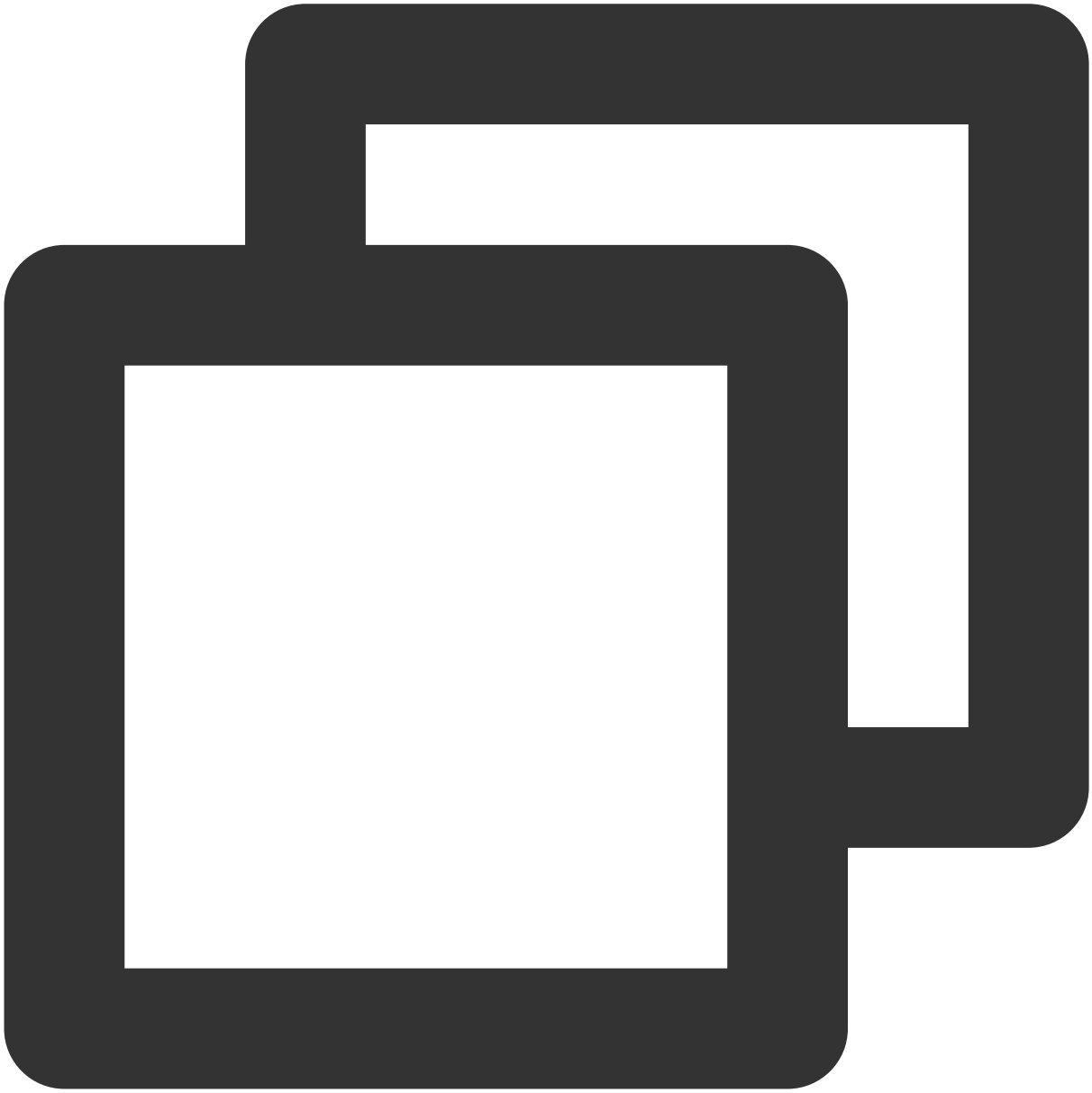
功能说明

用于查询已经开通媒体处理功能的存储桶。

注意：

COS 小程序 SDK 版本需要大于等于 v1.1.1。

使用示例



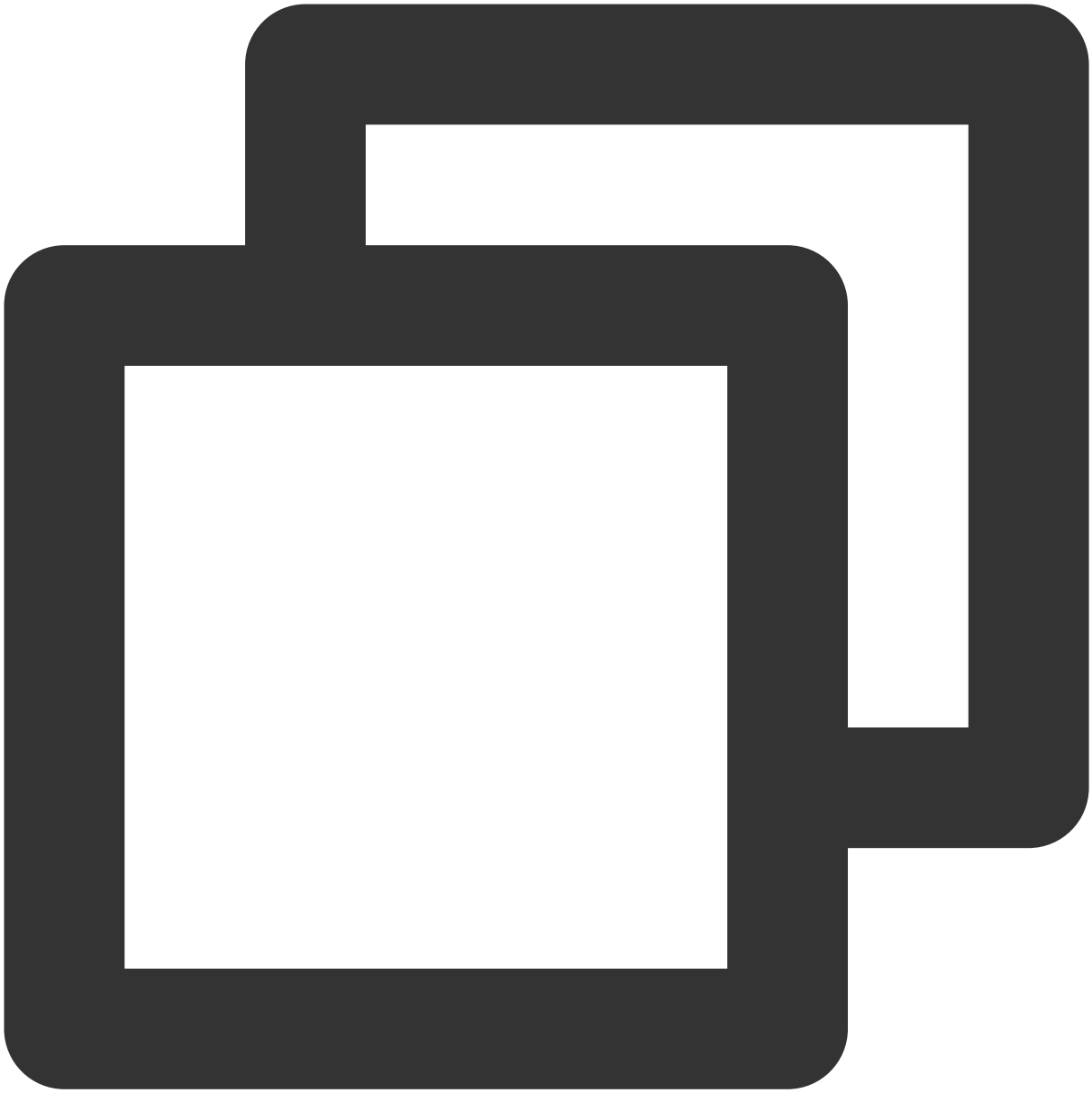
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
var host = 'ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/mediabucket';
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
```

```
    Key: 'mediabucket', /** 固定值，必须 */
    Url: url,
    Query: {
        pageNumber: '1', /** 第几页，非必须 */
        pageSize: '10', /** 每页个数，非必须 */
        // regions: 'ap-chengdu', /** 地域信息，例如'ap-beijing'，支持多个值用逗号分隔如'
        // bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-125000000
        // bucketName: 'test', /** 存储桶名称前缀，前缀搜索，例如'test'，支持多个值用逗号分
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
PageNumber	第几页	String	否
PageSize	每页个数	String	否
Regions	地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串，支持中国大陆地域，详情请参见 地域与域名	String	否
BucketNames	存储桶名称，以“,”分隔，支持多个存储桶，精确搜索	String	否
BucketName	存储桶名称前缀，前缀搜索	String	否

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- Response	响应结果	Object
- - RequestId	请求的唯一 ID	String
- - TotalCount	媒体 Bucket 总数	String
- - PageNumber	当前页数，同请求中的 pageNumber	String
- - PageSize	每页个数，同请求中的 pageSize	String
- - MediaBucketList	媒体 Bucket 列表	Object
- - - BucketId	存储桶 ID	String
- - - Name	存储桶名称，同 BucketId	String
- - - Region	所在的地域	String
- - - CreateTime	创建时间	String

媒体信息接口

最近更新时间：2024-03-13 11:41:12

简介

本文档提供关于媒体信息接口的 API 概览和 SDK 示例代码。

注意：

COS 小程序 SDK 版本需要大于等于 v1.1.1。

API	操作名	操作描述
GetMediaInfo	查询文件信息	用于查询媒体文件的信息

查询文件信息

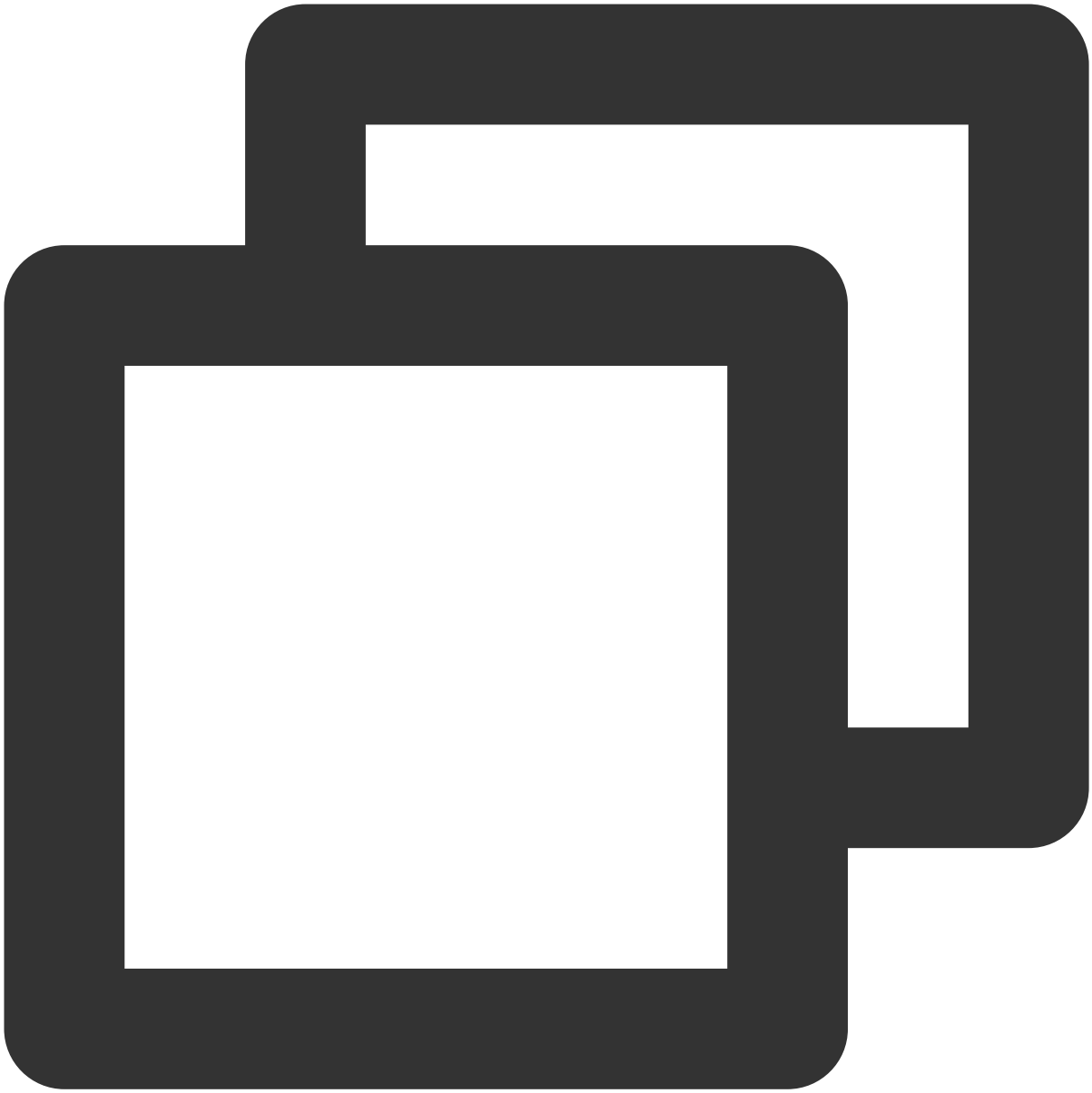
功能说明

用于查询媒体文件的信息。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



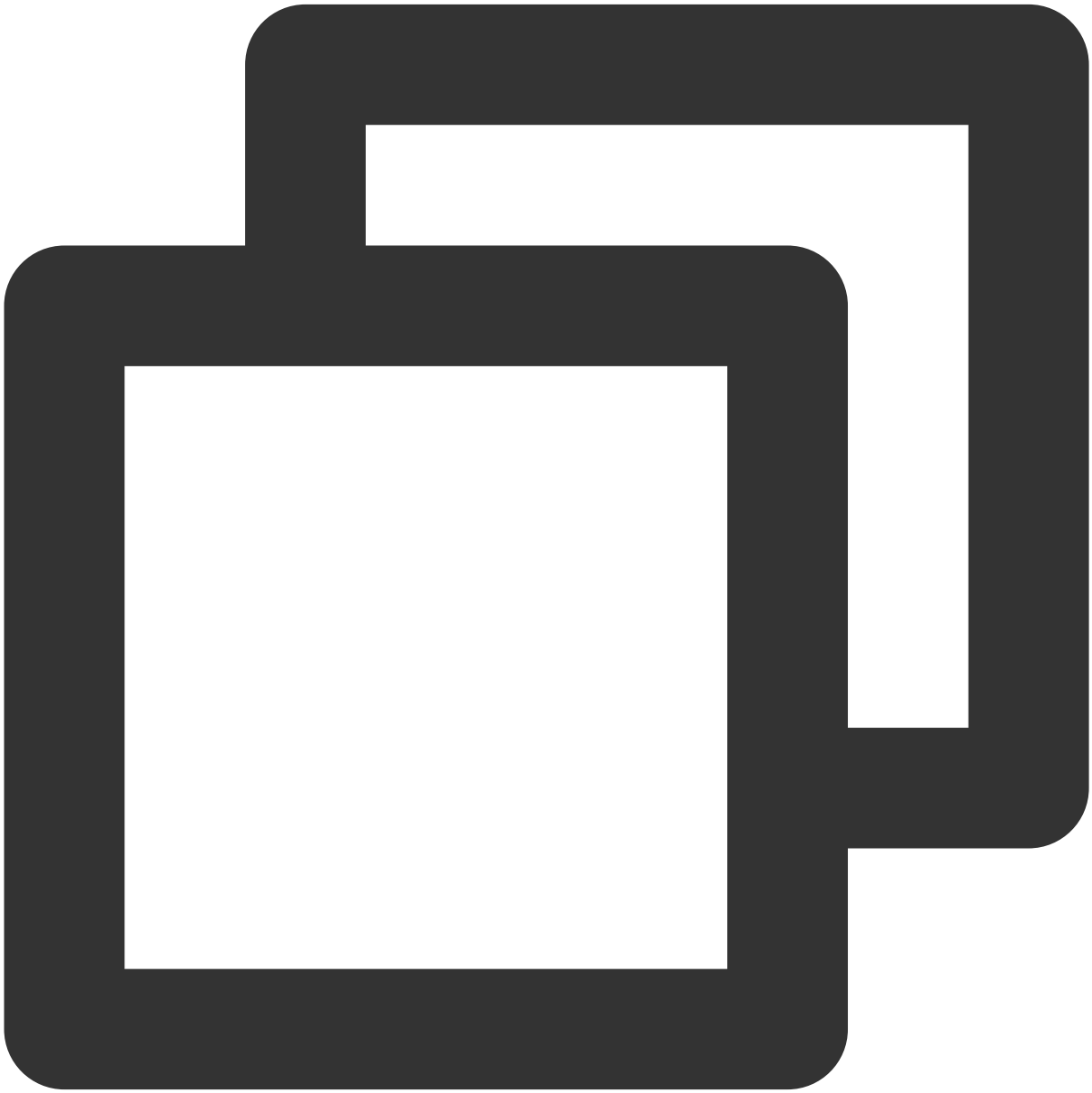
```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

```
        'ci-process': 'videoinfo' /** 固定值，必须 */
    }
}, function (err, data) {
    console.log(err || data);
});
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	固定值: mediabucket	String	是
ci-process	操作类型，固定使用 videoinfo	String	是

回调函数说明



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Response	响应结果	Object
- - MediaInfo	媒体的详细信息	Object
- - - Stream	流信息	Object
- - - - Video	视频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - Profile	视频编码档位	String
- - - - - Height	视频高，单位 px	Number
- - - - - Width	视频宽，单位 px	Number
- - - - - HasBFrame	是否有B帧。1表示有，0表示无	Number
- - - - - RefFrames	视频编码的参考帧个数	Number
- - - - - Sar	采样宽高比	String
- - - - - Dar	显示宽高比	String
- - - - - PixFormat	像素格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - FieldOrder	场的顺序	String
- - - - - Level	视频编码等级	Number
- - - - - Fps	视频帧率	Number
- - - - - AvgFps	平均帧率	String
- - - - - Timebase	时基	String
- - - - - StartTime	视频开始时间，单位为秒	Number
- - - - - Duration	视频时长，单位为秒	Number
- - - - - Bitrate	比特率，单位为 kbps	Number
- - - - - NumFrames	总帧数	Number
- - - - - Language	语言	String
- - - - Audio	音频信息	Container
- - - - - Index	该流的编号	Number
- - - - - CodecName	编解码格式名字	String
- - - - - CodecLongName	编解码格式的详细名字	String
- - - - - CodecTimeBase	编码时基	String
- - - - - CodecTagString	编码标签名	String
- - - - - CodecTag	编码标签	String
- - - - - SampleFmt	采样格式	String
- - - - - SampleRate	采样率	Number
- - - - - Channel	通道数量	Number
- - - - - ChannelLayout	通道格式	String

版权所有：腾讯云计算（北京）有限责任公司

- - - - - Timebase	时基	String
- - - - - StartTime	音频开始时间，单位秒	Number
- - - - - Duration	音频时长，单位秒	Number
- - - - - Bitrate	比特率，单位 kbps	Number
- - - - - Language	语言	String
- - - - Subtitle	字幕信息	Container
- - - - - Index	该流的编号	Number
- - - - - Language	语言，und 表示无查询结果	String
- - - Format	格式信息	Object
- - - - NumStream	Stream（包含 Video、Audio、Subtitle）的数量	Number
- - - - NumProgram	节目的数量	Number
- - - - FormatName	容器格式名字	String
- - - - FormatLongName	容器格式的详细名字	String
- - - - StartTime	起始时间，单位为秒	Number
- - - - Duration	时长，单位为秒	Number
- - - - Bitrate	比特率，单位为 kbps	Number
- - - - Size	大小，单位为 Byte	Number

版权所有：腾讯云计算（北京）有限责任公司

媒体截图接口

最近更新时间：2024-03-13 11:41:12

简介

本文档提供关于媒体截图接口的 API 概览和 SDK 示例代码。

注意：

COS 小程序 SDK 版本需要大于等于 v1.1.1。

API	操作名	操作描述
GetSnapshot	查询截图	用于查询媒体文件在某个时间的截图

查询截图

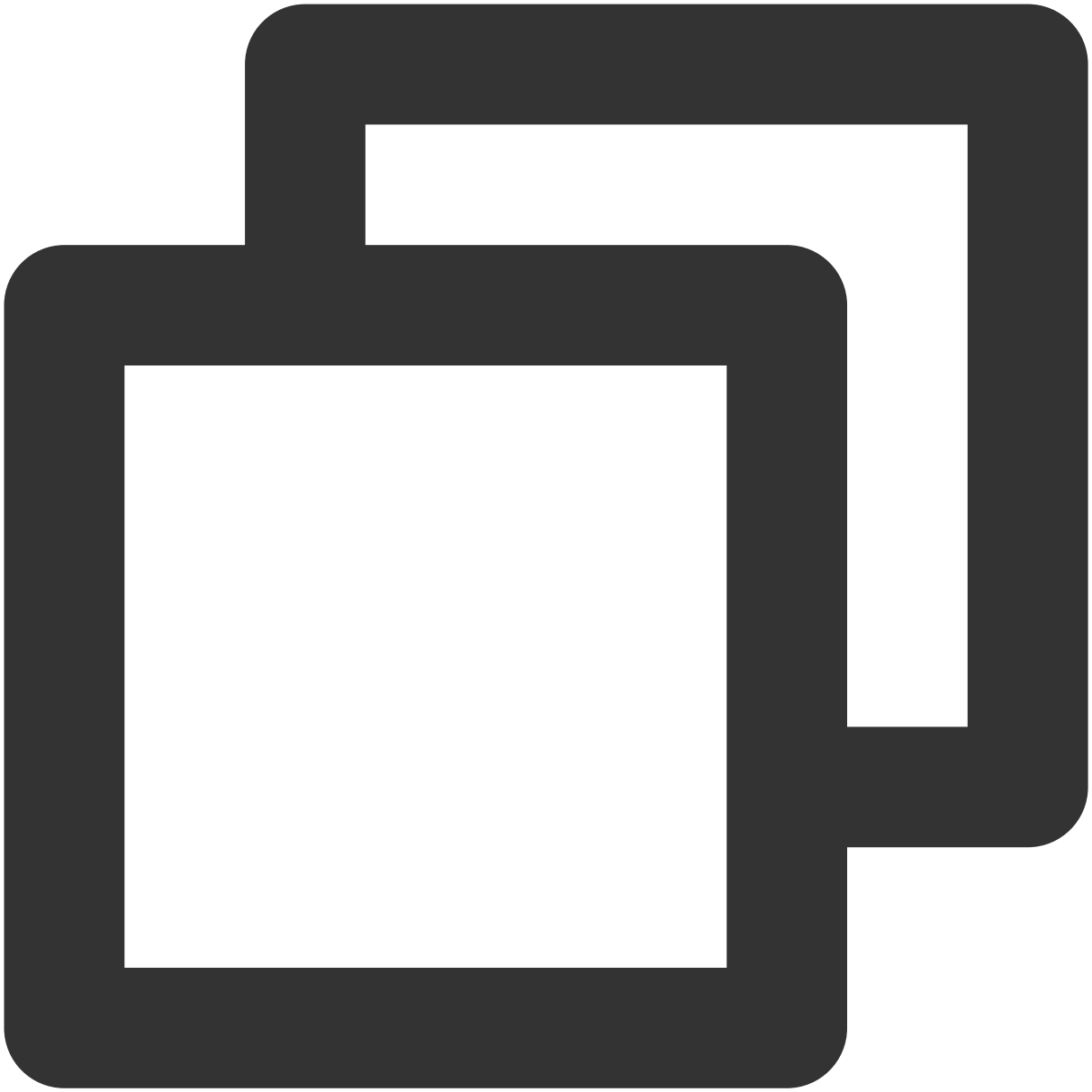
功能说明

用于查询媒体文件在某个时间的截图。

注意：

使用此接口前，请确保已打开官网控制台中数据处理下的媒体处理开关，否则会报错 `media bucket unbinded, bucket's host is unavailable` 。

使用示例



```
var config = {
  // 需要替换成您自己的存储桶信息
  Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
  Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
cos.request({
    Bucket: config.Bucket,
    Region: config.Region,
    Method: 'GET',
    Key: 'test.mp4',  /* 存储桶内的媒体文件，必须字段 */
    Query: {
```

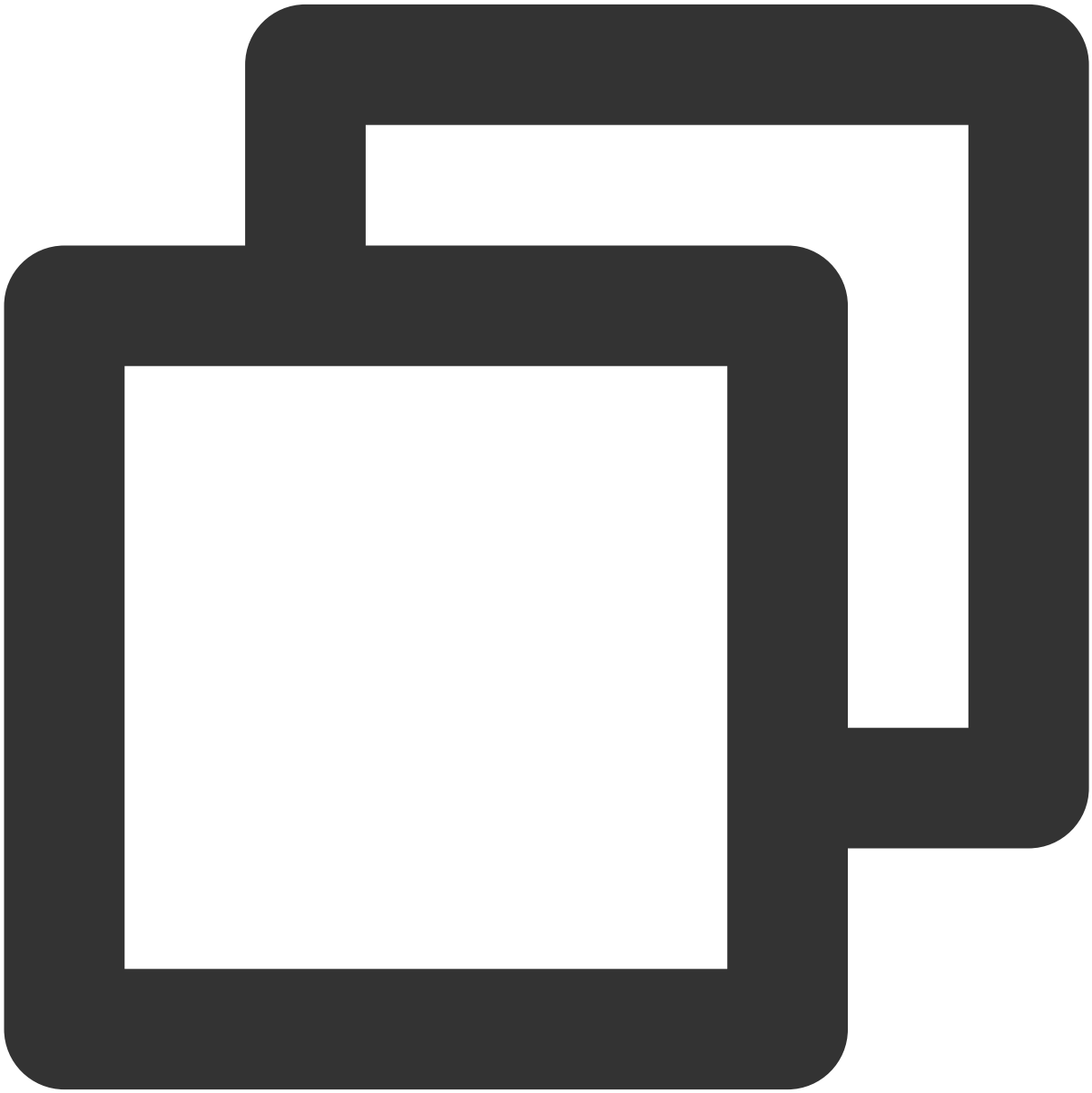
```
            'ci-process': 'snapshot', /** 固定值，必须 */
            time: 1, /** 截图的时间点，单位为秒，必须 */
            // width: 0, /** 截图的宽，非必须 */
            // height: 0, /** 截图的高，非必须 */
            // format: 'jpg', /** 截图的格式，支持 jpg 和 png，默认 jpg，非必须 */
            // rotate: 'auto', /** 图片旋转方式，默认为'auto'，非必须 */
            // mode: 'exactframe', /** 截帧方式，默认为'exactframe'，非必须 */
        },
        RawBody: true,
    },
    function(err, data){
        console.log(err || data);
    });
```

参数说明

参数名称	参数描述	类型	是否必填
Bucket	存储桶的名称，命名格式为 BucketName-APPID，此处填写的存储桶名称必须为此格式	String	是
Region	存储桶所在地域，枚举值请参见 地域和访问域名	String	是
Key	对象键（Object 的名称），对象在存储桶中的唯一标识，详情请参见 对象概述	String	是
ci-process	操作类型，固定使用 snapshot	String	是
time	截图的时间点，单位为秒	Number	是
width	截图的宽。默认为0	Number	否
height	截图的高。默认为0。当 width 和 height 都为0时，表示使用视频的宽高；如果单个为0，则以另外一个值按视频宽高比例自动适应	Number	否
format	截图的格式，支持 jpg 和 png，默认 jpg	String	否
rotate	图片旋转方式 auto：按视频旋转信息进行自动旋转 off：不旋转，默认值为 auto	String	否
mode	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，默认值为 exactframe	String	否

回调函数说明

版权所有：腾讯云计算（北京）有限责任公司



```
function(err, data) { ... }
```

参数名	参数描述	类型
err	请求发生错误时返回的对象，包括网络错误和业务错误。如果请求成功则为空，更多详情请参见 错误码	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object

data	请求成功时返回的对象，如果请求发生错误，则为空	Object
- statusCode	请求返回的 HTTP 状态码，例如200、403、404等	Number
- headers	请求返回的头部信息	Object
- RequestId	请求的唯一 ID	String
- Body	返回的文件内容，默认为 String 形式	String

内容审核
直播审核

最近更新时间：2024-03-13 11:41:11

简介

内容审核功能是由 数据万象（Cloud Infinite，CI）提供的，数据万象将处理能力与 COS SDK 完全结合，您可以直接按照本篇文档指引进行使用。

说明：

使用内容审核服务需拥有数据万象使用权限：

主账号请 单击此处 进行角色授权。

子账号请参见 授权子账号接入数据万象服务 文档。

本文档提供关于直播审核的 API 概览和 SDK 示例代码。

注意：

COS 小程序 SDK 版本需要大于等于 v1.1.1。

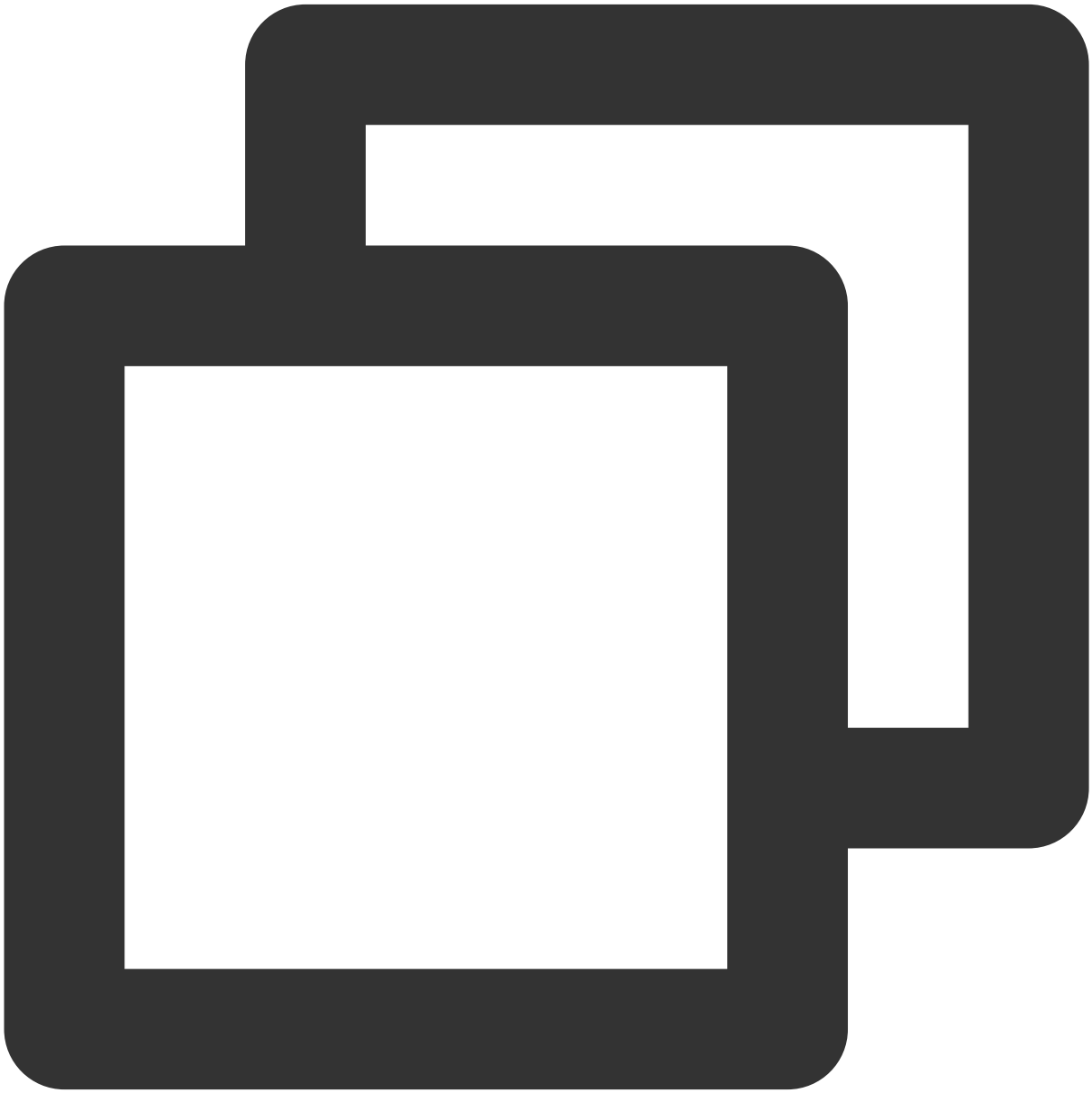
API	操作描述
提交直播审核任务	用于提交一个直播审核任务
查询直播审核任务结果	用于查询指定的直播审核任务

提交直播审核任务

功能说明

用于提交一个直播审核任务。

请求示例



```
function postLiveAuditing() {
  var config = {
    // 需要替换成您自己的存储桶信息
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/video/auditing';
  var body = COS.util.json2xml({
    Request: {
      Type: 'live_video',
```

```
        Input: {
          Url: 'rtmp://example.com/live/123', // 需要审核的直播流播放地址
          // DataId: '',
          // UserInfo: {},
        },
        Conf: {
          BizType: '766d07a7af937c26216c51db29793ea6', // BizType 可通过控制台创建直播审核
          // Callback: 'https://callback.com', // 回调地址，非必须
          // CallbackType: 1, // 回调片段类型，非必须
        }
      }
    });
    cos.request({
        Bucket: config.Bucket,
        Region: config.Region,
        Method: 'POST',
        Url: url,
        Key: '/video/auditing',
        ContentType: 'application/xml',
        Body: body
    },
    function(err, data){
        console.log(err || data);
    });
}
postLiveAuditing();
```

参数说明

节点名称（关键字）	父节点	描述	类型	是否必选
Request	无	直播流审核的具体配置项。	Container	是

Container 类型 Request 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Type	Request	审核的任务类型，直播流审核固定为 live_video。	String	是
Input	Request	包含需要审核的直播流信息。	Container	是
Conf	Request	包含审核规则的配置信息。	Container	是
StorageConf	Request	包含直播流转存的配置信息。	Container	否

版权所有：腾讯云计算（北京）有限责任公司

Container 类型 Input 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Url	Request.Input	需要审核的直播流播放地址，例如 rtmp://example.com/live/123。	String	是
DataId	Request.Input	该字段在审核结果中会返回原始内容，长度限制为512字节。您可以使用该字段对待审核的数据进行唯一业务标识。	String	否
UserInfo	Request.Input	自定义字段，可用于辅助行为数据分析。	Container	否

Container 节点 UserInfo 的内容：

节点名称（关键字）	父节点	描述	类型	是否必选
TokenId	Request.Input.UserInfo	一般用于表示账号信息，长度不超过128字节。	String	否
Nickname	Request.Input.UserInfo	一般用于表示昵称信息，长度不超过128字节。	String	否
DeviceId	Request.Input.UserInfo	一般用于表示设备信息，长度不超过128字节。	String	否
AppId	Request.Input.UserInfo	一般用于表示 App 的唯一标识，长度不超过128字节。	String	否
Room	Request.Input.UserInfo	一般用于表示房间号信息，长度不超过128字节。	String	否
IP	Request.Input.UserInfo	一般用于表示 IP 地址信息，长度不超过128字节。	String	否
Type	Request.Input.UserInfo	一般用于表示业务类型，长度不超过128字节。	String	否
ReceiveTokenId	Request.Input.UserInfo	一般用于表示接收消息的用户账号，长度不超过128字节。	String	否
Gender	Request.Input.UserInfo	一般用于表示性别信息，长度不超过128字	String	否

版权所有：腾讯云计算（北京）有限责任公司

		节。		
Level	Request.Input.UserInfo	一般用于表示等级信息，长度不超过128字节。	String	否
Role	Request.Input.UserInfo	一般用于表示角色信息，长度不超过128字节。	String	否

Container 类型 Conf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
BizType	Request.Conf	表示审核策略的唯一标识，您可以通过控制台上的审核策略页面，配置您希望审核的场景，例如涉黄、广告、违法违规等，配置指引：设置审核策略。 您可以在控制台上获取到 BizType。BizType 填写时，此条审核请求将按照该审核策略中配置的场景进行审核。	String	是
Callback	Request.Conf	回调地址，以 `http://` 或者 `https://` 开头的地址。	String	否
CallbackType	Request.Conf	回调片段类型，有效值：1（回调全部截帧和音频片段）、2（仅回调违规截帧和音频片段）。默认为 1。	Integer	否

Container 类型 StorageConf 的具体数据描述如下：

节点名称（关键字）	父节点	描述	类型	是否必选
Path	Request.StorageConf	表示直播流所要转存的路径，直播流的 ts 文件和 m3u8 文件将保存在本桶该目录下。 m3u8 文件保存文件名为 Path/{$JobId}.m3u8，ts 文件的保存文件名为 Path/{$JobId}-{$Realtime}.ts，其中 Realtime 为17位 `年月日时分秒毫秒` 时间。	String	否

返回结果说明

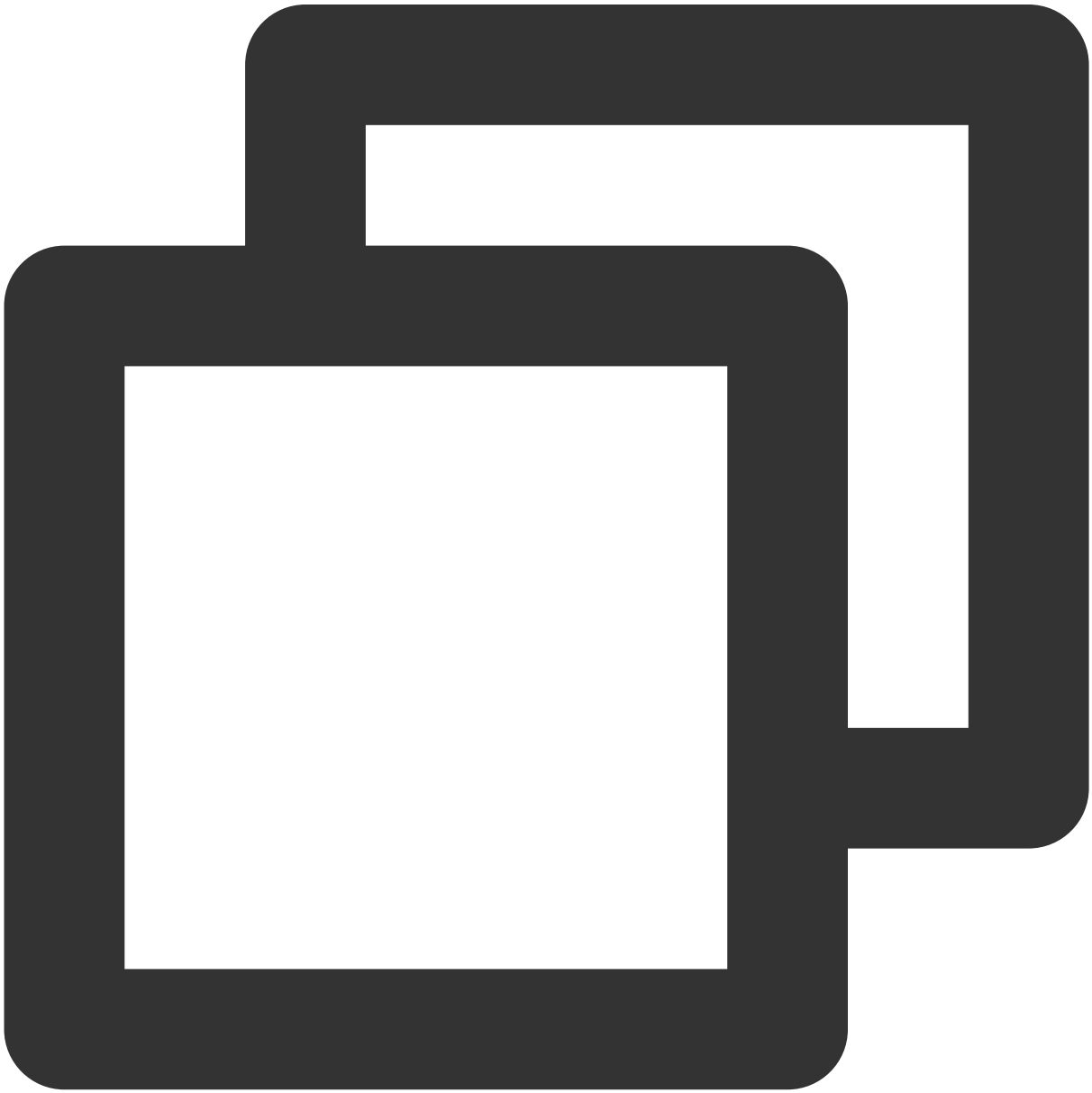
详情请参见 提交直播审核任务。

查询直播审核任务结果

功能说明

用于查询直播审核任务执行情况和结果。

请求示例



```
function getLiveAuditingResult() {
  var config = {
    // 需要替换成您自己的存储桶信息
```

```
    Bucket: 'examplebucket-1250000000', /* 存储桶，必须 */
    Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
  };
  var jobId = 'av0ca69557bd6111ed904c5254009411xx'; // jobId 可以通过提交直播审核任务返回
  var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
  var url = 'https://' + host + '/video/auditing/' + jobId;
  cos.request({
      Bucket: config.Bucket,
      Region: config.Region,
      Method: 'GET',
      Key: '/video/auditing/' + jobId,
      Url: url,
  },
  function(err, data){
      console.log(err || data);
  });
}
getLiveAuditingResult();
```

参数说明

参数名称	描述	类型	是否必选
jobId	需要查询的任务 ID	String	是

返回结果说明

详情请参见 查询直播审核任务结果。

SDK 产品个人信息保护规则
数据万象 SDK 产品个人信息保护规则

最近更新时间：2024-03-13 11:41:55

引言

数据万象 SDK（以下统称“SDK 产品”）由腾讯云计算（北京）有限责任公司（以下简称“我们”）开发，公司注册地为北京市海淀区知春路49号3层西部309。
《数据万象 SDK 产品个人信息保护规则》（以下简称“本规则”）主要向开发者及其终端用户（“终端用户”）告知，为了实现 SDK 产品的相关功能，SDK 产品需收集、使用和处理终端用户个人信息的情况。
请开发者及终端用户认真阅读本规则。如您是开发者，请您确认充分了解并同意本规则后再集成 SDK 产品，如果您不同意本规则及按照本规则履行对应的用户个人信息保护义务，应立即停止接入及使用 SDK 产品；同时，您应仅在征得终端用户的同意后集成 SDK 产品并处理终端用户的个人信息。

特别说明

如您是开发者，您应当：
1. **遵守法律、法规收集、使用和处理终端用户的个人信息，包括但不限于制定和公布有关个人信息保护的隐私政策等。**
2. **在集成 SDK 产品前，告知终端用户 SDK 产品收集、使用和处理终端用户个人信息的情况，并依法征得终端用户同意。**
3. **在征得终端用户的同意前，除非法律法规另有规定，不应收集任何终端用户的个人信息。**
4. **向终端用户提供易于操作且满足法律法规要求的用户权利实现机制，并告知终端用户如何查阅、复制、修改、删除个人信息，撤回同意，以及限制个人信息处理、转移个人信息、获取个人信息副本和注销账号。**
5. **遵守本规则的要求。**

如开发者和终端用户对本规则内容有任何疑问或建议，可随时通过本规则第七条提供的方式与我们联系。

1. 我们收集的信息及我们如何使用信息

1.1 为实现 SDK 产品功能所需收集的个人信息

本 SDK 产品功能的实现不需要收集终端用户的任何个人信息。

1.2 为实现 SDK 产品功能所需的权限

为实现 SDK 产品的相应功能所必须，我们会通过开发者的应用申请所需权限。

操作系统	权限名称	使用目的	是否可选
iOS	网络权限	允许访问网络	由 App 申请，SDK 不涉及

请注意，在不同设备和系统中，权限显示方式及关闭方式会有所不同，需同时参考其使用的设备及操作系统开发方的说明或指引。当终端用户关闭权限即代表其取消了相应的授权，我们和开发者将不会继续收集和使用相关权限所对应的个人信息，也无法为终端用户提供需要终端用户开启权限才能提供的对应的功能。

1.3 根据法律法规的规定，以下是征得用户同意的例外情形:

1. 为订立、履行与终端用户的合同所必需。
2. 为履行我们的法定义务所必需。
3. 为应对突发公共卫生事件，或者紧急情况下为保护终端用户的生命健康和财产安全所必需。
4. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理终端用户的个人信息。
5. 依照本法规定在合理的范围内处理终端用户自行公开或者其他已经合法公开的个人信息。
6. 法律行政法规规定的其他情形。

特别提示：如我们收集的信息无法单独或结合其他信息识别到终端用户的个人身份，其不属于法律意义上的个人信息。

2. 终端用户如何管理自己的信息

我们非常重视终端用户对其个人信息管理的权利，并竭力帮助终端用户管理个人信息，包括个人信息查阅、复制、删除、注销账号以及设置隐私功能等，以保障终端用户的权利。如您是开发者，您应当为终端用户提供实现查阅、复制、修改、删除个人信息、撤回同意和注销账号的方式。

基于终端用户的同意而进行的个人信息处理活动，终端用户有权撤回该同意。我们已向开发者提供关闭本 SDK 产品的能力，请开发者点击（Android、iOS）查看操作指引。由于我们与终端用户无直接的交互对话界面，终端用户可以直接联系开发者停止使用本 SDK 产品，也可通过本规则第七条提供的方式与我们联系。如您是终端用户，请您理解，特定的业务功能或服务需要您提供服务所需的信息才能得以完成，当您撤回同意后，我们无法继续为您提供对应的功能或服务，也不再处理您相应的个人信息。您撤回同意的决定，不会影响我们此前基于您的授权而开展的个人信息处理。

3. 信息的存储

3.1 存储信息的地点

我们遵守法律法规的规定，将在中华人民共和国境内收集和产生的个人信息存储在境内。

3.2 存储信息的期限

一般而言，我们仅在为实现目的所必需的最短时间内保留终端用户的个人信息，但下列情况除外：

为遵守适用的法律法规等有关规定。
为遵守法院判决、裁定或其他法律程序的规定。
为遵守相关政府机关执法的要求。

4. 信息安全

我们为终端用户的个人信息提供相应的安全保障，以防止信息的丢失、不当使用、未经授权访问或披露。
我们严格遵守法律法规保护终端用户的个人信息。
我们将在合理的安全水平内使用各种安全保护措施以保障信息的安全。
例如，我们使用加密技术、匿名化处理等手段来保护终端用户的个人信息。
我们建立严谨的管理制度、流程和组织确保信息安全。
例如，我们严格限制访问信息的人员范围，要求他们遵守保密义务，并进行审查。
若发生个人信息泄露等安全事件，我们会启动应急预案，阻止安全事件扩大，并以推送通知、公告等形式告知开发者。

5. 未成年人保护

本 SDK 产品主要面向成年人。
若您是开发者，如果终端用户是未满14周岁的未成年人（“儿童”），您应当向儿童的父母或其他监护人告知本规则，并在征得儿童儿童的父母或其他监护人同意的前提下处理儿童个人信息。如果我们发现开发者未征得儿童监护人同意向我们提供儿童个人信息的，我们将会采取措施尽快删除。
若您是儿童监护人，当您对您所监护儿童个人信息保护有相关疑问或权利请求时，您可以联系开发者, 或通过本规则第七条提供的方式与我们联系。

6. 变更

我们会适时修订本规则的内容。
如本规则的修订会导致终端用户在本规则项下权利的实质减损，我们将在变更生效前，通过网站公告等方式进行告知。如您是开发者，当更新后的本规则对处理终端用户的个人信息情况有变动的，您应当适时更新隐私政策，并以弹框形式通知终端用户并且征得其同意，**如果终端用户不同意接受本规则, 请停止集成 SDK 产品。**

7. 联系我们

我们设立了专门的个人信息保护团队和个人信息保护负责人，如果开发者和/或终端用户对本规则或个人信息保护相关事宜有任何疑问或投诉、建议时，可以通过以下方式与我们联系：

版权所有：腾讯云计算（北京）有限责任公司

通过 腾讯客服 与我们联系

将问题发送至 `Dataprivacy@tencent.com` 。

邮寄信件至：中国广东省深圳市南山区海天二路33号腾讯滨海大厦 数据隐私保护部(收)邮编：518054。

我们将尽快审核所涉问题, 并在15个工作日或法律法规规定的期限内予以反馈。

版权所有：腾讯云计算（北京）有限责任公司