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】

©2013-2023 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

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。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
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。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

SDK 文档
- SDK 概览
- Android SDK
  - 快速入门
  - 文档处理
    - 查询文档预览开通状态
    - 文档转HTML
    - 文档预览
  - 基础图片处理
  - 加载 TPG 图片
  - 加载 AVIF 图片
  - 使用 Glide 加载图片
- COS Android SDK
  - 快速入门
  - 快速体验
  - 存储桶操作
  - 对象操作
    - 上传对象
    - 下载对象
    - 复制与移动对象
    - 列出对象
    - 删除对象
    - 查询对象元数据
    - 生成预签名 URL
    - 恢复归档对象
    - 预请求跨域配置
    - 服务端加密
    - 单链接限速
    - 检索对象内容
  - 异地容灾
    - 版本控制
    - 跨地域复制
  - 数据管理
    - 生命周期
    - 日志管理
    - 存储桶标签
    - 对象标签
    - 静态网站
    - 清单
  - 图片处理
    - 图片持久化处理
    - 基础图片处理
    - 图片高级压缩
    - 盲水印
  - 媒体处理
    - 媒体 bucket 接口
    - 媒体信息接口
    - 媒体截图接口
  - 文档处理
    - 文档预览
  - 内容审核
    - 图片审核
    - 视频审核
    - 音频审核

版权所有：腾讯云计算（北京）有限责任公司

文本审核
文档审核
网页审核
AI 内容识别
图片二维码
语音识别
任务接口
语音 bucket 接口
队列接口
访问管理
跨域访问
自定义域名
访问控制
数据校验
CRC64 校验
设置自定义头部
设置访问域名（CDN/全球加速）
异常处理
升级到 XML Android SDK
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
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
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
.NET(C#) SDK
快速入门
图片处理
基础图片处理
盲水印
媒体处理

版权所有：腾讯云计算（北京）有限责任公司

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
图片样式
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
文档审核
网页审核
AI 内容识别
图片二维码
图片标签
图片质量评估
人脸识别
人脸核身
汽车识别
图片文字识别（OCR）
iOS SDK
快速入门
基础图片处理
加载 TPG 图片
加载 AVIF 图片
使用 SDWebImage 加载图片
COS iOS SDK
快速入门
快速体验

存储桶操作
对象操作
上传对象
下载对象
列出对象
复制与移动对象
检索对象内容
检查对象是否存在
删除对象
恢复归档对象
查询对象元数据
服务端加密
对象访问 URL
生成预签名链接
预请求跨域配置
异地容灾
版本控制
跨地域复制
数据管理
生命周期
日志管理
存储桶标签
对象标签
静态网站
清单
访问管理
跨域访问
自定义域名
访问控制
防盗链
内容识别
图片标签
图片二维码
语音识别
任务接口
语音 bucket 接口
队列接口
图片处理
盲水印
基础图片处理
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
文档审核
网页审核
设置自定义头部
设置访问域名（CDN/全球加速）
异常处理
升级到 XML iOS SDK
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
  - 图片二维码
  - 图片标签
  - 汽车识别
  - 以图搜图
- 媒体处理
  - 模板操作
  - 任务操作
  - 工作流操作
  - 桶操作
  - 队列操作
  - 视频截帧
  - 视频元信息获取
- 文档预览
  - 任务操作
  - 桶操作
  - 队列操作
  - 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
- 文件处理
  - 文件压缩
  - 文件哈希值计算
  - 文件解压
- 回调信息处理
- 异常处理

JavaScript SDK

- 快速入门
- 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
  - 图片文字识别（OCR）
- 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

版权所有：腾讯云计算（北京）有限责任公司

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
Node.js 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
文档审核
网页审核
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
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
图片文字识别（OCR）
图片质量评分
内容审核
图片审核
视频审核
音频审核
文档审核
文本审核

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
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
视频质量评分任务接口
音视频流分离任务接口
音频降噪任务接口
工作流接口
删除工作流
批量触发存量任务
搜索工作流
获取工作流实例列表
获取工作流实例详情
模板接口
动图模板接口
截图模板接口
水印模板接口
转码模板接口
拼接模板接口
极速高清转码模板接口
精彩集锦模板接口
人声分离模板接口
视频增强模板接口
超分辨率模板接口
图片处理模板接口
搜索模板接口
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
文档处理
文档转码

版权所有：腾讯云计算（北京）有限责任公司

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
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
模板接口
语音识别
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
图片审核
视频审核
音频审核
文本审核
文档审核
网页审核
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：2022-08-25 09:09:47

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
COS 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
COS 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：

- 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。

版权所有：腾讯云计算（北京）有限责任公司

Android SDK
快速入门

最近更新时间：2021-12-16 18:25:36

简介

为了开发者更方便快速地使用数据万象的基础图片处理功能和 CDN 的云闪图片分发功能，我们提供以下5个模块，开发者可根据具体需求进行选择。

模块	功能
cloud-infinite（默认模块）	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。
cloud-infinite-loader	使用 CIImageLoadRequest 实例，请求网络图片并返回图片 data 数据。
TPG	解码 TPG 格式图片并显示，可用于显示普通图片，也可用于 TPG 图。
AVIF	解码 AVIF 格式图片并显示，可用于显示普通图片，也可用于 AVIF 图。
cloud-infinite-glide	依赖于 glide、cloud-infinite 模块，提供了数据万象基础图片处理功能。

说明：

- 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。

相关资源

- SDK 源码和 Demo 请参见 CI Android SDK。
- SDK 接口与参数文档请参见 SDK API。
- 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。

步骤1：安装 cloud-infinite SDK

cloud-infinite 模块主要功能如下：

- 包含数据万象的基础图片处理操作，分别有图片缩放、剪裁、旋转、格式转换、质量变换、高斯模糊、锐化、水印、获取图片主题色等。
- 将每一个操作封装为一个 CITransform，而且支持各个基础操作可以组合使用，并将组合好的操作集合构建出可以直接进行网络请求的 URL。

使用 Gradle 集成

说明：
bintray 仓库已经下线，万象相关 SDK 已经迁移到 mavenCentral，引用路径和之前不同，您在更新的时候请使用新的引用路径。

在项目级别（通常是根目录下）的 build.gradle 中添加：

```
repositories {
google()
// 增加这行
mavenCentral()
}
```

在app或者其他类库的 build.gradle 中添加：

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

步骤2：开始使用

构建 CIImageLoadRequest 实例

1. 在使用 CloudInfinite 构建具有数据万象功能的图片 URL 时，首先需要实例化 CloudInfinite 类。

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

2. 实例化图片转换类 CITransformation，并设置相关操作，这里以 TPG 为例。更多功能请参见 基础图片处理。

```
CITransformation transform = new CITransformation();
transform.format(CIImageFormat.TPG, CIImageLoadOptions.LoadTypeUrlFooter)
```

3. 使用 CloudInfinite 实例构建具有万象功能的图片 URL。
   - 同步方式构建

```
CIImageLoadRequest request = cloudInfinite.requestWithBaseUrlSync(objectUrl, transform);
// 图片url
URL imageURL = request.getUrl();
// header参数
Map<String, List<String>> heaers = request.getHeaders();
```

   - 异步方式构建

```
cloudInfinite.requestWithBaseUrl(objectUrl, transform, new CloudInfiniteCallback() {
@Override
public void onImageLoadRequest(@NonNull final CIImageLoadRequest request) {
// 图片url
URL imageURL = request.getUrl();
// header参数
Map<String, List<String>> heaers = request.getHeaders();
}
});
```

> **说明：**
> 仅在进行图片格式转换并且 options 设置为 LoadTypeAcceptHeader 时，CIImageLoadRequest 的 header 参数为有效值。

步骤3：加载图片

构建成功 CIImageLoadRequest 实例后，使用 cloud-infinite-loader SDK 进行图片加载。

1. 集成 cloud-infinite-loader SDK。

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

2. 加入 INTERNET 权限。

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

3. 加载图片。
   - 直接加载 ImageView

```
CIImageLoader ciImageLoader = new CIImageLoader();
ciImageLoader.display(request, imageview);
```

- 请求得到图片数据

```
CIImageLoader ciImageLoader = new CIImageLoader();
ciImageLoader.loadData(request, new CIImageLoadDataCallBack() {
@Override
public void onLoadData(byte[] bytes) {
//伪代码给 imageview 设置 bytes 数据
//imageview.setBytes(bytes);
}

@Override
public void onFailure(QCloudClientException clientException, QCloudServiceException serviceException) {

}
});
```

文档处理
查询文档预览开通状态

最近更新时间：2022-12-28 17:43:36

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	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

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。

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。

> 注意：
> COS Android SDK 版本需要大于等于 v5.9.8。

示例代码

```
DescribeDocProcessBucketsRequest request = new DescribeDocProcessBucketsRequest();
request.setPageNumber(1);
request.setPageSize(20);
ciService.describeDocProcessBucketsAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// 详细字段请查看api文档或者SDK源码
DescribeDocProcessBucketsResult describeDocProcessBucketsResult = (DescribeDocProcessBucketsResult) result;
}
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

文档转HTML

最近更新时间：2022-12-28 17:43:44

简介

本文档提供关于文档转 HTML 的 API 概览以及 SDK 示例代码。

API	操作描述
文档转HTML	用于文档转HTML的功能

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。

文档转HTML

功能说明

用于文档转HTML的功能，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。

示例代码

以HTML格式预览文档

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
String localPath = "localdownloadpath"; // 保存在本地文件夹的路径
PreviewDocumentInHtmlRequest request = new PreviewDocumentInHtmlRequest(bucket,
cosPath, localPath);
cosXmlService.previewDocumentInHtmlAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PreviewDocumentInHtmlResult previewDocumentInHtmlResult = (PreviewDocumentInHtmlResult) result;
String previewFilePath = previewDocumentInHtmlResult.getPreviewFilePath();
}
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

以HTML格式链接预览文档

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
PreviewDocumentInHtmlLinkRequest request = new PreviewDocumentInHtmlLinkRequest(bucket,
cosPath);
cosXmlService.previewDocumentInHtmlLinkAsync(request, new CosXmlResultListener() {
```

```
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PreviewDocumentInHtmlLinkResult previewDocumentInHtmlLinkResult = (PreviewDocumentInHtmlLinkResult) result;
String previewUrl = previewDocumentInHtmlLinkResult.getPreviewUrl();
}
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

以HTML格式直出内容预览文档到字节数组

（注意：请不要通过本接口预览大文件，否则容易造成内存溢出）

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
try {
byte[] bytes = cosXmlService.previewDocumentInHtmlBytes(bucket, cosPath);
} catch (CosXmlClientException e) {
e.printStackTrace();
} catch (CosXmlServiceException e) {
e.printStackTrace();
}
```

说明：

更多完整示例，请前往 GitHub 查看。

文档预览

最近更新时间：2022-12-28 17:44:03

简介

本文档提供关于文档转码的 API 概览以及 SDK 示例代码。

API	操作描述
文档转码同步请求接口	用于文档转码同步请求的功能

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。

文档转码同步请求接口

功能说明

用于文档转码同步请求的功能，文档预览功能支持对多种文件类型生成图片格式预览。

> 注意：
> 该接口属于同步请求接口，SDK 暂不支持异步任务处理

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
String localPath = "localdownloadpath"; // 保存在本地文件夹的路径
int page = 1; // 需转换的文档页码，从 1 开始
PreviewDocumentRequest previewDocumentRequest = new PreviewDocumentRequest(bucket,
cosPath, localPath, page);

cosXmlService.previewDocumentAsync(previewDocumentRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// 详细字段请查看 api 文档或者 SDK 源码
PreviewDocumentResult previewDocumentResult = (PreviewDocumentResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

基础图片处理

最近更新时间：2021-12-16 18:25:30

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。

```
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
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。
- 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.qcloud.cos:tpg:1.3.2'
```

在使用图片格式转换时，如果需要转为 AVIF 格式，则需要依赖 avif SDK，如下所示。

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

使用 TPG 或 AVIF 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 设置为80后，得到处理结果图的图片质量为64（ 80x80% ）
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();
```

加载 TPG 图片

最近更新时间：2022-08-19 16:01:45

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);
```

版权所有：腾讯云计算（北京）有限责任公司

加载 AVIF 图片

最近更新时间：2022-08-19 16:01:40

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

使用 Glide 加载图片

最近更新时间：2022-08-19 16:01:35

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, Registry registry) {
//注册支持 CIImageLoadRequest 的 loader
registry.prepend(CIImageLoadRequest.class, InputStream.class, new CIImageRequestModelLoader.Factory());

//注册主色解码器
registry.prepend(InputStream.class, Bitmap.class, new ImageAveDecoder(glide.getBitmapPool()));
}
}
```

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.getBitmapPool()));
```

```
//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 CIImageRequestModelLoader.Factory());
```

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, Registry registry) {
/*-----------------解码器 开始-------------------------*/
//注册 TPG 静态图片解码器
registry.prepend(InputStream.class, Bitmap.class, new TpgDecoder(glide.getBitmapPool()));
//注册 TPG 动图解码器
ByteBufferTpgGifDecoder byteBufferTpgGifDecoder = new ByteBufferTpgGifDecoder(context, glide.getBitmapPool(), glide.getArrayPool());
registry.prepend(InputStream.class, GifDrawable.class, new StreamTpgGifDecoder(byteBufferTpgGifDecoder));
registry.prepend(ByteBuffer.class, GifDrawable.class, byteBufferTpgGifDecoder);
/*-----------------解码器 结束-------------------------*/
}
}
```

加载静态 TPG 图片

```
registry.prepend(InputStream.class, Bitmap.class, new TpgDecoder(glide.getBitmapPool()));
```

加载动图类型 TPG 图片

```
ByteBufferTpgGifDecoder byteBufferTpgGifDecoder = new ByteBufferTpgGifDecoder(context, glide.getBitmapPool(), glide.getArrayPool());
registry.prepend(InputStream.class, GifDrawable.class, new StreamTpgGifDecoder(byteBufferTpgGifDecoder));
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, Registry registry) {
/*------------------解码器 开始-------------------------*/
//注册 AVIF 静态图片解码器
registry.prepend(Registry.BUCKET_BITMAP, InputStream.class, Bitmap.class, new StreamAvifDecoder(glide.getBitmapPool(), glide.getArrayPool
()));
registry.prepend(Registry.BUCKET_BITMAP, ByteBuffer.class, Bitmap.class, new ByteBufferAvifDecoder(glide.getBitmapPool()));
//注册 AVIF 动图解码器
registry.prepend(InputStream.class, AvifSequenceDrawable.class, new StreamAvifSequenceDecoder(glide.getBitmapPool(), glide.getArrayPool()
));
registry.prepend(ByteBuffer.class, AvifSequenceDrawable.class, new ByteBufferAvifSequenceDecoder(glide.getBitmapPool()));
/*------------------解码器 结束-------------------------*/
}
}
```

加载静态 AVIF 图片

```
registry.prepend(Registry.BUCKET_BITMAP, InputStream.class, Bitmap.class, new StreamAvifDecoder(glide.getBitmapPool(), glide.getArrayPool
()));
registry.prepend(Registry.BUCKET_BITMAP, ByteBuffer.class, Bitmap.class, new ByteBufferAvifDecoder(glide.getBitmapPool()));
```

加载动图类型 AVIF 图片

```
registry.prepend(InputStream.class, AvifSequenceDrawable.class, new StreamAvifSequenceDecoder(glide.getBitmapPool(), glide.getArrayPool()
));
registry.prepend(ByteBuffer.class, AvifSequenceDrawable.class, new ByteBufferAvifSequenceDecoder(glide.getBitmapPool()));
```

版权所有：腾讯云计算（北京）有限责任公司

COS Android SDK 快速入门

最近更新时间：2023-01-04 20:35:27

相关资源

- SDK 源码下载请参见 XML Android SDK。
- 示例 Demo 请参见 XML Android SDK Demo。
- SDK 接口与参数文档请参见 SDK API 参考。
- 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：Android 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。如果您仍在使用 JSON 版本 SDK，请 升级到 XML Android SDK。

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. 请确保您的 Android 应用目标为 API 级别 15 (Ice Cream Sandwich) 或更高版本。
3. 您需要一个可以获取腾讯云临时密钥的远程地址，关于临时密钥的有关说明请参考 移动应用直传实践。

第一步：安装 SDK

方式一：自动集成（推荐）

> 说明：
> bintray 仓库已经下线，COS SDK 已经迁移到 mavenCentral，引用路径和之前不同，您在更新的时候请使用新的引用路径。

使用 mavenCentral 仓库

在项目级别（通常是根目录下）的 build.gradle 中添加：

```
repositories {
google()
// 增加这行
mavenCentral()
}
```

标准版 SDK

在应用级别（通常是 App 模块下）的 build.gradle 中添加依赖：

```
dependencies {
...
// 增加这行
implementation 'com.qcloud.cos:cos-android:5.9.+'
}
```

精简版 SDK

在应用级别（通常是 App 模块下）的 build.gradle 中添加依赖：

```
dependencies {
...
// 增加这行
implementation 'com.qcloud.cos:cos-android-lite:5.9.+'
}
```

关闭 beacon 上报功能

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。

若是想关闭该功能，请在应用级别（通常是 App 模块下）的 build.gradle 中进行如下操作：

版本为5.8.0以及以上：
修改 cos-android 的依赖为

```
dependencies {
...
// 修改为
implementation 'com.qcloud.cos:cos-android-nobeacon:x.x.x'

//lite 版本修改为
implementation 'com.qcloud.cos:cos-android-lite-nobeacon:x.x.x'
}
```

版本为5.5.8至5.7.9：
添加去掉 beacon 的语句

```
dependencies {
...
implementation ('com.qcloud.cos:cos-android:x.x.x'){
// 增加这行
exclude group: 'com.tencent.qcloud', module: 'beacon-android-release'
}
}
```

方式二：手动集成

1. 下载归档文件

您可以通过 快速下载地址 下载最新的正式包，也可以在 SDK Releases 里面找到我们所有历史版本的正式包。

下载完成并解压后，您可以看到里面包含了数个 jar 或 aar 包。下面是对它们的简单说明，请根据需要选择集成的包。

必选的库：

- cos-android：COS 协议实现
- qcloud-foundation：基础库
- bolts-tasks：第三方 Task 库
- okhttp：第三方 Networking 库
- okio：第三方 IO 库

可选的库：

- quic：QUIC 协议，当您使用到 QUIC 协议来传输数据时需要
- beacon： beacon 移动分析，用于改进 SDK

2. 集成到项目中

把需要的包放到您应用模块下的 libs 文件夹下，并在应用级别（通常是 App 模块下）的 build.gradle 文件中添加如下依赖：

```
dependencies {
...
// 增加这行
implementation fileTree(dir: 'libs', include: ['*.jar', '*.aar'])
}
```

第二步：配置权限

网络权限

SDK 需要网络权限，用于与 COS 服务器进行通信，请在应用模块下的 AndroidManifest.xml 中添加如下权限声明：

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

存储权限

如果您的应用场景中需要从外部存储中读写文件，请在应用模块下的 AndroidManifest.xml 中添加如下权限声明：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

> ⚠ 注意：
> 在 Android 6.0（API level 23）以上，您需要在运行时动态申请存储权限。

第三步：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。

1. 实现获取临时密钥

实现一个 BasicLifecycleCredentialProvider 的子类，实现请求临时密钥并返回结果的过程。

```
public static class MySessionCredentialProvider
extends BasicLifecycleCredentialProvider {

@Override
protected QCloudLifecycleCredentials fetchNewCredentials()
throws QCloudClientException {

// 首先从您的临时密钥服务器获取包含了密钥信息的响应

// 然后解析响应，获取临时密钥信息
String tmpSecretId = "SECRETID"; // 临时密钥 SecretId
String tmpSecretKey = "SECRETKEY"; // 临时密钥 SecretKey
String sessionToken = "SESSIONTOKEN"; // 临时密钥 Token
long expiredTime = 1556183496L;//临时密钥有效截止时间戳，单位是秒
```

```
//建议返回服务器时间作为签名的开始时间，避免由于用户手机本地时间偏差过大导致请求过期
// 返回服务器时间作为签名的起始时间
long startTime = 1556182000L; //临时密钥有效起始时间，单位是秒

// 最后返回临时密钥信息对象
return new SessionQCloudCredentials(tmpSecretId, tmpSecretKey,
sessionToken, startTime, expiredTime);
}
}
```

这里假设类名为 MySessionCredentialProvider 。初始化一个实例，来给 SDK 提供密钥。

```
QCloudCredentialProvider myCredentialProvider = new MySessionCredentialProvider();
```

使用永久密钥进行本地调试

您可以使用腾讯云的永久密钥来进行开发阶段的本地调试。**由于该方式存在泄漏密钥的风险，请务必在上线前替换为临时密钥的方式。**

```
String secretId = "SECRETID"; //永久密钥 secretId
String secretKey = "SECRETKEY"; //永久密钥 secretKey

// keyDuration 为请求中的密钥有效期，单位为秒
QCloudCredentialProvider myCredentialProvider =
new ShortTimeCredentialProvider(secretId, secretKey, 300);
```

使用服务端计算的签名对请求授权

实现一个 QCloudSelfSigner 的子类，实现获取服务端签名并加入请求授权。

```
QCloudSelfSigner myQCloudSelfSigner = new QCloudSelfSigner() {
/**
* 对请求进行签名
*
* @param request 需要签名的请求
* @throws QCloudClientException 客户端异常
*/
@Override
public void sign(QCloudHttpRequest request) throws QCloudClientException {
// 1. 把 request 的请求参数传给服务端计算签名
String auth = "get auth from server";
// 2. 给请求添加签名
request.addHeader(HttpConstants.Header.AUTHORIZATION, auth);
}
});
```

2. 初始化 COS Service

使用您提供密钥的实例 myCredentialProvider 或 服务端签名授权实例 myQCloudSelfSigner ，初始化一个 CosXmlService 的实例。

CosXmlService 提供了访问 COS 的所有接口，建议作为 **程序单例** 使用。

```
// 存储桶所在地域简称，例如广州地区是 ap-guangzhou
String region = "COS_REGION";

// 创建 CosXmlServiceConfig 对象，根据需要修改默认的配置参数
```

```
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.setRegion(region)
.isHttps(true) // 使用 HTTPS 请求, 默认为 HTTP 请求
.builder();

// 初始化 COS Service，获取实例
CosXmlService cosXmlService = new CosXmlService(context,
serviceConfig, myCredentialProvider);

// 通过服务端签名授权初始化 COS Service，获取实例
CosXmlService cosXmlService = new CosXmlService(context,
serviceConfig, myQCloudSelfSigner);
```

> **说明：**
> 关于存储桶不同地域的简称请参考 地域和访问域名。

第四步：访问 COS 服务

上传对象

SDK 支持上传本地文件、二进制数据、Uri 以及输入流。下面以上传本地文件为例：

```
// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString(); //本地文件的绝对路径
//若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
srcPath, uploadId);

//设置初始化分块上传回调(5.9.7版本以及后续版本支持)
cosxmlUploadTask.setInitMultipleUploadListener(new InitMultipleUploadListener() {
@Override
public void onSuccess(InitiateMultipartUpload initiateMultipartUpload) {
//用于下次续传上传的 uploadId
String uploadId = initiateMultipartUpload.uploadId;
}
});
//设置上传进度回调
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
```

```
//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调, 可以查看任务过程
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 生成预签名链接 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

下载对象

```
// 高级下载接口支持断点续传，所以会在下载前先发起 HEAD 请求获取文件信息。
// 如果您使用的是临时密钥或者使用子账号访问，请确保权限列表中包含 HeadObject 的权限。

// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
//本地目录路径
String savePathDir = context.getExternalCacheDir().toString();
//本地保存的文件名，若不填（null），则与 COS 上的文件名一样
String savedFileName = "exampleobject";

Context applicationContext = context.getApplicationContext(); // application
```

```
// context
COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext,
bucket, cosPath, savePathDir, savedFileName);

//设置下载进度回调
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
(COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调，可以查看任务过程
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 高级下载接口支持断点续传，所以会在下载前先发起 HEAD 请求获取文件信息。如果您使用的是临时密钥或者使用子账号访问，请确保权限列表中包含 HeadObject 的权限。

快速体验

最近更新时间：2022-08-25 09:01:08

背景

移动互联网时代，App 作为移动互联网服务的基础设施，往往需要上传和下载大量的数据，数据的安全性和可靠性尤为重要。现在开发者可以将数据存储相关的问题交给腾讯云对象存储（Cloud Object Storage，COS）服务，而只需要关心自己应用的业务逻辑即可，可减少很多工作量，提升开发效率。本文主要介绍如何快速搭建一个基于 COS 的应用传输服务，在腾讯云 COS 上实现应用数据的上传下载，在您的服务器上只需要部署您自己的业务、生成和管理临时密钥。

腾讯云 COS 提供 XML 版本的体验 demo，您可以参照本文档来体验 COS 传输实践 demo。

相关资源

本文涉及的 Demo 都存放在 Github 仓库，用户可前往获取。

准备工作

- Android 系统版本：4.4 及以上。
- 腾讯云 APPID、SecretId、SecretKey（可在云 API 密钥 中获取）。

搭建用户客户端

配置客户端

1. 在 Github 仓库 下载项目文件，然后用 IDE 打开。
2. 在环境变量中配置您的 APPID、SecretId、SecretKey。
3. 运行项目，体验 COS 传输实践 demo。

> ⚠ 注意：
>
> - SecretId、SecretKey 明文不要暴露到不安全环境下。
> - 本项目采用的 ShortTimeCredentialProvider 仅仅是为了演示，正式环境中请不要采用此方式，建议采用 通过临时密钥进行授权 的方式。
> - 环境变量更改后，Android Studio 可能需要重启，相关配置才会更新生效。

运行示例 Demo

查询存储桶列表

启动示例 App 后，将展示用户已创建的所有存储桶，如下图所示。



创建存储桶

点击右上角新建桶，在配置页面输入桶名称并选择存储桶的所属 地域，如下图所示。



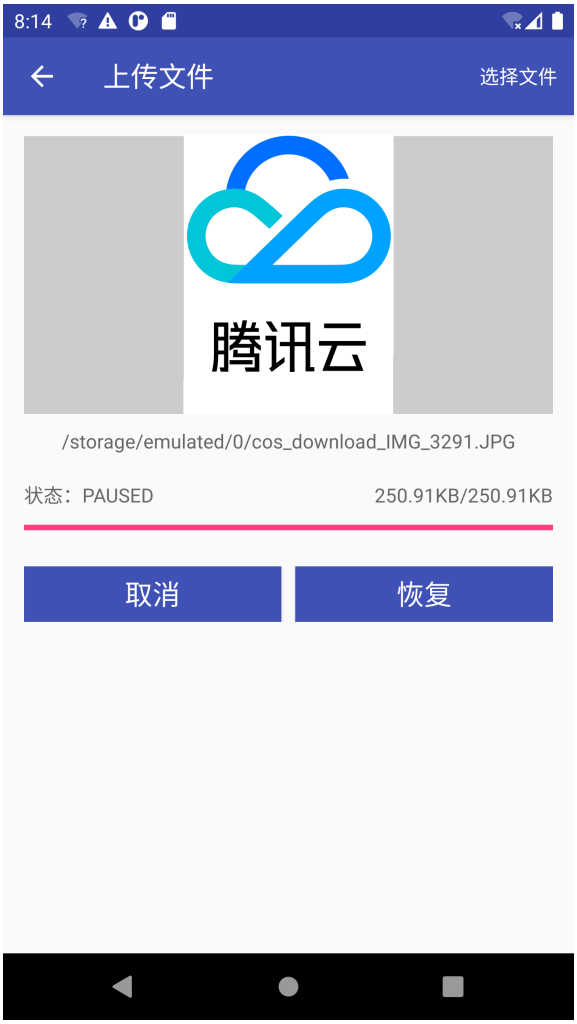
查询对象列表

选择点击某个存储桶，将看到该存储桶内所有文件以及文件夹，如下图所示。



上传文件

在文件列表页面点击右上角的上传，然后选择文件进行上传，如下图所示。



下载文件

在对象列表中，点击文件下方的下载按钮，即可下载文件，如下图所示。


8:15
下载文件
文件名：[redacted].pdf
状态：
8.3MB/33.42MB
取消
恢复

存储桶操作

最近更新时间：2022-08-25 09:01:17

简介

本文档提供关于存储桶基本操作的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Service（List Buckets）	查询存储桶列表	查询指定账号下所有的存储桶列表
PUT Bucket	创建存储桶	在指定账号下创建一个存储桶
HEAD Bucket	检索存储桶及其权限	检索存储桶是否存在且是否有权限访问
DELETE Bucket	删除存储桶	删除指定账号下的空存储桶

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。

查询存储桶列表

功能说明

用于查询指定账号下所有存储桶列表。

示例代码

```
GetServiceRequest getServiceRequest = new GetServiceRequest();
cosXmlService.getServiceAsync(getServiceRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetServiceResult getServiceResult = (GetServiceResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

创建存储桶

功能说明

创建一个存储桶（PUT Bucket）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketRequest putBucketRequest = new PutBucketRequest(bucket);
cosXmlService.putBucketAsync(putBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketResult putBucketResult = (PutBucketResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

检索存储桶及其权限

功能说明

HEAD Bucket 请求可以确认该存储桶是否存在，是否有权限访问。有以下几种情况：

- 存储桶存在且有读取权限，返回 HTTP 状态码为200。
- 无存储桶读取权限，返回 HTTP 状态码为403。
- 存储桶不存在，返回 HTTP 状态码为404。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
HeadBucketRequest headBucketRequest = new HeadBucketRequest(bucket);
cosXmlService.headBucketAsync(headBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
HeadBucketResult headBucketResult = (HeadBucketResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
```

```
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除存储桶

功能说明

删除指定的存储桶（DELETE Bucket）。

> 注意：
> 删除存储桶前，请确保存储桶内的数据和未完成上传的分块数据已全部清空，否则会无法删除存储桶。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketRequest deleteBucketRequest = new DeleteBucketRequest(bucket);
cosXmlService.deleteBucketAsync(deleteBucketRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketResult deleteBucketResult = (DeleteBucketResult) result;
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

对象操作
上传对象

最近更新时间：2022-08-25 09:01:24

简介

本文档提供关于对象的上传、复制操作相关的 API 概览以及 SDK 示例代码。

简单操作

API	操作名	操作描述
PUT Object	简单上传对象	上传一个对象至存储桶

分块操作

API	操作名	操作描述
List Multipart Uploads	查询分块上传	查询正在进行中的分块上传信息
Initiate Multipart Upload	初始化分块上传	初始化分块上传操作
Upload Part	上传分块	分块上传对象
List Parts	查询已上传块	查询特定分块上传操作中的已上传的块
Complete Multipart Upload	完成分块上传	完成整个文件的分块上传
Abort Multipart Upload	终止分块上传	终止一个分块上传操作并删除已上传的块

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。

高级接口（推荐）

上传对象

高级接口封装了简单上传、分块上传接口，根据文件大小智能的选择上传方式，同时支持续传功能。

示例代码一: 上传本地文件

```
// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString(); //本地文件的绝对路径
//若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
srcPath, uploadId);
```

```
//设置上传进度回调
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调, 可以查看任务过程
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码二: 上传二进制数据

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键

// 上传字节数组
byte[] bytes = "this is a test".getBytes(Charset.forName("UTF-8"));
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
```

```
bytes);

//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码三: 流式上传

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键

// 流式上传
InputStream inputStream =
new ByteArrayInputStream("this is a test".getBytes(Charset.forName(
"UTF-8")));
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
inputStream);

//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}
```

```
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

- **更多完整示例，请前往 GitHub 查看。**
- **上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 生成预签名链接 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。**

示例代码四: 通过 URI 上传

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键

// 文件的 Uri
Uri uri = Uri.parse("exampleObject");
// 若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
uri, uploadId);

//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
```

```
}
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码五: 设置智能分块阈值

```
// 默认对大于或等于2M的文件自动进行分块上传，可以通过如下代码修改分块阈值
TransferConfig transferConfig = new TransferConfig.Builder()
.setDivisionForUpload(2 * 1024 * 1024) // 设置大于等于 2M 的文件进行分块上传
.build();

TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码六: 上传暂停、继续与取消

对于上传任务，可以通过以下方式暂停：

```
// 如果上传过程中，已经发起了最后的 Complete Multipart Upload 请求，那么暂停会失败
boolean pauseSuccess = cosxmlUploadTask.pauseSafely();
```

暂停之后，可以通过以下方式续传：

```
// 如果暂停成功，可以恢复上传
if (pauseSuccess) {
cosxmlUploadTask.resume();
}
```

也通过以下方式取消上传：

```
cosxmlUploadTask.cancel();
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码七: 批量上传

```
//本地文件的绝对路径
File[] files = new File(context.getCacheDir(), "exampleDirectory").listFiles();

// 开始批量上传
for (File file : files) {
```

```
//若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
file.getAbsolutePath(), uploadId);

//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
}
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码八：创建目录

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
// 文件夹在存储桶中的位置标识符，即称对象键，必须以 '/' 结尾
String cosPath = "exampleobject/";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, new byte[0]);
cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutObjectResult putObjectResult =
(PutObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
```

```
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
>
> - 更多完整示例，请前往 GitHub 查看。
> - 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码九：设置低优先级任务

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
// 文件夹在存储桶中的位置标识符，即称对象键，必须以 '/' 结尾
String cosPath = "exampleobject/";
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString(); //本地文件的绝对路径
PutObjectRequest putObjectRequest= new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPriorityLow(); // 设置为低优先级上传任务
final COSXMLUploadTask uploadTask = transferManager.upload(putObjectRequest, null);
uploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {}


@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

简单操作

简单上传对象

功能说明

PUT Object 接口可以上传一个对象至指定存储桶中，该操作需要请求者对存储桶有 WRITE 权限。最大支持上传不超过5GB的对象，5GB以上对象请使用 分块上传 或 高级接口 上传。

> 注意：
>
> - Key（文件名）不能以/结尾，否则会被识别为文件夹。
> - 每个主账号（即同一个 APPID），存储桶的 ACL 规则数量最多为1000条，对象 ACL 规则数量不限制。如果您不需要进行对象 ACL 控制，请在上传时不要设置，默认继承存储桶权限。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
```

```
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键。
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString();//"本地文件的绝对路径";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath,
srcPath);

putObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
// todo Do something to update progress...
}
});
cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
PutObjectResult putObjectResult = (PutObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

表单上传对象

功能说明

使用表单请求上传对象。

示例代码

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键。
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString();//"本地文件的绝对路径";

PostObjectRequest postObjectRequest = new PostObjectRequest(bucket, cosPath,
srcPath);

postObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
```

```
public void onProgress(long progress, long max) {
// todo Do something to update progress...
}
});
cosXmlService.postObjectAsync(postObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
PutObjectResult putObjectResult = (PutObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

分块操作

这里说明下分块上传的流程。

分块上传的流程

1. 初始化分块上传（Initiate Multipart Upload），得到 UploadId
2. 使用 UploadId 上传分块（Upload Part），或者复制分块（Upload Part Copy）
3. 完成分块上传（Complete Multipart Upload）

分块继续上传的流程

1. 如果没有记录 UploadId，查询分块上传任务（List Multipart Uploads），得到对应文件的 UploadId
2. 使用 UploadId 列出已上传的分块（List Parts）
3. 使用 UploadId 上传剩余的分块（Upload Part），或者复制剩余的分块（Upload Part Copy）
4. 完成分块上传（Complete Multipart Upload）

终止分块上传的流程

1. 如果没有记录 UploadId，查询分块上传任务（List Multipart Uploads），得到对应文件的 UploadId
2. 终止分块上传并删除已上传分块（Abort Multipart Upload）

查询分块上传

功能说明

查询指定存储桶中正在进行的分块上传（List Multipart Uploads）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
```

```
ListMultiUploadsRequest listMultiUploadsRequest =
new ListMultiUploadsRequest(bucket);
cosXmlService.listMultiUploadsAsync(listMultiUploadsRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
ListMultiUploadsResult listMultiUploadsResult =
(ListMultiUploadsResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

初始化分块上传

功能说明

初始化 Multipart Upload 上传操作，获取对应的 uploadId（Initiate Multipart Upload）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。

InitMultipartUploadRequest initMultipartUploadRequest =
new InitMultipartUploadRequest(bucket, cosPath);
cosXmlService.initMultipartUploadAsync(initMultipartUploadRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
// 分片上传的 uploadId
uploadId = ((InitMultipartUploadResult) result)
.initMultipartUpload.uploadId;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
```

```
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

上传分块

分块上传对象（Upload Part）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
UploadPartRequest uploadPartRequest = new UploadPartRequest(bucket, cosPath,
partNumber, srcFile.getPath(), offset, PART_SIZE, uploadId);

uploadPartRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
// todo Do something to update progress...
}
});

cosXmlService.uploadPartAsync(uploadPartRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
String eTag = ((UploadPartResult) result).eTag;
eTags.put(partNumber, eTag);
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询已上传的分块

功能说明

查询特定分块上传操作中的已上传的块（List Parts）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。

ListPartsRequest listPartsRequest = new ListPartsRequest(bucket, cosPath,
uploadId);
cosXmlService.listPartsAsync(listPartsRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
ListParts listParts = ((ListPartsResult) result).listParts;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

完成分块上传

功能说明

完成整个文件的分块上传（Complete Multipart Upload）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。

CompleteMultiUploadRequest completeMultiUploadRequest =
new CompleteMultiUploadRequest(bucket,
cosPath, uploadId, eTags);
cosXmlService.completeMultiUploadAsync(completeMultiUploadRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
CompleteMultiUploadResult completeMultiUploadResult =
```

```
(CompleteMultiUploadResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

终止分块上传

功能说明

终止一个分块上传操作并删除已上传的块（Abort Multipart Upload）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。

AbortMultiUploadRequest abortMultiUploadRequest =
new AbortMultiUploadRequest(bucket,
cosPath, uploadId);
cosXmlService.abortMultiUploadAsync(abortMultiUploadRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
AbortMultiUploadResult abortMultiUploadResult =
(AbortMultiUploadResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

下载对象

最近更新时间：2022-08-25 09:01:28

简介

本文档提供关于对象的下载操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Object	下载对象	下载一个对象至本地

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。

高级接口（推荐）

下载对象

高级接口支持暂停、恢复以及取消下载请求，同时支持断点下载功能。

示例代码一：下载对象

```
// 高级下载接口支持断点续传，所以会在下载前先发起 HEAD 请求获取文件信息。
// 如果您使用的是临时密钥或者使用子账号访问，请确保权限列表中包含 HeadObject 的权限。

// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
//本地目录路径
String savePathDir = context.getExternalCacheDir().toString();
//本地保存的文件名，若不填（null），则与 COS 上的文件名一样
String savedFileName = "exampleobject";

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext,
bucket, cosPath, savePathDir, savedFileName);

//设置下载进度回调
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
(COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
}
```

```
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调，可以查看任务过程
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

示例代码二：下载暂停、继续与取消

对于下载任务，可以通过以下方式暂停：

```
cosxmlDownloadTask.pause();
```

暂停之后，可以通过以下方式续传：

```
cosxmlDownloadTask.resume();
```

也通过以下方式取消下载：

```
cosxmlDownloadTask.cancel();
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

示例代码三：设置下载支持断点续传

```
// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
// TransferManager 支持断点下载，您只需要保证 bucket、cosPath、savePathDir、savedFileName
// 参数一致，SDK 便会从上次已经下载的位置继续下载。
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
```

```
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
//本地目录路径
String savePathDir = context.getExternalCacheDir().toString();
//本地保存的文件名，若不填（null），则与 COS 上的文件名一样
String savedFileName = "exampleobject";

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathDir, savedFileName);

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext, getObjectRequest);
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码四: 批量下载

```
// 对象在存储桶中的位置标识符，即称对象键
String[] cosPaths = new String[] {
"exampleobject1",
"exampleobject2",
"exampleobject3",
};

for (String cosPath : cosPaths) {

COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext,
bucket, cosPath, savePathDir, savedFileName);
// 设置返回结果回调
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLDownloadTask.COSXMLDownloadTaskResult downloadResult =
(COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码五: 下载目录

```
boolean isTruncated = true;
String marker = null;
try {
while (isTruncated) {
GetBucketRequest getBucketRequest = new GetBucketRequest(region, bucket, directoryPath);
// 设置分页信息
getBucketRequest.setMarker(marker);
// 设置不查询子目录
getBucketRequest.setDelimiter("/");
GetBucketResult getBucketResult = cosXmlService.getBucket(getBucketRequest);
// 批量下载
for (ListBucket.Contents content : getBucketResult.listBucket.contentsList) {
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, content.key, savePathDir);
transferManager.download(context,getObjectRequest);
}
isTruncated = getBucketResult.listBucket.isTruncated;
marker = getBucketResult.listBucket.nextMarker;
}
} catch (CosXmlServiceException serviceException) {
serviceException.printStackTrace();
} catch (CosXmlClientException clientException) {
clientException.printStackTrace();
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

简单操作

下载对象

功能说明

下载一个 Object（文件/对象）至本地（GET Object）。

示例代码

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String savePath = context.getExternalCacheDir().toString(); //本地路径

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
savePath);
getObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
// todo Do something to update progress...
}
});
```

```
cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest,
CosXmlResult cosXmlResult) {
GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

复制与移动对象

最近更新时间：2022-08-25 09:01:33

简介

本文档提供关于复制、移动对象的 API 概览以及 SDK 示例代码。

简单操作

API	操作名	操作描述
PUT Object - Copy	设置对象复制（修改对象属性）	复制文件到目标路径
DELETE Object	删除单个对象	在存储桶中删除指定对象

分块操作

API	操作名	操作描述
List Multipart Uploads	查询分块上传/复制	查询正在进行中的分块上传/复制信息
Initiate Multipart Upload	初始化分块上传/复制	初始化分块上传/复制操作
Upload Part - Copy	复制分块	将其他对象复制为一个分块
List Parts	查询已上传/复制块	查询特定分块操作中的已上传/复制的块
Complete Multipart Upload	完成分块上传/复制	完成整个文件的分块上传/复制
Abort Multipart Upload	终止分块上传/复制	终止一个分块操作并删除已上传/复制的块

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。

复制对象

高级接口封装了简单复制、分块复制接口的异步请求，并支持暂停、恢复以及取消复制请求。

示例代码

```
// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

String sourceAppid = "1250000000"; //账号 APPID
String sourceBucket = "sourcebucket-1250000000"; //源对象所在的存储桶
String sourceRegion = "COS_REGION"; //源对象的存储桶所在的地域
String sourceCosPath = "sourceObject"; //源对象的对象键
//构造源对象属性
CopyObjectRequest.CopySourceStruct copySourceStruct =
new CopyObjectRequest.CopySourceStruct(
sourceAppid, sourceBucket, sourceRegion, sourceCosPath);
//目标桶
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
//目标对象
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
```

```
//复制对象
COSXMLCopyTask cosxmlCopyTask = transferManager.copy(bucket, cosPath,
copySourceStruct);

//设置返回结果回调
cosxmlCopyTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLCopyTask.COSXMLCopyTaskResult copyResult =
(COSXMLCopyTask.COSXMLCopyTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调, 可以查看任务过程
cosxmlCopyTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

移动对象

移动对象主要包括两个操作：复制源对象到目标位置，删除源对象。

由于 COS 通过存储桶名称（Bucket）和对象键（ObjectKey）来标识对象。移动对象也就意味着修改这个对象的标识，COS Java SDK 目前没有提供修改对象唯一标识名的单独接口，但是可以通过组合**复制对象**加上**删除对象**的基本操作，来达到修改对象标识的目的，从而实现移动对象。

例如将 mybucket-1250000000 这个存储桶中的 picture.jpg 这个对象移动到同个存储桶的 doc 路径下。首先可以复制 picture.jpg 对象到存储桶的 doc 路径下，即对象键设定为 doc/picture.jpg，复制完成后删除 picture.jpg 这个对象，来实现“移动”的效果。

同样的，如果想将 mybucket-1250000000 这个存储桶里的 picture.jpg 这个对象移动到 myanothorbucket-1250000000 这个存储桶里，可以先将对象复制到 myanothorbucket-1250000000 存储桶，然后删除原来的对象。

示例代码

```
final String sourceAppid = "1250000000"; //账号 appid
final String sourceBucket = "sourcebucket-1250000000"; //"源对象所在的存储桶
final String sourceRegion = "COS_REGION"; //源对象的存储桶所在的地域
final String sourceKey = "sourceObject"; //源对象键
//构造源对象属性
CopyObjectRequest.CopySourceStruct copySource = new CopyObjectRequest.CopySourceStruct(sourceAppid, sourceBucket,
sourceRegion, sourceKey);

String bucket = "examplebucket-1250000000"; //目标存储桶，格式：BucketName-APPID
String key = "exampleobject"; //目标对象的对象键
```

```
// copy(String bucket, String cosPath, CopyObjectRequest.CopySourceStruct copySourceStruct){
COSXMLCopyTask copyTask = transferManager.copy(bucket, key, copySource);
copyTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
try {
// 复制成功后删除文件
DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(sourceBucket, sourceKey);
DeleteObjectResult deleteResult = cosXmlService.deleteObject(deleteObjectRequest);
} catch (CosXmlClientException e) {
e.printStackTrace();
} catch (CosXmlServiceException e) {
e.printStackTrace();
}
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException exception, CosXmlServiceException serviceException) {

}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

复制对象（修改属性）

复制文件到目标路径（PUT Object-Copy）。

示例代码一: 复制对象时保留对象属性

```
String sourceAppid = "1250000000"; //账号 APPID
String sourceBucket = "sourcebucket-1250000000"; //源对象所在的存储桶
String sourceRegion = "COS_REGION"; //源对象的存储桶所在的地域
String sourceCosPath = "sourceObject"; //源对象键
// 构造源对象属性
CopyObjectRequest.CopySourceStruct copySourceStruct =
new CopyObjectRequest.CopySourceStruct(
sourceAppid, sourceBucket, sourceRegion, sourceCosPath);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
copySourceStruct);

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
CopyObjectResult copyObjectResult = (CopyObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即:
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
```

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：复制对象时替换对象属性

```
String sourceAppid = "1250000000"; //账号 APPID
String sourceBucket = "sourcebucket-1250000000"; //源对象所在的存储桶
String sourceRegion = "COS_REGION"; //源对象的存储桶所在的地域
String sourceCosPath = "sourceObject"; //源对象键
// 构造源对象属性
CopyObjectRequest.CopySourceStruct copySourceStruct =
new CopyObjectRequest.CopySourceStruct(
sourceAppid, sourceBucket, sourceRegion, sourceCosPath);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
copySourceStruct);
copyObjectRequest.setCopyMetaDataDirective(MetaDataDirective.REPLACED);
copyObjectRequest.setXCOSMeta("x-cos-metadata-oldKey", "newValue");

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
CopyObjectResult copyObjectResult = (CopyObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码三: 修改对象元数据

```
String appId = "1250000000"; //账号 APPID
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String region = "COS_REGION"; //源对象的存储桶所在的地域
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
// 构造源对象属性
CopyObjectRequest.CopySourceStruct copySourceStruct =
new CopyObjectRequest.CopySourceStruct(
appId, bucket, region, cosPath);

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
copySourceStruct);
copyObjectRequest.setCopyMetaDataDirective(MetaDataDirective.REPLACED);
// 修改元数据为新值
copyObjectRequest.setXCOSMeta("x-cos-metadata-oldKey", "newValue");

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
CopyObjectResult copyObjectResult = (CopyObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码四: 修改对象存储类型

```
String appId = "1250000000"; //账号 APPID
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String region = "COS_REGION"; //源对象的存储桶所在的地域
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键
// 构造源对象属性
CopyObjectRequest.CopySourceStruct copySourceStruct =
```

```
new CopyObjectRequest.CopySourceStruct(
appId, bucket, region, cosPath);

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
copySourceStruct);
// 修改为低频存储
copyObjectRequest.setCosStorageClass(COSStorageClass.STANDARD_IA);

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
CopyObjectResult copyObjectResult = (CopyObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

列出对象

最近更新时间：2022-08-25 09:01:37

简介

本文档提供关于列出对象操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Bucket（List Objects）	查询对象列表	查询存储桶下的部分或者全部对象
GET Bucket Object Versions	查询对象及其历史版本列表	查询存储桶下的部分或者全部对象及其历史版本信息

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。

查询对象列表

功能说明

查询存储桶下的部分或者全部对象。

示例代码一：获取第一页数据

```
String bucketName = "examplebucket-1250000000"; //格式：BucketName-APPID;
final GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

// 前缀匹配，用来规定返回的对象前缀地址
getBucketRequest.setPrefix("dir/");

// 单次返回最大的条目数量，默认1000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketResult getBucketResult = (GetBucketResult) result;
if (getBucketResult.listBucket.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
prevPageResult = getBucketResult;
}
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：请求下一页数据

```
String bucketName = "examplebucket-1250000000"; //格式：BucketName-APPID;


GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

// 前缀匹配，用来规定返回的对象前缀地址
getBucketRequest.setPrefix("dir/");

// prevPageResult 是上一页的返回结果，这里的 nextMarker 表示下一页的起始位置
String nextMarker = prevPageResult.listBucket.nextMarker;
getBucketRequest.setMarker(nextMarker);

// 单次返回最大的条目数量，默认1000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketResult getBucketResult = (GetBucketResult) result;
if (getBucketResult.listBucket.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
}
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码三：获取对象列表与子目录

```
String bucketName = "examplebucket-1250000000"; //格式：BucketName-APPID;
GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

// 前缀匹配，用来规定返回的对象前缀地址
getBucketRequest.setPrefix("dir/");
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 单次返回最大的条目数量，默认1000
getBucketRequest.setMaxKeys(100);

// 定界符为一个符号，如果有 Prefix，
// 则将 Prefix 到 delimiter 之间的相同路径归为一类，定义为 Common Prefix，
// 然后列出所有 Common Prefix。如果没有 Prefix，则从路径起点开始
getBucketRequest.setDelimiter("/");

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketResult getBucketResult = (GetBucketResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

查询对象历史版本列表

功能说明

查询开启版本控制的存储桶下的部分或者全部对象。

示例代码一：获取对象历史版本列表第一页数据

```
String bucketName = "examplebucket-1250000000"; //格式：BucketName-APPID;
final GetBucketObjectVersionsRequest getBucketRequest =
new GetBucketObjectVersionsRequest(bucketName);

// 前缀匹配，用来规定返回的对象前缀地址
getBucketRequest.setPrefix("dir/");

// 单次返回最大的条目数量，默认1000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketObjectVersionsAsync(getBucketRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketObjectVersionsResult getBucketResult =
```

```
(GetBucketObjectVersionsResult) result;
if (getBucketResult.listVersionResult.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
prevPageResult = getBucketResult;
}
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码二：获取对象历史版本列表下一页数据

```
String bucketName = "examplebucket-1250000000"; //格式：BucketName-APPID;
final GetBucketObjectVersionsRequest getBucketRequest =
new GetBucketObjectVersionsRequest(bucketName);

// 前缀匹配，用来规定返回的对象前缀地址
getBucketRequest.setPrefix("dir/");

// 单次返回最大的条目数量，默认1000
getBucketRequest.setMaxKeys(100);

// prevPageResult 是上一页的返回结果，这里的 nextMarker 与 nextVersionIdMarker
// 表示下一页的起始位置
getBucketRequest.setKeyMarker(prevPageResult.listVersionResult
.nextKeyMarker);
getBucketRequest.setVersionIdMarker(prevPageResult.listVersionResult
.nextVersionIdMarker);

cosXmlService.getBucketObjectVersionsAsync(getBucketRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketObjectVersionsResult getBucketResult =
(GetBucketObjectVersionsResult) result;
if (getBucketResult.listVersionResult.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
prevPageResult = getBucketResult;
}
}
```

```
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

删除对象

最近更新时间：2022-08-25 09:01:42

简介

本文档提供关于对象的删除操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DELETE Object	删除单个对象	在存储桶中删除指定对象
DELETE Multiple Objects	删除多个对象	在存储桶中批量删除对象

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。

删除单个对象

功能说明

删除指定的对象（DELETE Object）。

示例代码

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键

DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(bucket,
cosPath);
cosXmlService.deleteObjectAsync(deleteObjectRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
DeleteObjectResult deleteObjectResult = (DeleteObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除多个对象

功能说明

批量删除多个对象（Delete Multi Objects）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
List<String> objectList = new ArrayList<String>();
objectList.add("exampleobject1"); //对象在存储桶中的位置标识符，即对象键
objectList.add("exampleobject2"); //对象在存储桶中的位置标识符，即对象键

DeleteMultiObjectRequest deleteMultiObjectRequest =
new DeleteMultiObjectRequest(bucket, objectList);
// Quiet 模式只返回报错的 Object 信息。否则返回每个 Object 的删除结果。
deleteMultiObjectRequest.setQuiet(true);
cosXmlService.deleteMultiObjectAsync(deleteMultiObjectRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
DeleteMultiObjectResult deleteMultiObjectResult =
(DeleteMultiObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

删除目录

功能说明

COS 上的文件夹概念是以 '/' 分隔对象名，形成类似文件系统的路径，从而模拟出来的。所以删除文件夹的操作，在 COS 上相当于删除一批有着同样前缀的对象。例如：文件夹 'prefix/' ，代表的是以 'prefix/' 为前缀的所有对象，所以删除 'prefix/' 意味着删除以 'prefix/' 为前缀的所有对象。

目前 COS Android SDK 没有提供一个接口去支持这样的操作，但是可以通过基本操作的组合，达到同样的效果。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String prefix = "folder1/"; //指定前缀

GetBucketRequest getBucketRequest = new GetBucketRequest(bucket);
```

版权所有：腾讯云计算（北京）有限责任公司

```
getBucketRequest.setPrefix(prefix);

// prefix表示要删除的文件夹
getBucketRequest.setPrefix(prefix);
// 设置最大遍历出多少个对象, 一次listobject最大支持1000
getBucketRequest.setMaxKeys(1000);
GetBucketResult getBucketResult = null;

do {
try {
getBucketResult = cosXmlService.getBucket(getBucketRequest);
List<ListBucket.Contents> contents = getBucketResult.listBucket.contentsList;
DeleteMultiObjectRequest deleteMultiObjectRequest = new DeleteMultiObjectRequest(bucket);
for (ListBucket.Contents content : contents) {
deleteMultiObjectRequest.setObjectList(content.key);
}
cosXmlService.deleteMultiObject(deleteMultiObjectRequest);
getBucketRequest.setMarker(getBucketResult.listBucket.nextMarker);
} catch (CosXmlClientException e) {
e.printStackTrace();
return;
} catch (CosXmlServiceException e) {
e.printStackTrace();
return;
}
} while (getBucketResult.listBucket.isTruncated);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询对象元数据

最近更新时间：2022-08-25 09:01:47

简介

本文档提供关于查询对象元数据操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
HEAD Object	查询对象元数据	查询对象的元数据信息

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。

查询对象元数据

功能说明

查询 Object 的 Meta 信息。

示例代码

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
HeadObjectRequest headObjectRequest = new HeadObjectRequest(bucket, cosPath);
cosXmlService.headObjectAsync(headObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
HeadObjectResult headObjectResult = (HeadObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

生成预签名 URL

最近更新时间：2022-08-25 09:01:51

简介

本文档提供关于生成对象预签名链接的示例代码。

> 说明：
>
> - 建议用户使用临时密钥生成预签名，通过临时授权的方式进一步提高预签名上传、下载等请求的安全性。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来生成预签名，建议永久密钥的权限范围仅限于上传或下载操作，以规避风险。

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。

生成对象预签名链接

示例代码一：生成预签名上传链接

```
try {
String bucket = "examplebucket-1250000000"; //存储桶名称
String cosPath = "exampleobject"; //即对象在存储桶中的位置标识符。
String method = "PUT"; //请求 HTTP 方法
PresignedUrlRequest presignedUrlRequest = new PresignedUrlRequest(bucket
, cosPath) {
@Override
public RequestBodySerializer getRequestBody()
throws CosXmlClientException {
//用于计算 put 等需要带上 body 的请求的签名 URL
return RequestBodySerializer.string("text/plain",
"this is test");
}
};
presignedUrlRequest.setRequestMethod(method);
// 设置签名有效期为 60s，注意这里是签名有效期，您需要自行保证密钥有效期
presignedUrlRequest.setSignKeyTime(60);
// 设置不签名 Host
presignedUrlRequest.addNoSignHeader("Host");
String urlWithSign = cosXmlService.getPresignedURL(presignedUrlRequest);
} catch (CosXmlClientException e) {
e.printStackTrace();
}
```

> 说明：
>
> 更多完整示例，请前往 GitHub 查看。

示例代码二：生成预签名下载链接

```
try {
String bucket = "examplebucket-1250000000"; //存储桶名称
String cosPath = "exampleobject"; //即对象在存储桶中的位置标识符。
String method = "GET"; //请求 HTTP 方法.
```

```
PresignedUrlRequest presignedUrlRequest = new PresignedUrlRequest(bucket
, cosPath);
presignedUrlRequest.setRequestMethod(method);

// 设置签名有效期为 60s，注意这里是签名有效期，您需要自行保证密钥有效期
presignedUrlRequest.setSignKeyTime(60);
// 设置不签名 Host
presignedUrlRequest.addNoSignHeader("Host");

String urlWithSign = cosXmlService.getPresignedURL(presignedUrlRequest);

} catch (CosXmlClientException e) {
e.printStackTrace();
}
```

说明：
更多完整示例，请前往 GitHub 查看。

恢复归档对象

最近更新时间：2022-08-25 09:01:56

简介

本文档提供关于恢复归档对象操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
POST Object restore	恢复归档对象	将归档类型的对象取回访问

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。

恢复归档对象

功能说明

将归档类型的对象取回访问（POST Object restore）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。
RestoreRequest restoreRequest = new RestoreRequest(bucket, cosPath);
restoreRequest.setExpireDays(5); // 保留5天
restoreRequest.setTier(RestoreConfigure.Tier.Standard); // 标准恢复模式

cosXmlService.restoreObjectAsync(restoreRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
RestoreResult restoreResult = (RestoreResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

预请求跨域配置

最近更新时间：2022-08-25 09:02:02

简介

本文档提供关于预请求跨域配置操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
Options Object	预请求跨域配置	用预请求来确认是否可以发送真正的跨域请求

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。

预请求跨域配置

功能说明

获取预请求跨域配置（Options Object）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String origin = "https://cloud.tencent.com";
String accessMethod = "PUT";
OptionObjectRequest optionObjectRequest = new OptionObjectRequest(bucket,
cosPath, origin,
accessMethod);
cosXmlService.optionObjectAsync(optionObjectRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
OptionObjectResult optionObjectResult = (OptionObjectResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

服务端加密

最近更新时间：2022-08-25 09:02:06

简介

本文档提供关于如何使用在上传对象时开启服务端加密。服务端加密的密钥分为三种：

- COS 托管加密密钥
- KMS 托管加密密钥
- 客户提供的加密密钥

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。

使用 COS 托管加密密钥的服务端加密（SSE-COS）保护数据

功能说明

由腾讯云 COS 托管主密钥和管理数据。COS 会帮助您在数据写入数据中心时自动加密，并在您取用该数据时自动解密。目前支持使用 COS 主密钥对数据进行 AES-256 加密。

示例代码

```
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
// 设置使用 COS 托管加密密钥的服务端加密（SSE-COS）保护数据
putObjectRequest.setCOSServerSideEncryption();

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

使用 KMS 托管加密密钥的服务端加密（SSE-KMS）保护数据

功能说明

SSE-KMS 加密即使用 KMS 托管密钥的服务端加密。KMS 是腾讯云推出的一款安全管理类服务，使用经过第三方认证的硬件安全模块 HSM（Hardware Security Module）来生成和保护密钥。它能够帮助用户轻松创建和管理密钥，满足用户多应用多业务的密钥管理需求以及满足监管和合规要求。关于如何开通 KMS 服务请参考：服务端加密概述。

示例代码

```
// 服务端加密密钥
String customKey = "用户主密钥 CMK";
String encryptContext = "加密上下文";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);

// 设置使用客户提供的用户主密钥的服务端加密 （SSE-KMS）保护数据
try {
putObjectRequest.setCOSServerSideEncryptionWithKMS(customKey, encryptContext);
} catch (CosXmlClientException e) {
e.printStackTrace();
}
// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
```

版权所有：腾讯云计算（北京）有限责任公司

> 说明：
> 更多完整示例，请前往 GitHub 查看。

使用客户提供的加密密钥的服务端加密（SSE-C）保护数据

功能说明

加密密钥由用户自己提供，用户在上传对象时，COS 将使用用户提供的加密密钥对用户的数据进行 AES-256 加密。

> 注意：
> - 该加密所运行的服务需要使用 HTTPS 请求。
> - 用户需要提供一个32字节的字符串作为密钥，支持数字、字母、字符的组合，不支持中文。
> - 如果上传文件时设置了密钥加密，那么在使用 GET（下载）、HEAD（查询）源对象时也需要在请求中带上相同的密钥，才能正常响应。

示例代码

```
// 服务端加密密钥
String customKey = "服务端加密密钥";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
// 设置使用客户提供的加密密钥的服务端加密 （SSE-C）保护数据
try {
putObjectRequest.setCOSServerSideEncryptionWithCustomerKey(customKey);
} catch (CosXmlClientException e) {
e.printStackTrace();
}

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

单链接限速

最近更新时间：2022-08-25 09:02:11

简介

本文档提供关于调用上传下载接口时对链接进行限速。

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。

使用说明

限速值设置范围为 819200 - 838860800，单位默认为 bit/s，即100KB/s - 100MB/s，如果超出该范围将返回400错误。

示例代码一：上传时对单链接限速

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString(); //本地文件的绝对路径
//若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
// 设置单链接限速，单位为 bit/s，示例设置为 1M/s
putObjectRequest.setTrafficLimit(1024 * 1024 * 8);

// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);

//设置上传进度回调
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
```

版权所有：腾讯云计算（北京）有限责任公司

```
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调, 可以查看任务过程
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码二：下载时对单链接限速

```
//.cssg-snippet-body-start:[transfer-download-object]
// 高级下载接口支持断点续传，所以会在下载前先发起 HEAD 请求获取文件信息。
// 如果您使用的是临时密钥或者使用子账号访问，请确保权限列表中包含 HeadObject 的权限。

// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);

// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
//本地目录路径
String savePathDir = context.getExternalCacheDir().toString();
//本地保存的文件名，若不填（null），则与 COS 上的文件名一样
String savedFileName = "exampleobject";

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathDir, savedFileName);
// 设置单链接限速，单位为 bit/s，示例设置为 1M/s
getObjectRequest.setTrafficLimit(1024 * 1024 * 8);

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext, getObjectRequest);

//设置下载进度回调
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
```

```
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
(COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
//设置任务状态回调，可以查看任务过程
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});
```

说明：

更多完整示例，请前往 GitHub 查看。

检索对象内容

最近更新时间：2022-08-25 09:02:16

简介

本文档提供关于检索对象内容操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
SELECT Object Content	检索对象内容	从指定对象（CSV 格式或者 JSON 格式）中检索内容

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。

检索对象内容

功能说明

COS Select 支持检索以下格式的对象数据：

- CSV 格式：对象以 CSV 格式存储，并以固定的分隔符划分。
- JSON 格式：对象以 JSON 格式存储，可以是 JSON 文件或者 JSON 列表。

> ⚠ 注意：
>
> - 使用 COS Select，您必须具有 cos:GetObject 的授权。
> - CSV、JSON 对象需要以 UTF-8 格式编码。
> - COS Select 支持检索 GZIP 或者 BZIP2 压缩的 CSV、JSON 对象。
> - COS Select 支持检索 SSE-COS 加密的 CSV、JSON 对象。

示例代码

```
String bucket = "examplebucket-1250000000";
// 对象必须为 JSON 或者 csv 格式的文件
String cosPath = "exampleobject";
final String expression = "Select * from COSObject";

SelectObjectContentRequest selectObjectContentRequest = new SelectObjectContentRequest(
bucket, cosPath, expression, true,
new InputSerialization(CompressionType.NONE, new JSONInput(JSONType.DOCUMENT)),
new OutputSerialization(new JSONOutput(","))
);

// 设置查询结果回调，可能会回调多次
selectObjectContentRequest.setSelectObjectContentProgressListener(new SelectObjectContentListener() {
@Override
public void onProcess(SelectObjectContentEvent event) {

}
});
cosXmlService.selectObjectContentAsync(selectObjectContentRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
SelectObjectContentResult selectObjectContentResult =
(SelectObjectContentResult) result;
```

```
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

异地容灾
版本控制

最近更新时间：2022-08-25 09:02:22

简介

本文档提供关于版本控制的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket versioning	设置版本控制	设置存储桶的版本控制功能
GET Bucket versioning	查询版本控制	查询存储桶的版本控制信息

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。

设置版本控制

功能说明

设置指定存储桶的版本控制功能。开启版本控制功能后，只能暂停，不能关闭。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketVersioningRequest putBucketVersioningRequest =
new PutBucketVersioningRequest(bucket);
//true：开启版本控制; false：暂停版本控制
putBucketVersioningRequest.setEnableVersion(true);

cosXmlService.putBucketVersionAsync(putBucketVersioningRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketVersioningResult putBucketVersioningResult =
(PutBucketVersioningResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询版本控制

功能说明

查询指定存储桶的版本控制信息。

- 获取存储桶版本控制的状态，需要有该存储桶的读权限。
- 有三种版本控制状态：未启用版本控制、启用版本控制和暂停版本控制。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketVersioningRequest getBucketVersioningRequest =
new GetBucketVersioningRequest(bucket);

cosXmlService.getBucketVersioningAsync(getBucketVersioningRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketVersioningResult getBucketVersioningResult =
(GetBucketVersioningResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

跨地域复制

最近更新时间：2022-08-25 09:02:26

简介

本文档提供关于跨地域复制的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket replication	设置跨地域复制	设置存储桶的跨地域复制规则
GET Bucket replication	查询跨地域复制	查询存储桶的跨地域复制规则
DELETE Bucket replication	删除跨地域复制	删除存储桶的跨地域复制规则

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。

设置跨地域复制

功能说明

设置指定存储桶的跨地域复制规则。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketReplicationRequest putBucketReplicationRequest =
new PutBucketReplicationRequest(bucket);

String ownerUin = "100000000001"; //发起者身份标示：OwnerUin
String subUin = "100000000001"; //发起者身份标示：SubUin
putBucketReplicationRequest.setReplicationConfigurationWithRole(ownerUin,
subUin);

PutBucketReplicationRequest.RuleStruct ruleStruct =
new PutBucketReplicationRequest.RuleStruct();
//用来标注具体 Rule 的名称
ruleStruct.id = "replication_01";
//标识 Rule 是否生效。true：生效；false：不生效
ruleStruct.isEnable = true;
//目标存储桶地域信息
ruleStruct.region = "ap-beijing";
// 目标存储桶
ruleStruct.bucket = "destinationbucket-1250000000";
//前缀匹配策略
ruleStruct.prefix = "dir/";
putBucketReplicationRequest.setReplicationConfigurationWithRule(ruleStruct);

cosXmlService.putBucketReplicationAsync(putBucketReplicationRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketReplicationResult putBucketReplicationResult =
(PutBucketReplicationResult) result;
}
```

```
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

查询跨地域复制

功能说明

查询指定存储桶的跨地域复制规则。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketReplicationRequest getBucketReplicationRequest =
new GetBucketReplicationRequest(bucket);

cosXmlService.getBucketReplicationAsync(getBucketReplicationRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketReplicationResult getBucketReplicationResult =
(GetBucketReplicationResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

> 更多完整示例，请前往 GitHub 查看。

删除跨地域复制

功能说明

删除指定存储桶的跨地域复制规则。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketReplicationRequest deleteBucketReplicationRequest =
new DeleteBucketReplicationRequest(bucket);

cosXmlService.deleteBucketReplicationAsync(deleteBucketReplicationRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketReplicationResult deleteBucketReplicationResult =
(DeleteBucketReplicationResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

数据管理
生命周期

最近更新时间：2022-08-25 09:02:32

简介

本文档提供关于生命周期的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket lifecycle	设置生命周期	设置存储桶的生命周期管理的配置
GET Bucket lifecycle	查询生命周期	查询存储桶生命周期管理的配置
DELETE Bucket lifecycle	删除生命周期	删除存储桶生命周期管理的配置

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。

设置生命周期

功能说明

设置指定存储桶的生命周期配置信息。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketLifecycleRequest putBucketLifecycleRequest =
new PutBucketLifecycleRequest(bucket);

// 声明周期配置规则信息
LifecycleConfiguration.Rule rule = new LifecycleConfiguration.Rule();
rule.id = "Lifecycle ID";
LifecycleConfiguration.Filter filter = new LifecycleConfiguration.Filter();
// 指定规则所适用的前缀
filter.prefix = "dir/";
rule.filter = filter;
// 指明规则是否启用
rule.status = "Enabled";
// 指明规则对应的动作在对象最后的修改日期过后多少天操作
LifecycleConfiguration.Transition transition =
new LifecycleConfiguration.Transition();
transition.days = 100;
transition.storageClass = COSStorageClass.STANDARD.getStorageClass();
rule.transition = transition;

putBucketLifecycleRequest.setRuleList(rule);

cosXmlService.putBucketLifecycleAsync(putBucketLifecycleRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketLifecycleResult putBucketLifecycleResult =
(PutBucketLifecycleResult) result;
```

版权所有：腾讯云计算（北京）有限责任公司

```
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

查询生命周期

功能说明

查询存储桶的生命周期管理配置。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketLifecycleRequest getBucketLifecycleRequest =
new GetBucketLifecycleRequest(bucket);

cosXmlService.getBucketLifecycleAsync(getBucketLifecycleRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketLifecycleResult getBucketLifecycleResult =
(GetBucketLifecycleResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除生命周期

功能说明

删除存储桶生命周期管理的配置。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
new DeleteBucketLifecycleRequest(bucket);

cosXmlService.deleteBucketLifecycleAsync(deleteBucketLifecycleRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketLifecycleResult deleteBucketLifecycleResult =
(DeleteBucketLifecycleResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

日志管理

最近更新时间：2022-08-25 09:02:37

简介

本文档提供关于日志管理的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket logging	设置日志管理	为源存储桶开启日志记录
GET Bucket logging	查询日志管理	查询源存储桶的日志配置信息

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。

设置日志管理

功能说明

PUT Bucket logging 用于为源存储桶开启日志记录，将源存储桶的访问日志保存到指定的目标存储桶中。

示例代码

```
String srcBucket = "examplebucket-1250000000"; //格式：BucketName-APPID
String targetBucket = "examplebucket-1250000000"; //格式：BucketName-APPID
PutBucketLoggingRequest putBucketLoggingRequest =
new PutBucketLoggingRequest(srcBucket);
// 目标存储桶
putBucketLoggingRequest.setTargetBucket(targetBucket);
// 日志存储的指定位置
putBucketLoggingRequest.setTargetPrefix("dir/");

cosXmlService.putBucketLoggingAsync(putBucketLoggingRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketLoggingResult putBucketLoggingResult =
(PutBucketLoggingResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询日志管理

功能说明

GET Bucket logging 用于查询指定存储桶的日志配置信息。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketLoggingRequest getBucketLoggingRequest =
new GetBucketLoggingRequest(bucket);

cosXmlService.getBucketLoggingAsync(getBucketLoggingRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketLoggingResult getBucketLoggingResult =
(GetBucketLoggingResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

存储桶标签

最近更新时间：2022-08-25 09:02:42

简介

本文档提供关于存储桶标签的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket tagging	设置存储桶标签	为已存在的存储桶设置标签
GET Bucket tagging	查询存储桶标签	查询指定存储桶下已有的存储桶标签
DELETE Bucket tagging	删除存储桶标签	删除指定的存储桶标签

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。

设置存储桶标签

功能说明

PUT Bucket tagging 用于为已存在的存储桶设置标签。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketTaggingRequest putBucketTaggingRequest =
new PutBucketTaggingRequest(bucket);
// 设置标签
putBucketTaggingRequest.addTag("key", "value");
putBucketTaggingRequest.addTag("hello", "world");

cosXmlService.putBucketTaggingAsync(putBucketTaggingRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketTaggingResult putBucketTaggingResult =
(PutBucketTaggingResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询存储桶标签

功能说明

GET Bucket tagging 用于查询指定存储桶下已有的存储桶标签。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketTaggingRequest getBucketTaggingRequest =
new GetBucketTaggingRequest(bucket);

cosXmlService.getBucketTaggingAsync(getBucketTaggingRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketTaggingResult getBucketTaggingResult =
(GetBucketTaggingResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除存储桶标签

功能说明

DELETE Bucket tagging 用于删除指定存储桶下已有的存储桶标签。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketTaggingRequest deleteBucketTaggingRequest =
new DeleteBucketTaggingRequest(bucket);

cosXmlService.deleteBucketTaggingAsync(deleteBucketTaggingRequest,
new CosXmlResultListener() {
```

版权所有：腾讯云计算（北京）有限责任公司

```
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketTaggingResult getBucketTaggingResult =
(DeleteBucketTaggingResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}

}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

对象标签

最近更新时间：2022-08-25 09:02:46

简介

本文档提供关于对象标签的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Object tagging	设置对象标签	为已上传的对象设置标签
GET Object tagging	查询对象标签	查询指定对象下已有的对象标签
DELETE Object tagging	删除对象标签	删除指定对象下已有的对象标签

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。

设置对象标签

上传时添加标签

功能说明

上传对象时，可以通过为请求添加特定的 Header 来给对象打标签，例如设置 x-cos-tagging 的值为 Key1=Value1&Key2=Value2，标签集合中的 Key 和 Value 必须先进行 URL 编码。

示例代码

```
// 初始化 TransferConfig，这里使用默认配置，如果需要定制，请参考 SDK 接口文档
TransferConfig transferConfig = new TransferConfig.Builder().build();
//初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
transferConfig);
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = new File(context.getCacheDir(), "exampleobject")
.toString(); //本地文件的绝对路径
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
try {
// 设置对象标签，标签集合中的 Key 和 Value 必须先进行 URL 编码
putObjectRequest.setRequestHeaders("x-cos-tagging", "Key1=Value&Key2=Value2", false);
} catch (CosXmlClientException e) {
e.printStackTrace();
}
// 若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
String uploadId = null;
// 上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
srcPath, uploadId);
//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {

@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
```

```
(COSXMLUploadTask.COSXMLUploadTaskResult) result;
}
// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

为已存在的对象添加标签

功能说明

COS 支持为已存在的对象设置标签。通过为对象添加键值对作为对象标签，可以协助您分组管理已有的对象资源。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
PutObjectTaggingRequest putObjectTaggingRequest = new PutObjectTaggingRequest(bucket, cosPath);
putObjectTaggingRequest.addTag("key", "value");
try {
PutObjectTaggingResult putObjectTaggingResult = cosXmlService.putObjectTagging(putObjectTaggingRequest);
} catch (CosXmlClientException clientException) {
clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
serviceException.printStackTrace();
}
```

说明：
更多完整示例，请前往 GitHub 查看。

查询对象标签

功能说明

查询指定对象下已有的对象标签。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
GetObjectTaggingRequest getObjectTaggingRequest = new GetObjectTaggingRequest(bucket, cosPath);
```

```
try {
GetObjectTaggingResult getObjectTaggingResult = cosXmlService.getObjectTagging(getObjectTaggingRequest);
} catch (CosXmlClientException clientException) {
clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
serviceException.printStackTrace();
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除对象标签

功能说明

删除指定对象下已有的对象标签。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
DeleteObjectTaggingRequest deleteObjectTaggingRequest = new DeleteObjectTaggingRequest(bucket, cosPath);
try {
DeleteObjectTaggingResult deleteObjectTaggingResult = cosXmlService.deleteObjectTagging(deleteObjectTaggingRequest);
} catch (CosXmlClientException clientException) {
clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
serviceException.printStackTrace();
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

静态网站

最近更新时间：2022-08-25 09:02:52

简介

本文档提供关于静态网站的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket website	设置静态网站	设置存储桶的静态网站配置
GET Bucket website	查询静态网站配置	查询存储桶的静态网站配置
DELETE Bucket website	删除静态网站配置	删除存储桶的静态网站配置

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。

设置静态网站

功能说明

PUT Bucket website 用于为存储桶配置静态网站。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketWebsiteRequest putBucketWebsiteRequest =
new PutBucketWebsiteRequest(bucket);
// 设置 index 文档
putBucketWebsiteRequest.setIndexDocument("index.html");

cosXmlService.putBucketWebsiteAsync(putBucketWebsiteRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketWebsiteResult putBucketWebsiteResult =
(PutBucketWebsiteResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

更多完整示例，请前往 GitHub 查看。

查询静态网站配置

功能说明

GET Bucket website 用于查询与存储桶关联的静态网站配置信息。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketWebsiteRequest getBucketWebsiteRequest =
new GetBucketWebsiteRequest(bucket);
cosXmlService.getBucketWebsiteAsync(getBucketWebsiteRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketWebsiteResult getBucketWebsiteResult =
(GetBucketWebsiteResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

删除静态网站配置

功能说明

DELETE Bucket website 用于删除存储桶中的静态网站配置。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketWebsiteRequest deleteBucketWebsiteRequest =
new DeleteBucketWebsiteRequest(bucket);

cosXmlService.deleteBucketWebsiteAsync(deleteBucketWebsiteRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
```

```
DeleteBucketWebsiteResult getBucketWebsiteResult =
(DeleteBucketWebsiteResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

清单

最近更新时间：2022-08-25 09:02:56

简介

本文档提供关于清单的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket inventory	设置清单任务	设置存储桶的清单任务
GET Bucket inventory	查询清单任务	查询存储桶的清单任务
DELETE Bucket inventory	删除清单任务	删除存储桶的清单任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

设置清单任务

功能说明

PUT Bucket inventory 用于在存储桶中创建清单任务。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketInventoryRequest putBucketInventoryRequest =
new PutBucketInventoryRequest(bucket);
putBucketInventoryRequest.setInventoryId("exampleInventoryId");
// 是否在清单中包含对象版本：
// 如果设置为 All，清单中将会包含所有对象版本，
// 并在清单中增加VersionId，IsLatest，DeleteMarker 这几个字段
// 如果设置为 Current，则清单中不包含对象版本信息
putBucketInventoryRequest.setIncludedObjectVersions(InventoryConfiguration
.IncludedObjectVersions.ALL);
// 备份频率
putBucketInventoryRequest.setScheduleFrequency(InventoryConfiguration
.SCHEDULE_FREQUENCY_DAILY);
// 备份路径
putBucketInventoryRequest.setDestination("CSV", "1000000000",
"examplebucket-1250000000", "region", "dir/");

cosXmlService.putBucketInventoryAsync(putBucketInventoryRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketInventoryResult putBucketInventoryResult =
(PutBucketInventoryResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
```

```
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

错误码说明

该请求可能会发生的一些常见的特殊错误如下：

错误码	描述	状态码
InvalidArgument	不合法的参数值	HTTP 400 Bad Request
TooManyConfigurations	清单数量已经达到1000条的上限	HTTP 400 Bad Request
AccessDenied	未授权的访问。您可能不具备访问该存储桶的权限	HTTP 403 Forbidden

查询清单任务

功能说明

GET Bucket inventory 用于查询存储桶中用户的清单任务信息。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketInventoryRequest getBucketInventoryRequest =
new GetBucketInventoryRequest(bucket);
getBucketInventoryRequest.setInventoryId("exampleInventoryId");

cosXmlService.getBucketInventoryAsync(getBucketInventoryRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketInventoryResult getBucketInventoryResult =
(GetBucketInventoryResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
```

```
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除清单任务

功能说明

DELETE Bucket inventory 用于删除存储桶中指定的清单任务。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketInventoryRequest deleteBucketInventoryRequest =
new DeleteBucketInventoryRequest(bucket);
deleteBucketInventoryRequest.setInventoryId("exampleInventoryId");

cosXmlService.deleteBucketInventoryAsync(deleteBucketInventoryRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketInventoryResult deleteBucketInventoryResult =
(DeleteBucketInventoryResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

图片处理
图片持久化处理

最近更新时间：2022-08-25 09:03:00

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
	样式设置	设置图片的样式，方便管理不同需求的图片

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。

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

```
List<PicOperationRule> rules = new LinkedList<>();
// 添加一条将图片转化为 png 格式的 rule，处理后的图片在存储桶中的位置标识符为
// examplepngobject
rules.add(new PicOperationRule("examplepngobject", "imageView2/format/png"));
PicOperations picOperations = new PicOperations(true, rules);

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPicOperations(picOperations);

// 上传成功后，您将会得到 2 张图片，分别是原始图片和处理后图片
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
```

说明：

更多完整示例，请前往 GitHub 查看。

基础图片处理

最近更新时间：2022-08-25 09:03:04

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

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。

缩放

示例代码

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String savePath = context.getExternalCacheDir().toString(); //本地路径

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
savePath);
getObjectRequest.addQuery("imageMogr2/thumbnail/!50p", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest,
CosXmlResult cosXmlResult) {
GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
```

```
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

裁剪

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

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String savePath = context.getExternalCacheDir().toString(); //本地路径

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
savePath);
getObjectRequest.addQuery("imageMogr2/iradius/150", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest,
CosXmlResult cosXmlResult) {
GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

旋转

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

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String savePath = context.getExternalCacheDir().toString(); //本地路径
```

```
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
savePath);
getObjectRequest.addQuery("imageMogr2/rotate/90", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest,
CosXmlResult cosXmlResult) {
GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：

更多完整示例，请前往 GitHub 查看。

图片高级压缩

最近更新时间：2022-08-25 09:03:07

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

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。

图片高级压缩

功能说明

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：下载时进行高级压缩

```
String bucket = "examplebucket-1250000000"; //存储桶名称，格式：BucketName-APPID
String cosPath = "exampleobject"; //对象位于存储桶中的位置标识符，即对象键
String savePath = context.getExternalCacheDir().toString(); //本地路径

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
savePath);
getObjectRequest.addQuery("imageMogr2/format/tpg", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest,
CosXmlResult cosXmlResult) {
GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

盲水印

最近更新时间：2022-08-25 09:03:11

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

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。

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

```
List<PicOperationRule> rules = new LinkedList<>();
// 添加一条将盲水印 rule，处理后的图片在存储桶中的位置标识符为
// examplewatermarkobject
rules.add(new PicOperationRule("examplewatermarkobject",
"watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAw"));
PicOperations picOperations = new PicOperations(true, rules);

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPicOperations(picOperations);

// 上传成功后，您将会得到 2 张图片，分别是原始图片和处理后图片
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
```

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, cosPath, savePathDir, savedFileName);
// 添加文字盲水印
getObjectRequest.addQuery("watermark/3/type/3/text/dGVuY2VudCBjbG91ZA==", null);

COSXMLDownloadTask cosxmlDownloadTask =
transferManager.download(applicationContext, getObjectRequest);
```

说明：
更多完整示例，请前往 GitHub 查看。

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：2022-08-25 09:03:15

简介

本文档提供关于媒体 bucket 接口的 API 概览以及 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

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。

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 Android SDK 版本需要大于等于 v5.7.6。

示例代码

```
GetDescribeMediaBucketsRequest request = new GetDescribeMediaBucketsRequest();
// 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
request.setRegions("ap-guangzhou,ap-beijing");
// 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
request.setBucketNames("examplebucket-1250000000");
// 存储桶名称前缀，前缀搜索
request.setBucketName("example");
// 第几页
request.setPageNumber(1);
// 每页个数
request.setPageSize(20);
cosXmlService.getDescribeMediaBucketsAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 查询到的已经开通媒体处理功能的存储桶
// 详细字段请查看api文档或者SDK源码 DescribeMediaBucketsResult类
GetDescribeMediaBucketsResult result = (GetDescribeMediaBucketsResult) cosResult;
}

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

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：2022-08-25 09:03:19

简介

本文档提供关于媒体信息接口的 API 概览以及 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

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。

查询文件信息

功能说明

用于查询媒体文件的信息。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.7.6。

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 = "exampleobject";
GetMediaInfoRequest request = new GetMediaInfoRequest(bucket, cosPath);
cosXmlService.getMediaInfoAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 获取到的媒体文件的信息
// 详细字段请查看api文档或者SDK源码 MediaInfo类
GetMediaInfoResult result = (GetMediaInfoResult) cosResult;
}

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

> ⓘ 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：2022-08-25 09:03:22

简介

本文档提供关于媒体截图接口的 API 概览以及 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

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。

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 Android SDK 版本需要大于等于 v5.7.6。

示例代码

```
//存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
//文档位于存储桶中的位置标识符，即对象键
String cosPath = "exampleobject.mp4";
// 保存在本地文件夹的路径
String localPath = "localdownloadpath";
// 截图的本地文件名
String fileName = "snapshot.jpg";
// 截帧的时间，单位 s
float time = 1;
GetSnapshotRequest getSnapshotRequest = new GetSnapshotRequest(bucket, cosPath, localPath,
fileName, time);
//截图的宽。默认为0
getSnapshotRequest.setWidth(100);
//截图的高。默认为0
getSnapshotRequest.setHeight(100);
//截图的格式，支持 jpg 和 png，默认 jpg
getSnapshotRequest.setFormat("jpg");
//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
getSnapshotRequest.setRotate("auto");
//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
getSnapshotRequest.setMode("exactframe");

cosXmlService.getSnapshotAsync(getSnapshotRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
GetSnapshotResult result = (GetSnapshotResult) cosResult;
}

@Override
```

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

说明：

更多完整示例，请前往 GitHub 查看。

文档处理
文档预览

最近更新时间：2022-08-25 09:03:28

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	操作描述
同步请求接口	用于同步请求文档预览功能

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。

同步请求接口

功能说明

文档预览功能同步请求。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
String localPath = "localdownloadpath"; // 保存在本地文件夹的路径
int page = 1; // 需转换的文档页码，从 1 开始
PreviewDocumentRequest previewDocumentRequest = new PreviewDocumentRequest(bucket,
cosPath, localPath, page);

cosXmlService.previewDocumentAsync(previewDocumentRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PreviewDocumentResult previewDocumentResult = (PreviewDocumentResult) result;
}

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

内容审核
图片审核

最近更新时间：2022-10-14 15:38:50

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String key = "exampleobject"; //对象键
SensitiveContentRecognitionRequest sensitiveContentRecognitionRequest = new SensitiveContentRecognitionRequest(bucket, key);
sensitiveContentRecognitionRequest.addType("ads");
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.sensitiveContentRecognitionAsync(sensitiveContentRecognitionRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
SensitiveContentRecognitionResult sensitiveContentRecognitionResult = (SensitiveContentRecognitionResult) result;
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException, CosXmlServiceException serviceException) {
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.ImagesAuditInput();
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.ImagesAuditInput();
image2.object = cosPath2;
image2.dataId = "DataId2";
image2.interval = 2;
image2.maxFrames = 5;
PostImagesAudit.ImagesAuditInput image3 = new PostImagesAudit.ImagesAuditInput();
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，例如：Porn,Ads。
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 clientException, CosXmlServiceException serviceException) {
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 clientException, CosXmlServiceException serviceException) {
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：2022-10-14 15:39:00

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）。默认为 Simple。
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，例如：Porn,Ads。
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面截图和视频声音。默认值为0。
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 clientException, CosXmlServiceException serviceException) {
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 clientException, CosXmlServiceException serviceException) {
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：2022-10-14 15:39:11

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）。默认为 Simple。
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，例如：Porn,Ads。
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 clientException, CosXmlServiceException serviceException) {
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 clientException, CosXmlServiceException serviceException) {
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：2022-10-14 15:39:18

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 utf8 编码字符。若超出长度限制，接口将会报错。
String content = Base64.encodeToString("测试文本".getBytes(Charset.forName("UTF-8")), Base64.NO_WRAP);
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）。默认为 Simple。
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，例如：Porn,Ads。
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 clientException, CosXmlServiceException serviceException) {
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 clientException, CosXmlServiceException serviceException) {
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：2022-10-14 15:39:26

简介

版权所有：腾讯云计算（北京）有限责任公司

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，例如：Porn,Ads。
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 clientException, CosXmlServiceException serviceException) {
```

```
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 clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

网页审核

最近更新时间：2022-08-25 09:03:50

简介

本文档提供关于网页审核接口的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PostWebPageAudit	提交网页审核任务	用于提交网页审核任务
GetWebPageAudit	查询网页审核任务结果	用于查询网页审核任务结果

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。

提交网页审核任务

功能说明

用于提交网页审核任务。

> ⚠ 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"
//网页的链接地址
String url = "https://github.com";
PostWebPageAuditRequest request = new PostWebPageAuditRequest(bucket);
request.setUrl(url);
//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，例如：Porn,Ads。
request.setDetectType("Porn,Ads");
//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。
request.setReturnHighlightHtml(true);

// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.postWebPageAuditAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 提交网页审核任务的结果
// 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 clientException, CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
```

```
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询网页审核任务结果

功能说明

用于查询网页审核任务结果。

> 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";
GetWebPageAuditRequest request = new GetWebPageAuditRequest(bucket, jobId);
// CIService 是 CosXmlService 的子类，初始化方法和 CosXmlService 一致
ciService.getWebPageAuditAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 查询网页审核任务的结果
// 详细字段请查看api文档或者SDK源码
GetWebPageAuditResult result = (GetWebPageAuditResult) cosResult;
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

AI 内容识别
图片二维码

最近更新时间：2022-08-25 09:03:55

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

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。

识别二维码

功能说明

二维码支持在上传时添加以及下载时识别。

示例代码：上传时识别二维码

```
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.pdf"; //文档位于存储桶中的位置标识符，即对象键
String localPath = "localdownloadpath"; // 二维码图片本地路径
final PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, localPath);

List<PicOperationRule> rules = new LinkedList<>();
rules.add(new PicOperationRule("/test.png", "QRcode/cover/0"));
PicOperations picOperations = new PicOperations(false, rules);
putObjectRequest.setPicOperations(picOperations);

cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutObjectResult putObjectResult = (PutObjectResult) result;
PicUploadResult picUploadResult = putObjectResult.picUploadResult; // 图片处理结局
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

语音识别
任务接口

最近更新时间：2022-08-25 09:03:59

简介

版权所有：腾讯云计算（北京）有限责任公司

本文档提供关于提交语音识别任务、查询指定的语音识别任务、批量拉取语音识别任务相关的 API 概览以及 SDK 示例代码。

API	说明
提交语音识别任务	接口用于提交一个语音识别任务。
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	用于拉取符合条件的语音识别任务。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

提交语音识别任务

功能说明

接口用于提交一个语音识别任务。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.9.2。

示例代码

```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
CreateSpeechJobsRequest request = new CreateSpeechJobsRequest(bucket);
// 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 inputPath = "dir1/input.m4a";
//设置语音文件在 COS 上的 key
request.setInputObject(inputPath);
//设置任务所在的队列 ID
request.setQueueId("p74b5265ab1df466682b7b355007d0dfc");
//输出cos路径
String outputPath = "dir1/putput.txt";
//设置结果输出地址
request.setOutput("ap-guangzhou", bucket, outputPath);
//设置引擎模型类型。
//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）；
//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。
request.setEngineModelType("8k_zh");
//设置语音声道数。1：单声道；2：双声道（仅支持 8k_zh 引擎模型）
request.setChannelNum(2);
//设置识别结果返回形式。
// 0： 识别结果文本(含分段时间戳)； 1：仅支持16k中文引擎，含识别结果详情(词时间戳列表，一般用于生成字幕场景)
request.setResTextFormat(0);

ciService.createSpeechJobsAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 提交语音识别任务的结果
```

```
// 详细字段请查看api文档或者SDK源码
CreateSpeechJobsResult result = (CreateSpeechJobsResult) cosResult;
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询指定的语音识别任务

功能说明

用于查询指定的语音识别任务。

> 注意：
> COS Android SDK 版本需要大于等于 v5.9.2。

示例代码

```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
//语音识别任务的jobId（提交语音识别任务结果中获取）
String jobId = "s3841e6aa0cbd11ed923405b602cab698";
DescribeSpeechJobRequest request = new DescribeSpeechJobRequest(bucket, jobId);
ciService.describeSpeechJobAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 查询指定的语音识别任务的结果
// 详细字段请查看api文档或者SDK源码
DescribeSpeechJobResult result = (DescribeSpeechJobResult) cosResult;
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

批量拉取语音识别任务

功能说明

用于拉取符合条件的语音识别任务。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.9.2。

示例代码

```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
//要查询的队列id
String queueId = "p74b5265ab1df466682b7b355007d0dfc";
DescribeSpeechJobsRequest request = new DescribeSpeechJobsRequest(bucket, queueId);
//Desc 或者 Asc。默认为 Desc。
request.setOrderByTime("Desc");
//请求的上下文，用于翻页。上次返回的值。
request.setNextToken("NextToken");
//拉取的最大任务数。默认为10。最大为100。
request.setSize(50);
//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。
request.setStates("All");
//拉取创建时间大于该时间的任务。格式为：%Y-%m-%dT%H:%m:%S%z
request.setStartCreationTime("%Y-%m-%dT%H:%m:%S%z");
//拉取创建时间小于该时间的任务。格式为：%Y-%m-%dT%H:%m:%S%z
request.setEndCreationTime("%Y-%m-%dT%H:%m:%S%z");

ciService.describeSpeechJobsAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 拉取符合条件的语音识别任务的结果
// 详细字段请查看api文档或者SDK源码
DescribeSpeechJobsResult result = (DescribeSpeechJobsResult) cosResult;
}

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

> ? 说明：
> 更多完整示例，请前往 GitHub 查看。

语音 bucket 接口

最近更新时间：2022-08-25 09:04:03

简介

本文档提供关于查询语音识别开通状态相关的 API 概览以及 SDK 示例代码。

API	说明
查询语音识别开通状态	接口用于查询存储桶是否已开通语音识别功能。

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。

查询语音识别开通状态

功能说明

接口用于查询存储桶是否已开通语音识别功能。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.9.2。

示例代码

```
DescribeSpeechBucketsRequest request = new DescribeSpeechBucketsRequest();
//设置第几页
request.setPageNumber(1);
//设置每页个数
request.setPageSize(10);
//设置地域信息 以“,”分隔字符串，支持 All、ap-shanghai、ap-beijing
request.setRegions("All");
//设置存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
//request.setBucketNames("examplebucket-1250000000,examplebucket123-1250000000");
//设置存储桶名称前缀，前缀搜索
//request.setBucketName("examplebucket");
ciService.describeSpeechBucketsAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 查询已经开通语音识别功能的存储桶的结果
// 详细字段请查看api文档或者SDK源码
DescribeSpeechBucketsResult result = (DescribeSpeechBucketsResult) cosResult;
}

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

说明：
更多完整示例，请前往 GitHub 查看。

队列接口

最近更新时间：2022-08-25 09:04:07

简介

本文档提供关于查询语音识别队列相关的 API 概览以及 SDK 示例代码。

API	说明
查询语音识别队列	接口用于查询语音识别队列。

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。

查询语音识别队列

功能说明

接口用于查询语音识别队列。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.9.2。

示例代码

```
// 存储桶名称，格式为 BucketName-APPID
String bucket = "examplebucket-1250000000";
DescribeSpeechQueuesRequest request = new DescribeSpeechQueuesRequest(bucket);
//设置第几页
request.setPageNumber(1);
//设置每页个数
request.setPageSize(10);
//设置状态
// Active 表示队列内的作业会被语音识别服务调度执行
// Paused 表示队列暂停，作业不再会被语音识别服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于识别中的任务将继续执行，不受影响
request.setState("Active");
//设置队列 ID，以“,”符号分割字符串
//request.setQueueIds("p74b5265ab1df466682b7b355007d0dfc"+",p74b9995ab1df455782b7b355007d0dfc");
ciService.describeSpeechQueuesAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult cosResult) {
// result 查询语音识别队列的结果
// 详细字段请查看api文档或者SDK源码
DescribeSpeechQueuesResult result = (DescribeSpeechQueuesResult) cosResult;
}

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

访问管理
跨域访问

最近更新时间：2022-08-25 09:04:12

简介

本文档提供关于跨域访问的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket cors	设置跨域配置	设置存储桶的跨域名访问权限
GET Bucket cors	查询跨域配置	查询存储桶的跨域名访问配置信息
DELETE Bucket cors	删除跨域配置	删除存储桶的跨域名访问配置信息

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。

设置跨域配置

功能说明

设置指定存储桶的跨域名访问配置信息（PUT Bucket cors）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketCORSRequest putBucketCORSRequest = new PutBucketCORSRequest(bucket);

CORSConfiguration.CORSRule corsRule = new CORSConfiguration.CORSRule();

// 配置规则的 ID
corsRule.id = "123";
// 允许的访问来源，支持通配符 *，格式为：协议://域名[:端口]
corsRule.allowedOrigin = "https://cloud.tencent.com";
// 设置 OPTIONS 请求得到结果的有效期
corsRule.maxAgeSeconds = 5000;

List<String> methods = new LinkedList<>();
methods.add("PUT");
methods.add("POST");
methods.add("GET");
// 允许的 HTTP 操作，例如：GET，PUT，HEAD，POST，DELETE
corsRule.allowedMethod = methods;

List<String> headers = new LinkedList<>();
headers.add("host");
headers.add("content-type");
// 在发送 OPTIONS 请求时告知服务端，接下来的请求可以使用的 HTTP 请求头部，支持通配符 *
corsRule.allowedHeader = headers;

List<String> exposeHeaders = new LinkedList<>();
exposeHeaders.add("x-cos-meta-1");
// 设置浏览器可以接收到的来自服务端的自定义头部信息
```

版权所有：腾讯云计算（北京）有限责任公司

```
corsRule.exposeHeader = exposeHeaders;

putBucketCORSRequest.addCORSRule(corsRule);

cosXmlService.putBucketCORSAsync(putBucketCORSRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketCORSResult putBucketCORSResult = (PutBucketCORSResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

查询跨域配置

功能说明

查询指定存储桶的跨域名访问配置信息（GET Bucket cors）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketCORSRequest getBucketCORSRequest = new GetBucketCORSRequest(bucket);
cosXmlService.getBucketCORSAsync(getBucketCORSRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketCORSResult getBucketCORSResult = (GetBucketCORSResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
```

```
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除跨域配置

功能说明

删除指定存储桶的跨域名访问配置（DELETE Bucket cors）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
DeleteBucketCORSRequest deleteBucketCORSRequest =
new DeleteBucketCORSRequest(bucket);
cosXmlService.deleteBucketCORSAsync(deleteBucketCORSRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
DeleteBucketCORSResult deleteBucketCORSResult =
(DeleteBucketCORSResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

自定义域名

最近更新时间：2022-08-25 09:04:15

简介

本文档提供关于自定义域名的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket domain	设置自定义域名	设置存储桶的自定义域名信息
GET Bucket domain	查询自定义域名	查询存储桶的自定义域名信息

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。

设置自定义域名

功能说明

PUT Bucket domain 用于为存储桶配置自定义域名。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketDomainRequest putBucketDomainRequest =
new PutBucketDomainRequest(bucket);
DomainConfiguration.DomainRule domainRule = new DomainConfiguration.DomainRule(
DomainConfiguration.STATUS_ENABLED,
"www.example.com",
DomainConfiguration.TYPE_REST
);
domainRule.forcedReplacement = DomainConfiguration.REPLACE_CNAME;
putBucketDomainRequest.addDomainRule(domainRule);

cosXmlService.putBucketDomainAsync(putBucketDomainRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketDomainResult putBucketDomainResult =
(PutBucketDomainResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
```

```
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

返回错误码说明

该请求可能会发生的一些常见的特殊错误如下：

状态码	说明
HTTP 409 Conflict	该域名记录已存在，且请求中没有设置强制覆盖。或者该域名记录不存在，且请求中设置了强制覆盖
HTTP 451 Unavailable For Legal Reasons	该域名是中国境内域名，并且没有备案

查询自定义域名

功能说明

GET Bucket domain 用于查询存储桶的自定义域名信息。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketDomainRequest getBucketDomainRequest =
new GetBucketDomainRequest(bucket);
cosXmlService.getBucketDomainAsync(getBucketDomainRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketDomainResult getBucketTaggingResult =
(GetBucketDomainResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

返回参数说明

参数名称	描述	类型

参数名称	描述	类型
x-cos-domain-txt-verification	域名校验信息，该字段是一个 MD5 校验值，原串格式为：cos[Region][BucketName-APPID][BucketCreateTime]，其中 Region 为存储桶所在地域，BucketCreateTime 为存储桶 GMT 创建时间	String

访问控制

最近更新时间：2022-08-25 09:04:19

简介

本文档提供关于存储桶、对象的访问控制列表（ACL）的相关 API 概览以及 SDK 示例代码。

存储桶 ACL

API	操作名	操作描述
PUT Bucket acl	设置存储桶 ACL	设置指定存储桶的访问权限控制列表（ACL）
GET Bucket acl	查询存储桶 ACL	查询指定存储桶的访问权限控制列表（ACL）

对象 ACL

API	操作名	操作描述
PUT Object acl	设置对象 ACL	设置存储桶中某个对象的访问控制列表
GET Object acl	查询对象 ACL	查询对象的访问控制列表

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。

存储桶 ACL

设置存储桶 ACL

功能说明

设置指定存储桶的访问权限控制列表（ACL）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
PutBucketACLRequest putBucketACLRequest = new PutBucketACLRequest(bucket);

// 设置 bucket 访问权限
putBucketACLRequest.setXCOSACL("public-read");

// 赋予被授权者读的权限
ACLAccount readACLS = new ACLAccount();
readACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSGrantRead(readACLS);

// 赋予被授权者写的权限
ACLAccount writeACLS = new ACLAccount();
writeACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSGrantWrite(writeACLS);

// 赋予被授权者读写的权限
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSReadWrite(writeandReadACLS);

cosXmlService.putBucketACLAsync(putBucketACLRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutBucketACLResult putBucketACLResult = (PutBucketACLResult) result;
```

```
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

查询存储桶 ACL

功能说明

查询指定存储桶的访问权限控制列表（ACL）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
GetBucketACLRequest getBucketACLRequest = new GetBucketACLRequest(bucket);
cosXmlService.getBucketACLAsync(getBucketACLRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetBucketACLResult getBucketACLResult = (GetBucketACLResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

说明：
更多完整示例，请前往 GitHub 查看。

对象 ACL

设置对象 ACL

功能说明

设置存储桶中某个对象的访问控制列表（ACL）。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。
PutObjectACLRequest putObjectACLRequest = new PutObjectACLRequest(bucket,
cosPath);

// 设置 对象 访问权限
putObjectACLRequest.setXCOSACL("public-read");

// 赋予被授权者读的权限
ACLAccount readACLS = new ACLAccount();
readACLS.addAccount("100000000001", "100000000001");
putObjectACLRequest.setXCOSGrantRead(readACLS);

// 赋予被授权者读写的权限
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addAccount("100000000001", "100000000001");
putObjectACLRequest.setXCOSReadWrite(writeandReadACLS);

cosXmlService.putObjectACLAsync(putObjectACLRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
PutObjectACLResult putObjectACLResult = (PutObjectACLResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询对象 ACL

功能说明

查询对象的访问控制列表。

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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即对象键。
GetObjectACLRequest getBucketACLRequest = new GetObjectACLRequest(bucket,
cosPath);
cosXmlService.getObjectACLAsync(getBucketACLRequest,
new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
GetObjectACLResult getObjectACLResult = (GetObjectACLResult) result;
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest cosXmlRequest,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

数据校验
CRC64 校验

最近更新时间：2022-08-25 09:04:25

简介

数据在客户端和服务器间传输时可能会出现错误，对象存储（Cloud Object Storage，COS）除了可以通过 MD5 和自定义属性 验证数据完整性外，还可以通过 CRC64 检验码来进行数据校验。

COS 会对新上传的对象进行 CRC64 计算，并将结果作为对象的属性进行存储，随后在返回的响应头部中携带 x-cos-hash-crc64ecma，该头部表示上传对象的 CRC64 值，根据 ECMA-182标准 计算得到。对于 CRC64 特性上线前就已经存在于 COS 的对象，COS 不会对其计算 CRC64 值，所以获取此类对象时不会返回其 CRC64 值。

操作说明

目前支持 CRC64 的 API 如下：

- 简单上传接口
  - PUT Object 和 POST Object：用户可在返回的响应头中获得文件 CRC64 校验值。
- 分块上传接口
  - Upload Part：用户可以根据 COS 返回的 CRC64 值与本地计算的数值进行比较验证。
  - Complete Multipart Upload：如果每个分块都有 CRC64 属性，则会返回整个对象的 CRC64 值，如果某些分块不具备 CRC64 值，则不返回。
- 执行 Upload Part - Copy 时，会返回对应的 CRC64 值。
- 执行 PUT Object - Copy 时，如果源对象存在 CRC64 值，则返回 CRC64，否则不返回。
- 执行 HEAD Object 和 GET Object 时，如果对象存在 CRC64，则返回。用户可以根据 COS 返回的 CRC64 值和本地计算的 CRC64 进行比较验证。

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。

SDK 说明

您在上传或者下载成功后，可以在响应头部中获取 CRC64 值。

> ⚠ 注意：
> COS Android SDK 版本需要大于等于 v5.7.5。

上传请求示例

```
// 1. 初始化 TransferService。在相同配置的情况下，您应该复用同一个 TransferService
TransferConfig transferConfig = new TransferConfig.Builder()
.build();
TransferService transferService = new TransferService(cosXmlService, transferConfig);

// 2. 初始化 PutObjectRequest
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String srcPath = "examplefilepath"; //本地文件的绝对路径
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket,
cosPath, srcPath);

// 3. 调用 upload 方法上传文件
final COSUploadTask uploadTask = transferService.upload(putObjectRequest);
uploadTask.setCosXmlResultListener(new CosXmlResultListener() {
```

```
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// 上传成功，可以在这里拿到文件的 CRC64
String crc64 = result.getHeader("x-cos-hash-crc64ecma");
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

下载请求示例

```
// 1. 初始化 TransferService。在相同配置的情况下，您应该复用同一个 TransferService
TransferConfig transferConfig = new TransferConfig.Builder()
.build();
TransferService transferService = new TransferService(cosXmlService, transferConfig);

// 2. 初始化 GetObjectRequest
// 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://console.cloud.tencent.com/cos5/bucket
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; //对象在存储桶中的位置标识符，即称对象键
String savePathDir = context.getCacheDir().toString(); //本地目录路径
//本地保存的文件名，若不填（null），则与 COS 上的文件名一样
String savedFileName = "exampleobject";
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket,
cosPath, savePathDir, savedFileName);

// 3. 调用 download 方法下载文件
final COSDownloadTask downloadTask = transferService.download(getObjectRequest);
downloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// 下载成功，可以在这里拿到 COS 上的文件 CRC64
String cosCRC64 = result.getHeader("x-cos-hash-crc64ecma");
}

// 如果您使用 kotlin 语言来调用，请注意回调方法中的异常是可空的，否则不会回调 onFail 方法，即：
// clientException 的类型为 CosXmlClientException?，serviceException 的类型为 CosXmlServiceException?
@Override
public void onFail(CosXmlRequest request,
@Nullable CosXmlClientException clientException,
```

```
@Nullable CosXmlServiceException serviceException) {
if (clientException != null) {
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

CRC64 校验

通过 TransferService 进行上传和下载时，SDK 默认进行了数据校验的工作，如果您仍然希望能够自己进行 CRC64 校验，可以参考如下代码。

```
// 1. 参考以上上传或者下载请求示例代码获取 COS 上文件的 CRC64 值
String cosCRC64 = "examplecoscrc64";

// 2. 计算本地文件的 CRC64
File localFile = new File("examplefilepath");
String localCRC64 = DigestUtils.getCRC64String(localFile);

// 3. 比对 localCRC64 和 cosCRC64 是否一致
if (localCRC64.equals(cosCRC64)) {
// CRC64 对比正确
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

设置自定义头部

最近更新时间：2022-08-25 09:04:29

简介

本文档主要介绍 SDK 如何在请求时携带自定义头部。

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。

功能说明

COS 在上传对象时可以携带以 x-cos-meta- 开头的自定义头部，包括用户自定义元数据头部后缀和用户自定义元数据信息，这些头部将作为对象元数据保存。

如果您开通了万象服务，可以携带 Pic-Operations 头部，实现后台自动图片处理，详细的 API 说明请参考 数据万象持久化。

示例代码

```
// 存储桶region可以在COS控制台指定存储桶的概览页查看 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
String region = "ap-beijing"; // 您的存储桶地域
String commonHeaderKey = "commonexamplekey"; // 自定义公共 Header 的键
String commonHeaderValue = "commonexamplevalue"; // 自定义公共 Header 的值
String requestHeaderKey = "requestexamplekey"; // 自定义请求 Header 的键
String requestHeaderValue = "requestexamplevalue"; // 自定义请求 Header 的值

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setRegion(region)
.setDebuggable(false)
// 给所有的请求添加公共的自定义 Header
.addHeader(commonHeaderKey, commonHeaderValue)
.builder();

CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig,
credentialProvider);

// 给单个请求添加自定义 Header，优先级比公共 Header 更高
HeadObjectRequest headObjectRequest = new HeadObjectRequest(bucket, cosPath);
try {
headObjectRequest.setRequestHeaders(requestHeaderKey, requestHeaderValue, false);
} catch (CosXmlClientException e) {
e.printStackTrace();
}

// 发起请求
cosXmlService.headObjectAsync(headObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
HeadObjectResult headObjectResult = (HeadObjectResult) result;
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException clientException,
CosXmlServiceException serviceException) {
if (clientException != null) {
```

```
clientException.printStackTrace();
} else {
serviceException.printStackTrace();
}
}
});
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

设置访问域名（CDN/全球加速）

最近更新时间：2022-08-25 09:04:32

简介

本文档提供关于如何使用非默认域名请求 COS 服务。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

CDN 默认加速域名

关于如何开启默认加速域名请参考 开启默认 CDN 加速域名。

以下代码展示了如何使用默认加速域名访问 COS 服务。

示例代码

```
// 存储桶region可以在COS控制台指定存储桶的概览页查看 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
String region = "ap-beijing"; // 您的存储桶地域
// 存储桶的默认加速域名
String cdnDomain = "examplebucket-1250000000.file.myqcloud.com";

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setRegion(region)
.setDebuggable(false)
.setHostFormat(cdnDomain) // 修改请求的域名
.addHeader("Host", cdnDomain) // 修改 header 中的 host 字段
.builder();

// 不提供 credentialProvider 类，下载时可以通过给 url 添加 params 参数，来
// 支持 CDN 权限校验
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

CDN 自定义加速域名

关于如何开启 CDN 自定义加速域名请参考 开启自定义 CDN 加速域名。

以下代码展示了如何使用自定义加速域名访问 COS 服务。

示例代码

```
// 存储桶region可以在COS控制台指定存储桶的概览页查看 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
String region = "ap-beijing"; // 您的存储桶地域
String cdnCustomDomain = "exampledomain.com"; // 自定义加速域名

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setRegion(region)
.setDebuggable(false)
```

版权所有：腾讯云计算（北京）有限责任公司

```
.setHostFormat(cdnCustomDomain) // 修改请求的域名
.addHeader("Host", cdnCustomDomain) // 修改 header 中的 host 字段
.builder();
// 不提供 credentialProvider 类，下载时可以通过给 url 添加 params 参数，来
// 支持 CDN 权限校验
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

自定义源站域名

关于如何设置自定义源站域名请参考 自定义源站域名。

以下代码展示了如何使用自定义源站域名访问 COS 服务。

示例代码

```
// 存储桶region可以在COS控制台指定存储桶的概览页查看 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
String region = "ap-beijing"; // 您的存储桶地域
String customDomain = "exampledomain.com"; // 自定义域名

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setRegion(region)
.setDebuggable(false)
.setHostFormat(customDomain) // 修改请求的域名
.builder();

CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig,
credentialProvider);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

全球加速域名

关于全球加速功能请参考 全球加速功能概述。

以下代码展示了如何使用全球加速域名访问 COS 服务。

示例代码

```
// 存储桶region可以在COS控制台指定存储桶的概览页查看 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
String region = "ap-beijing"; // 您的存储桶地域

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setRegion(region)
.setDebuggable(false)
.setAccelerate(true) // 使能全球加速域名
.builder();
```

```
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig,
credentialProvider);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

异常处理

最近更新时间：2022-08-25 09:04:36

简介

调用 SDK 接口请求 COS 服务失败时，系统将抛出 CosXmlClientException（客户端异常）或者 CosXmlServiceException（服务端异常）。

- CosXmlClientException 是由于客户端无法和 COS 服务端正常进行交互所引起。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。
- CosXmlServiceException 是客户端和 COS 服务端交互正常，但操作 COS 资源失败。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等。

客户端异常

CosClientException 集成自 Exception，使用方法同 Exception，同时添加一个额外的成员 errorCode，如下：

成员	描述	类型
errorCode	客户端错误码，如10000表示参数检验失败，更多详情请参见 SDK 错误码	int

服务端异常

CosXmlServiceException 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。
CosXmlServiceException 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：

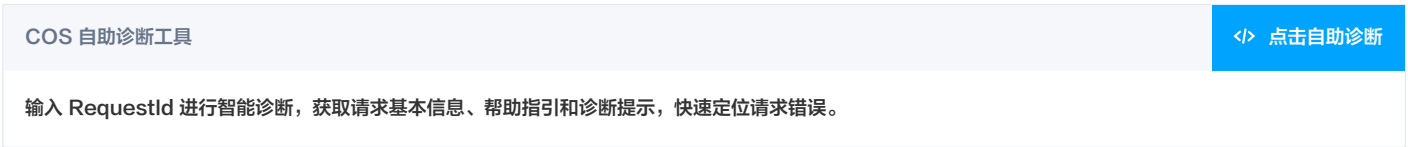
成员	描述	类型
requestId	请求 ID，用于表示一个请求，对于排查问题十分重要	string
statusCode	response 的 status 状态码，4xx 是指请求因客户端而失败，5xx 是服务端异常导致的失败，更多详情请参见 COS 错误信息	string
errorCode	请求失败时 body 返回的 Error Code，更多详情请参见 COS 错误信息	string
errorMessage	请求失败时 body 返回的 Error Message，更多详情请参见 COS 错误信息	string

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 COS 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 COS 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。

升级到 XML Android SDK

最近更新时间：2022-08-25 09:04:41

如果您细心对比过 JSON Android SDK 和 XML Android SDK 的文档，您会发现并不是一个简单的增量更新。XML Android SDK 不仅在架构、可用性和安全性上有了非常大的提升，而且在易用性、健壮性和传输性能上也做了非常大的改进。如果您想要升级到 XML Android SDK，请参考下面的指引，一步步完成 SDK 的升级工作。

功能对比

下表列出了 JSON Android SDK 和 XML Android SDK 的主要功能对比：

功能	XML Android SDK	JSON Android SDK
文件上传	支持本地文件、字节流、输入流上传 默认覆盖上传 智能判断上传模式 简单上传最大支持5GB 分块上传最大支持48.82TB（50,000GB）	只支持本地文件上传 可选择是否覆盖 需要手动选择是简单还是分块上传 简单上传最大支持20MB 分块上传最大支持64GB
文件删除	支持批量删除	只支持单文件删除
存储桶基本操作	创建存储桶 获取存储桶 删除存储桶	不支持
存储桶 ACL 操作	设置存储桶 ACL 获取设置存储桶 ACL 删除设置存储桶 ACL	不支持
存储桶生命周期	创建存储桶生命周期 获取存储桶生命周期 删除存储桶生命周期	不支持
目录操作	不单独提供接口	创建目录 查询目录 删除目录

升级步骤

请按照以下步骤升级 Android SDK。

1. 更新 Android SDK

COS XML Android SDK Android SDK 发布在 MavenCentral 的 maven 包管理平台，推荐您使用自动集成方式进行更新。

在应用的根目录下的 build.gradle 中添加依赖，代码如下：

```
dependencies {
...
// 增加这行
compile 'com.qcloud.cos:cos-android:5.6.+'
}
```

当然，您也可以继续选择手动 jar 包依赖，您可以在这里 COS XML Android SDK-release 下载所有的 jar 包。

2. 更改 SDK 鉴权方式

在 JSON Android SDK 中您需要自己在后台计算好签名，再返回客户端使用。而在 XML SDK 使用了新的鉴权算法，在 XML Android SDK 中，强烈建议您后台接入我们的临时密钥（STS）方案。该方案不需要您了解签名计算过程，只需要在服务器端接入 CAM，将拿到的临时密钥返回到客户端，并设置到 SDK 中，SDK 会负责管理密钥和计算签名。临时密钥在一段时间后会自动失效，而永久密钥不会泄露。
您还可以按照不同的粒度来控制访问权限。具体的步骤请参考 移动应用直传实践 以及 临时密钥生成及使用指引。

如果您仍然采用后台手动计算签名，再返回客户端使用的方式，请注意我们的签名算法发生了改变。签名不再区分单次和多次签名，而是通过设置签名的有效期来保证安全性。请参考 XML 请求签名 文档更新您签名的实现。

3. 更改 SDK 初始化

在 XML Android SDK 中，我们的初始化接口发生了一些变化：

- 为了区分，CosXmlServiceConfig 代替了 COSClientConfig，CosXmlService 代替了 COSClient，但他们的作用相同。
- 您需要在初始化时实例化一个密钥提供者 QCloudCredentialProvider，用于提供一个有效的密钥，建议使用临时密钥。

JSON SDK 的初始化方式如下：

```
//创建COSClientConfig对象，根据需要修改默认的配置参数
COSClientConfig config = new COSClientConfig();
//设置地域
config.setEndPoint(COSEndPoint.COS_GZ);

Context context = getApplicationContext();
String appid = "腾讯云注册的appid";
String peristenceId = "持久化Id";

//创建COSlient对象，实现对象存储的操作
COSClient cos = new COSClient(context,appid,config,peristenceId);
```

XML SDK 的初始化方式如下：

```
String region = "COS_REGION";

CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.setRegion(region)
.isHttps(true) // 使用 HTTPS 请求，默认为 HTTP 请求
.builder();

URL url = null;
try {
// URL 是后台临时密钥服务的地址，如何搭建服务请参考（https://cloud.tencent.com/document/product/436/14048）
url = new URL("https://your_auth_server_url");
} catch (MalformedURLException e) {
e.printStackTrace();
return;
}

/**
* 初始化 {@link QCloudCredentialProvider} 对象，来给 SDK 提供临时密钥
*/
QCloudCredentialProvider credentialProvider = new SessionCredentialProvider(new HttpRequest.Builder<String>()
.url(url)
.method("GET")
.build());

CosXmlService cosXmlService = new CosXmlService(context, serviceConfig, credentialProvider);
```

4. 更改存储桶名称和可用区域简称

XML SDK 的存储桶名称和可用区域简称与 JSON SDK 的不同，需要您进行相应的更改。

存储桶 Bucket

XML Android SDK 存储桶名称由两部分组成：用户自定义字符串 和 APPID，两者以中划线“-”相连。例如 examplebucket-1250000000 ，其中 examplebucket 为用户自定义字符串， 1250000000 为 APPID。

> 说明：
> APPID 是腾讯云账户的账户标识之一，用于关联云资源。在用户成功申请腾讯云账户后，系统自动为用户分配一个 APPID。可通过 账号信息 控制台查看 APPID。

在设置 Bucket 时，请参考下面的示例代码：

```
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject.doc";
String srcPath = Environment.getExternalStorageDirectory().getPath() + "/exampleobject.doc";
//上传文件
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath, srcPath, uploadId);
```

存储桶可用区域简称 Region

XML Android SDK 的存储桶可用区域简称发生了变化，下列表格列出了不同区域在 JSON Android SDK 和 XML Android SDK 中的对应关系：

地域	XML Android SDK 地域简称	JSON Android SDK 地域简称
北京一区（华北）	ap-beijing-1	tj
北京	ap-beijing	bj
上海（华东）	ap-shanghai	sh
广州（华南）	ap-guangzhou	gz
成都（西南）	ap-chengdu	cd
重庆	ap-chongqing	无
香港	ap-hongkong	hk
新加坡	ap-singapore	sgp
多伦多	na-toronto	ca
法兰克福	eu-frankfurt	ger
孟买	ap-mumbai	无
首尔	ap-seoul	无
硅谷	na-siliconvalley	无
弗吉尼亚	na-ashburn	无
曼谷	ap-bangkok	无
莫斯科	eu-moscow	无

在初始化时，请将存储桶所在区域简称设置到 CosXmlServiceConfig 中：

```
String appid = "1250000000";
String region = "ap-guangzhou";

CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.setAppidAndRegion(appid, region)
.builder();
```

5. 更改 API

升级到 XML SDK 之后，一些操作的 API 发生了变化，请您根据实际需求进行相应的更改。我们同时做了封装让 SDK 更加易用，具体请参考我们的示例和 快速入门 文档。

API 变化有以下三点：

（1）没有单独的目录接口

在 XML SDK 中，不再提供单独的目录接口。对象存储中本身是没有文件夹和目录的概念的，对象存储不会因为上传对象 project/a.txt 而创建一个 project 文件夹。为了满足用户使用习惯，对象存储在控制台、COS browser 等图形化工具中模拟了「 文件夹 」或「 目录 」的展示方式，具体实现是通过创建一个键值为 project/ ，内容为空的对象，展示方式上模拟了传统文件夹。

例如：上传对象 project/doc/a.txt ，分隔符 / 会模拟「 文件夹 」的展示方式，于是可以看到控制台上出现「 文件夹 」project 和 doc，其中 doc 是 project 下一级「 文件夹 」，并包含了 a.txt 。

因此，如果您的应用场景只是上传文件，可以直接上传即可，不需要先创建文件夹。

如果您的使用场景里面有文件夹的概念，需要提供创建文件夹的功能，您可以上传一个路径以 '/' 结尾的 0KB 文件。这样在您调用 GetBucket 接口时，就可以将这样的文件当做文件夹。

（2）TransferManager

在 XML SDK 中，我们封装了上传、下载和复制操作，命名为 TransferManager ，同时对 API 设计和传输性能都做了优化，建议您直接使用。TransferManager 的主要特性有：

- 支持上传下载过程的暂停和恢复。
- 支持根据文件大小智能选择简单上传还是分块上传，您可以设置该判断临界。
- 支持任务状态的监听。

使用 TransferManager 上传的示例代码：

```
// 初始化 TransferConfig
TransferConfig transferConfig = new TransferConfig.Builder().build();

/*若有特殊要求，则可以如下进行初始化定制。例如限定当对象 >= 2M 时，启用分块上传，且分块上传的分块大小为1M，当源对象大于5M时启用分块复制，且分块复制的大小为5M。*/
transferConfig = new TransferConfig.Builder()
.setDividsionForCopy(5 * 1024 * 1024) // 是否启用分块复制的最小对象大小
.setSliceSizeForCopy(5 * 1024 * 1024) // 分块复制时的分块大小
.setDivisionForUpload(2 * 1024 * 1024) // 是否启用分块上传的最小对象大小
.setSliceSizeForUpload(1024 * 1024) // 分块上传时的分块大小
.build();

// 初始化 TransferManager
TransferManager transferManager = new TransferManager(cosXmlService, 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 = new File(context.getExternalCacheDir(), "exampleobject").toString(); //本地文件的绝对路径
String uploadId = null; //若存在初始化分块上传的 UploadId，则赋值对应的 uploadId 值用于续传；否则，赋值 null
// 上传对象
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath, srcPath, uploadId);

/**
* 若是上传字节数组，则可调用 TransferManager 的 upload(string, string, byte[]) 方法实现;
* byte[] bytes = "this is a test".getBytes(Charset.forName("UTF-8"));
* cosxmlUploadTask = transferManager.upload(bucket, cosPath, bytes);
*/

/**
* 若是上传字节流，则可调用 TransferManager 的 upload(String, String, InputStream) 方法实现;
* InputStream inputStream = new ByteArrayInputStream("this is a test".getBytes(Charset.forName("UTF-8")));
```

```
* cosxmlUploadTask = transferManager.upload(bucket, cosPath, inputStream);
*/

//设置上传进度回调
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
// todo Do something to update progress...
}
});
//设置返回结果回调
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTask.COSXMLUploadTaskResult cOSXMLUploadTaskResult = (COSXMLUploadTask.COSXMLUploadTaskResult) result;
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException exception, CosXmlServiceException serviceException) {
// todo Upload failed because of CosXmlClientException or CosXmlServiceException...
}
});
//设置任务状态回调, 可以查看任务过程
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
// todo notify transfer state
}
});

/**
若有特殊要求，则可以如下操作：
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setRegion(region); //设置存储桶所在的地域
putObjectRequest.setNeedMD5(true); //是否启用 Md5 校验
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
*/

//取消上传
cosxmlUploadTask.cancel();


//暂停上传
cosxmlUploadTask.pause();

//恢复上传
cosxmlUploadTask.resume();
```

（3）新增 API

XML Android SDK 新增 API，您可根据需求进行调用。包括：

- 存储桶的操作，如 PutBucketRequest、GetBucketRequest、ListBucketRequest 等。
- 存储桶 ACL 的操作，如 PutBucketACLRequest、GetBucketACLRequest 等。
- 存储桶生命周期的操作，如 PutBucketLifecycleRequest、GetBucketLifecycleRequest 等。

具体请参考我们的 Android SDK 快速入门 文档。

C SDK
快速入门

最近更新时间：2023-01-04 20:35:51

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：

```
./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，涉及网络，内存等部分 */
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_cname, curl参数等全局配置信息
* 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，不再需要单独释放内存
*/
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 依据用户所在地域的 COS 服务域名填写
cos_str_set(&options->config->access_key_id, "<用户的SecretId>"); //用户注册 COS 服务后所获得的 SecretId
cos_str_set(&options->config->access_key_secret, "<用户的SecretKey>"); //用户注册 COS 服务后所获得的 SecretKey
cos_str_set(&options->config->appid, "<用户的AppId>"); //用户注册 COS 服务后所获得的 AppId

/* 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_secret 均需要设置为对应临时密钥所配套的 SecretId 和 Secr
```

```
etKey */
//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 Content-MD5 头部，enable 为 COS_TRUE 时上传请求将自动添加Content-MD5 头部，如果不设置此项则默认将添加 Content-MD5 头部 */
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_cname, curl参数等全局配置信息
* 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，不再需要单独释放内存
*/
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_cname, curl参数等全局配置信息
* 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，不再需要单独释放内存
*/
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_cname, curl参数等全局配置信息
* 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，不再需要单独释放内存
*/
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_cname, curl参数等全局配置信息
* 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，不再需要单独释放内存
*/
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_headers);
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_cname, curl 参数等全局配置信息
* 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，不再需要单独释放内存
*/
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：2022-04-22 12:10:56

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/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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\":[{\"fileid\":\"test.png\",\"rule\":\"imageView2/format/png\"}]}");
cos_str_set(&file, "test.jpg");
cos_str_set(&object, "test.jpg");
s = ci_put_object_from_file(options, &bucket, &object, &file, headers, &resp_headers, &results);
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/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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\":[{\"fileid\":\"test.png\",\"rule\":\"imageView2/format/png\"}]}");

s = ci_image_process(options, &bucket, &object, headers, &resp_headers, &results);
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

基础图片处理

最近更新时间：2022-04-22 12:11:00

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。

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/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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, &resp_headers);
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/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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, &resp_headers);
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/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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, &resp_headers);
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：2022-04-22 12:11:03

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://cloud.tencent.com/document/product/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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, &resp_headers);
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, &resp_headers);
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：2022-04-22 12:11:08

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_request,
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型
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

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://cloud.tencent.com/document/product/460/31066 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
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_bucket_list, node) {
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.data);
}
}

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.<Region>.myqcloud.com
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://cloud.tencent.com/document/product/436/48988
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_headers, &media_buckets_result);
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：2022-04-22 12:11:12

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。

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型
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
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
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型
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://cloud.tencent.com/document/product/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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.data);
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.data);
cos_warn_log("stream.video.codec_long_name: %s", result->stream.video.codec_long_name.data);
cos_warn_log("stream.video.codec_time_base: %s", result->stream.video.codec_time_base.data);
cos_warn_log("stream.video.codec_tag_string: %s", result->stream.video.codec_tag_string.data);
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.data);
cos_warn_log("stream.video.field_order: %s", result->stream.video.field_order.data);
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);
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.data);
cos_warn_log("stream.audio.codec_long_name: %s", result->stream.audio.codec_long_name.data);
cos_warn_log("stream.audio.codec_time_base: %s", result->stream.audio.codec_time_base.data);
cos_warn_log("stream.audio.codec_tag_string: %s", result->stream.audio.codec_tag_string.data);
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.data);
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_layout.data);
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);
```

版权所有：腾讯云计算（北京）有限责任公司

```
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.data);
}

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://cloud.tencent.com/document/product/436/55672
s = ci_get_media_info(options, &bucket, &object, NULL, &resp_headers, &media_info);
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：2022-04-22 12:11:16

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

> 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_request,
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

参数名称	参数描述	类型
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

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://cloud.tencent.com/document/product/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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://cloud.tencent.com/document/product/436/55671
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, &download_buffer, &resp_headers);
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_request,
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

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://cloud.tencent.com/document/product/436/6224 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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://cloud.tencent.com/document/product/436/55671
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, &pic_file, &resp_headers);
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;
}
```

版权所有：腾讯云计算（北京）有限责任公司

内容审核
视频审核

最近更新时间：2022-10-14 15:36:47

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 *job_options,
cos_table_t *headers,
cos_table_t **resp_headers,
ci_video_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_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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型
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（审核中）其中一个	String
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://cloud.tencent.com/document/product/460/31066 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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.<Region>.myqcloud.com
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://cloud.tencent.com/document/product/436/47316
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_headers, &job_result);
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

参数名称	参数描述	类型
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
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
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

版权所有：腾讯云计算（北京）有限责任公司

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://cloud.tencent.com/document/product/460/31066 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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->jobs_detail.snapshot_info_list, node) {
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_time);
cos_warn_log("snapshot_info->text: %s", snapshot_info->text.data);

cos_warn_log("snapshot_info->porn_info.hit_flag: %d", snapshot_info->porn_info.hit_flag);
cos_warn_log("snapshot_info->porn_info.score: %d", snapshot_info->porn_info.score);
cos_warn_log("snapshot_info->porn_info.label: %s", snapshot_info->porn_info.label.data);
cos_warn_log("snapshot_info->porn_info.sub_lable: %s", snapshot_info->porn_info.sub_lable.data);
cos_warn_log("snapshot_info->ads_info.hit_flag: %d", snapshot_info->ads_info.hit_flag);
cos_warn_log("snapshot_info->ads_info.score: %d", snapshot_info->ads_info.score);
cos_warn_log("snapshot_info->ads_info.label: %s", snapshot_info->ads_info.label.data);
cos_warn_log("snapshot_info->ads_info.sub_lable: %s", snapshot_info->ads_info.sub_lable.data);
}

index = 0;
cos_list_for_each_entry(ci_auditing_audio_section_result_t, audio_section_info, &result->jobs_detail.audio_section_info_list, node) {
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->offset_time);
cos_warn_log("audio_section_info->duration: %d", audio_section_info->duration);

cos_warn_log("audio_section_info->porn_info.hit_flag: %d", audio_section_info->porn_info.hit_flag);
cos_warn_log("audio_section_info->porn_info.score: %d", audio_section_info->porn_info.score);
cos_warn_log("audio_section_info->porn_info.key_words: %s", audio_section_info->porn_info.key_words.data);
cos_warn_log("audio_section_info->ads_info.hit_flag: %d", audio_section_info->ads_info.hit_flag);
cos_warn_log("audio_section_info->ads_info.score: %d", audio_section_info->ads_info.score);
cos_warn_log("audio_section_info->ads_info.key_words: %s", audio_section_info->ads_info.key_words.data);
}
}

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.<Region>.myqcloud.com
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://cloud.tencent.com/document/product/436/47316
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_headers, &job_result);
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, &resp_headers, &auditing_result);
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：2022-04-22 12:11:27

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/436/6224 文档
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 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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\":[{\"fileid\":\"test.png\",\"rule\":\"QRcode/cover/1\"}]}");
// 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_headers, &results);
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, 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);
}

//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/436/6224 文档
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 文档
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.cloud.tencent.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.cloud.tencent.com/cam/capi 页面获取
static char TEST_APPID[] = "<APPID>"; //your appid
//the cos bucket name, syntax: [bucket]-[appid], for example: mybucket-1253666666，可在 https://console.cloud.tencent.com/cos5/bucket 查看
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, &result2);
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：2023-01-04 20:36:07

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，如果客户编译环境的gcc版本不同，需要重新编译libcossdk.a
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版本编译的，如果客户编译环境的openssl版本不同，需要重新编译poco
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'
```

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<char, std::char_traits<char>, std::allocator<char> > const&)
```

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，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
"SecretKey":"********************************", // sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
"Region":"ap-guangzhou", // 存储桶地域，替换为客户存储桶所在地域，可以在COS控制台指定存储桶的概览页查看存储桶地域信息，参考 https://console.cloud.tencent.com/cos5/bucket/ ，关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
```

```
"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https://cloud.tencent.com/document/product/436/14048
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bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https://console.cloud.tencent.com/cos5/bucket
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_display_name << std::endl;

for (std::vector<qcloud_cos::Bucket>::const_iterator itr = buckets.begin(); itr != buckets.end(); ++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::endl;
}
```

```
// 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bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https://console.cloud.tencent.com/cos5/bucket
std::string object_name = "exampleobject"; //exampleobject 即为对象键（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，替换为用户指定的对象名。
qcloud_cos::PutObjectByFileReq req(bucket_name, object_name, "/path/to/local/file"); // 替换为用户指定的文件路径
//req.SetXCosStorageClass("STANDARD_IA"); // 默认为STANDARD，可以调用 Set 方法设置存储类型
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bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https://console.cloud.tencent.com/cos5/bucket
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(); itr != cotents.end(); ++itr) {
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bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https://console.cloud.tencent.com/cos5/bucket
std::string object_name = "exampleobject"; // exampleobject 即为对象键（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，替换为用户指定的对象名。
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bucketname-appid 组成，appid必须填入，可以在COS控制台查看存储桶名称。 https://console.cloud.tencent.com/cos5/bucket
std::string object_name = "exampleobject"; // exampleobject 即为对象键（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，替换为用户指定的对象名。。
// 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：2022-04-22 12:11:58

简介

版权所有：腾讯云计算（北京）有限责任公司

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::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	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，如果不指定的话默认保存到当前存储桶
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，存入原图文件存储的同目录
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，后加样式名，如样式名为test，则 rule 字段为style/test
};

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），不填则不进行图片处理
};
```

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表示疑似有盲水印，60以下可认为未提取到盲水印
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, CloudImageProcessResp* response);
```

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：2022-04-22 12:12:01

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* response);
```

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：2022-04-22 12:12:08

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/" + object_name + "_watermark";
// 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* response);
```

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(watermark_text);
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：2022-04-22 12:12:13

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, DescribeMediaBucketsResp* response);
```

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

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：2022-04-22 12:12:17

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

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：2022-04-22 12:12:20

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

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：2022-10-14 15:37:11

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。

API	操作描述
图片单次审核	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	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。

方法原型

```
CosResult GetImageAuditing(const GetImageAuditingReq& req, GetImageAuditingResp* resp);
```

请求示例

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

GetImageAuditingReq req(bucket_name);
GetImageAuditingResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetObjectKey(object_name);
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.GetImageAuditing(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 GetImageAuditingResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	GetImageAuditing 操作的请求	GetImageAuditingReq	是
resp	GetImageAuditing 操作的响应	GetImageAuditingResp	是

GetImageAuditingReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// COS 存储桶中的图片文件名称，需要审核图片文件对象
void SetObjectKey(const std::string& object_name);
// 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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 审核图片Url，可为任意公网可访问图片链接
// 设置了detect-url时，默认审核detect-url，无需填写ObjectKey
// 不设置detect-url时，默认审核ObjectKey
void SetDetectUrl(const std::string& detect_url);
// 审核GIF动图时，使用该参数截帧审核，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
void SetInterval(int interval);
// 审核动图时最大截帧数量，默认为5
void SetMaxFrames(int max_frames);
// 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。
void SetLargeImageDetect(int large_image_detect);
// 图片标识，该字段返回原始内容，长度限制为512字节
void SetDataId(const std::string& data_id);
```

GetImageAuditingResp 提供以下成员函数：

```
// 获取API请求执行返回的任务详情
ImageAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 ImageAuditingJobsDetail 结构中，具体返回参数可查看 图片单次审核 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

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

```
CosResult CosAPI::BatchImageAuditing(const BatchImageAuditingReq& req, BatchImageAuditingResp* resp);
```

请求示例

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";
std::string object_name_a = "test1.jpg";
std::string object_name_b = "test2.jpg";

BatchImageAuditingReq req(bucket_name);
BatchImageAuditingResp resp;

// input 1
AuditingInput input_a = AuditingInput();
input_a.SetObject(object_name_a);
req.AddInput(input_a);

// input2
AuditingInput input_b = AuditingInput();
input_a.SetObject(object_name_b);
req.AddInput(input_b);

// 审核配置
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.BatchImageAuditing(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 BatchImageAuditingResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	BatchImageAuditing 操作的请求	BatchImageAuditingReq	是
resp	BatchImageAuditing 操作的响应	BatchImageAuditingResp	是
input	BatchImageAuditing 操作请求的Input参数	AuditingInput	是

BatchImageAuditingReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置审核配置
void SetConf(const Conf& 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，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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 添加单个图片审核的Input
void AddInput(const AuditingInput& input);
```

```
// 设置需要批量审核的Input数组
void SetInputs(const std::vector<AuditingInput>& inputs);
```

AuditingInput 提供以下成员函数：

```
// COS 存储桶中的图片文件名称，需要审核图片文件对象
void SetObject(const std::string& object);
// 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。
void SetUrl(const std::string& url);
// 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
void SetInterval(const int interval);
// 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
void SetMaxFrames(const int max_frames);
// 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
void SetDataId(const std::string& data_id);
// 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。
void SetLargeImageDetect(const int large_image_detect);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
```

BatchImageAuditingResp 提供以下成员函数：

```
// 获取API请求执行返回的任务详情
std::vector<ImageAuditingJobsDetail> GetJobsDetails();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 ImageAuditingJobsDetail 结构数组中，具体返回参数可查看 图片批量审核 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

查询图片审核任务结果

功能说明

用于提交一个图片审核任务。

方法原型

```
CosResult DescribeImageAuditingJob(const DescribeImageAuditingJobReq &req, DescribeImageAuditingJobResp *resp);
```

请求示例

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

DescribeImageAuditingJobReq req(bucket_name);
DescribeImageAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("vab1ca9fc8a3ed11ea834c525400863904");

// 调用接口,获取任务响应对象
```

```
CosResult result = cos.DescribeImageAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 DescribeImageAuditingJobResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeImageAuditingJob 操作的请求	DescribeImageAuditingJobReq	是
resp	DescribeImageAuditingJob 操作的响应	DescribeImageAuditingJobResp	是

DescribeVideoAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeVideoAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
ImageAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 ImageAuditingJobsDetail 结构中，具体返回参数可查看 查询图片审核任务结果 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

视频审核

最近更新时间：2022-10-14 15:37:31

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

```
CosResult CreateVideoAuditingJob(const CreateVideoAuditingJobReq& req, CreateVideoAuditingJobResp* resp);
```

请求示例

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";
std::string object_name = "test.mp4";

CreateVideoAuditingJobReq req(bucket_name);
CreateVideoAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetObject(object_name);
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");
SnapShotConf snap_shot = SnapShotConf();
snap_shot.SetCount(100);
snap_shot.SetMode("Interval");
snap_shot.SetTimeInterval(10);
req.SetSnapShot(snap_shot);

// 调用接口,获取任务响应对象
CosResult result = cos.CreateVideoAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 CreateVideoAuditingJobResp 的成员函数
} else {
```

```
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	CreateVideoAuditing 操作的请求	CreateVideoAuditingReq	是
resp	CreateVideoAuditing 操作的响应	CreateVideoAuditingResp	是

CreateVideoAuditingReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置input需要审核的视频
void SetInput(const AuditingInput& input);
// 设置审核配置规则
void SetConf(const Conf& conf);

// 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，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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 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。
void SetSnapShot(const SnapShotConf& snap_shot);
// 回调地址，以http://或者https://开头的地址。
void SetCallBack(const std::string& callback);
// 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。
void SetCallBackVersion(const std::string& callback_version);
// 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。
void SetDetectContent(const int detect_content);

// 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。
void SetObject(const std::string& object);
// 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。
void SetUrl(const std::string& url);
// 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。
void SetDataId(const std::string& data_id);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
```

CreateVideoAuditingResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
VideoAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 VideoAuditingJobsDetail 结构中，具体返回参数可查看 提交视频审核任务 文档。

- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。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

```
CosResult DescribeVideoAuditingJob(const DescribeVideoAuditingJobReq& req, DescribeVideoAuditingJobResp* resp);
```

请求示例

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

DescribeVideoAuditingJobReq req(bucket_name);
DescribeVideoAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("vab1ca9fc8a3ed11ea834c525400863904");

// 调用接口,获取任务响应对象
CosResult result = cos.DescribeVideoAuditingJob(req, &resp);
if (result.IsSucc()) {
// 查询审核任务成功，可以调用 DescribeVideoAuditingJobResp 的成员函数
} else {
// 查询审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeVideoAuditingJob 操作的请求	DescribeVideoAuditingJobReq	是
resp	DescribeVideoAuditingJob 操作的响应	DescribeVideoAuditingJobResp	是

DescribeVideoAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeVideoAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
VideoAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 VideoAuditingJobsDetail 结构中，具体返回参数可查看 提交视频审核任务 文档。

- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

音频审核

最近更新时间：2022-10-14 15:37:39

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

```
CosResult CosAPI::CreateAudioAuditingJob(const CreateAudioAuditingJobReq& req, CreateAudioAuditingJobResp* resp);
```

请求示例

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "examplebucket-1250000000";
std::string object_name = "test.mp3";

CreateAudioAuditingJobReq req(bucket_name);
CreateAudioAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetObject(object_name);
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.CreateAudioAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 CreateAudioAuditingJobResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	CreateAudioAuditing 操作的请求	CreateAudioAuditingJobReq	是
resp	CreateAudioAuditing 操作的响应	CreateAudioAuditingJobResp	是

CreateAudioAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置input需要审核的视频
void SetInput(const AuditingInput& input);
// 设置审核配置规则
void SetConf(const Conf& conf);

// 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，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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 回调地址，以http://或者https://开头的地址。
void SetCallBack(const std::string& callback);
// 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。
void SetCallBackVersion(const std::string& callback_version);


// 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。
void SetObject(const std::string& object);
// 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。
void SetUrl(const std::string& url);
// 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。
void SetDataId(const std::string& data_id);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
```

CreateAudioAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
AudioAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 AudioAuditingJobsDetail 结构中，具体返回参数可查看 提交音频审核任务 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

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。

方法原型

```
CosResult DescribeAudioAuditingJob(const DescribeAudioAuditingJobReq& req, DescribeAudioAuditingJobResp* resp);
```

请求示例

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

DescribeAudioAuditingJobReq req(bucket_name);
DescribeAudioAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("aab1ca9fc8a3ed11ea834c525400863904");

// 调用接口,获取任务响应对象
CosResult result = cos.DescribeAudioAuditingJob(req, &resp);
if (result.IsSucc()) {
// 查询审核任务成功，可以调用 DescribeAudioAuditingJobResp 的成员函数
} else {
// 查询审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeAudioAuditingJob 操作的请求	DescribeAudioAuditingJobReq	是
resp	DescribeAudioAuditingJob 操作的响应	DescribeAudioAuditingJobResp	是

DescribeAudioAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeAudioAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
AudioAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 AudioAuditingJobsDetail 结构中，具体返回参数可查看 查询音频审核任务结果 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

文本审核

最近更新时间：2022-10-14 15:37:47

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

```
CosResult CosAPI::CreateTextAuditingJob(const CreateTextAuditingJobReq& req, CreateTextAuditingJobResp* resp);
```

请求示例

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

CreateTextAuditingJobReq req(bucket_name);
CreateTextAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetObject(object_name);
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.CreateTextAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 CreateTextAuditingJobResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填

参数	参数描述	类型	是否必填
req	CreateTextAuditing 操作的请求	CreateTextAuditingJobReq	是
resp	CreateTextAuditing 操作的响应	CreateTextAuditingJobResp	是

CreateTextAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置input需要审核的视频
void SetInput(const AuditingInput& input);
// 设置审核配置规则
void SetConf(const Conf& conf);

// 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，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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 回调地址，以http://或者https://开头的地址。
void SetCallBack(const std::string& callback);
// 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。
void SetCallBackVersion(const std::string& callback_version);


// 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 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。
void SetObject(const std::string& object);
// 文本文件的完整链接，例如：https://www.test.com/test.txt。
void SetUrl(const std::string& url);
// 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。
void SetDataId(const std::string& data_id);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
// 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。
void SetContent(const std::string& content) { m_input.SetContent(content); }
```

CreateTextAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
TextAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 TextAuditingJobsDetail 结构中，具体返回参数可查看 提交文本审核任务 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

查询文本审核任务结果

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

```
CosResult DescribeTextAuditingJob(const DescribeTextAuditingJobReq& req, DescribeTextAuditingJobResp* resp);
```

请求示例

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

DescribeTextAuditingJobReq req(bucket_name);
DescribeTextAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("aab1ca9fc8a3ed11ea834c525400863904");

// 调用接口,获取任务响应对象
CosResult result = cos.DescribeTextAuditingJob(req, &resp);
if (result.IsSucc()) {
// 查询审核任务成功，可以调用 DescribeTextAuditingJobResp 的成员函数
} else {
// 查询审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeTextAuditingJob 操作的请求	DescribeTextAuditingJobReq	是
resp	DescribeTextAuditingJob 操作的响应	DescribeTextAuditingJobResp	是

DescribeTextAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeTextAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
TextAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 TextAuditingJobsDetail 结构中，具体返回参数可查看 查询文本审核任务结果 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

文档审核

最近更新时间：2022-10-14 15:37:56

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

```
CosResult CosAPI::CreateDocumentAuditingJob(const CreateDocumentAuditingJobReq& req, CreateDocumentAuditingJobResp* resp);
```

请求示例

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

CreateDocumentAuditingJobReq req(bucket_name);
CreateDocumentAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetObject(object_name);
req.SetType("docx");
req.SetBizType("b81d45f94b91a683255e9a9506f45a11");
// req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.CreateDocumentAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 CreateDocumentAuditingJobResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	CreateDocumentAuditing 操作的请求	CreateDocumentAuditingJobReq	是
resp	CreateDocumentAuditing 操作的响应	CreateDocumentAuditingJobResp	是

CreateDocumentAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置input需要审核的视频
void SetInput(const AuditingInput& input);
// 设置审核配置规则
void SetConf(const Conf& conf);

// 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，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。
void SetBizType(const std::string& biz_type);
// 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。
void SetDetectType(const std::string& detect_type);
// 回调地址，以http://或者https://开头的地址。
void SetCallBack(const std::string& callback);



// input
// 存储在 COS 存储桶中的文件名称，例如在目录 test 中的文件test.doc，则文件名称为 test/test.doc。Object 和 Url 只能选择其中一种。
void SetObject(const std::string& object);
// 文档文件的链接地址，例如 http://www.example.com/doctest.doc，Object 和 Url 只能选择其中一种。
void SetUrl(const std::string& url);
// 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。
void SetDataId(const std::string& data_id);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
// 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。
void SetType(const std::string& type);
```

CreateDocumentAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
DocumentAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 DocumentAuditingJobsDetail 结构中，具体返回参数可查看 提交文档审核任务 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

查询文档审核任务结果

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

```
CosResult DescribeDocumentAuditingJob(const DescribeDocumentAuditingJobReq& req, DescribeDocumentAuditingJobResp* resp);
```

请求示例

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

DescribeDocumentAuditingJobReq req(bucket_name);
DescribeDocumentAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("stb1ca9fc8a34c525400863904****");

// 调用接口,获取任务响应对象
CosResult result = cos.DescribeDocumentAuditingJob(req, &resp);
if (result.IsSucc()) {
// 查询审核任务成功，可以调用 DescribeDocumentAuditingJobResp 的成员函数
} else {
// 查询审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeDocumentAuditingJob 操作的请求	DescribeDocumentAuditingJobReq	是
resp	DescribeDocumentAuditingJob 操作的响应	DescribeDocumentAuditingJobResp	是

DescribeDocumentAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeDocumentAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
DocumentAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 DocumentAuditingJobsDetail 结构中，具体返回参数可查看 查询文档审核任务结果 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

网页审核

最近更新时间：2022-10-14 15:38:04

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

提交网页审核任务

功能说明

用于提交一个网页审核任务。

方法原型

```
CosResult CosAPI::CreateWebPageAuditingJob(const CreateWebPageAuditingJobReq& req, CreateWebPageAuditingJobResp* resp);
```

请求示例

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

CreateWebPageAuditingJobReq req(bucket_name);
CreateWebPageAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetUrl(url);
req.SetDetectType("Porn,Ads");

// 调用接口,获取任务响应对象
CosResult result = cos.CreateWebPageAuditingJob(req, &resp);
if (result.IsSucc()) {
// 创建审核任务成功，可以调用 CreateWebPageAuditingJobResp 的成员函数
} else {
// 创建审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填

参数	参数描述	类型	是否必填
req	CreateWebPageAuditing 操作的请求	CreateWebPageAuditingJobReq	是
resp	CreateWebPageAuditing 操作的响应	CreateWebPageAuditingJobResp	是

CreateWebPageAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置input需要审核的视频
void SetInput(const AuditingInput& input);
// 设置审核配置规则
void SetConf(const Conf& conf);

// 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。
void SetDetectType(const std::string& detect_type);
// 回调地址，以http://或者https://开头的地址。
void SetCallBack(const std::string& callback);
// 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。
void SetReturnHighligthHtml(const bool return_highlight_html);


// input
// 网页文件的链接地址，例如 http://www.test.com。
void SetUrl(const std::string& url);
// 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。
void SetDataId(const std::string& data_id);
// 用户业务字段。
void SetUserInfo(const UserInfo& user_info);
```

CreateWebPageAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
WebPageAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 WebPageAuditingJobsDetail 结构中，具体返回参数可查看 提交网页审核任务 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

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。

方法原型

```
CosResult DescribeWebPageAuditingJob(const DescribeWebPageAuditingJobReq& req, DescribeWebPageAuditingJobResp* resp);
```

请求示例

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

DescribeWebPageAuditingJobReq req(bucket_name);
DescribeWebPageAuditingJobResp resp;

// 添加请求参数 参数详情请见api接口文档
req.SetJobId("aab1ca9fc8a3ed11ea834c525400863904");

// 调用接口,获取任务响应对象
CosResult result = cos.DescribeWebPageAuditingJob(req, &resp);
if (result.IsSucc()) {
// 查询审核任务成功，可以调用 DescribeWebPageAuditingJobResp 的成员函数
} else {
// 查询审核任务失败，可以调用 CosResult 的成员函数输出错误信息
}
```

参数说明

参数	参数描述	类型	是否必填
req	DescribeWebPageAuditingJob 操作的请求	DescribeWebPageAuditingJobReq	是
resp	DescribeWebPageAuditingJob 操作的响应	DescribeWebPageAuditingJobResp	是

DescribeWebPageAuditingJobReq 提供以下成员函数：

```
// 设置执行操作的bucket
void SetBucketName(const std::string& bucket_name);
// 设置查询的审核任务ID
void SetJobId(const std::string& job_id);
```

DescribeWebPageAuditingJobResp 提供的成员函数如下：

```
// 获取API请求执行返回的任务详情
WebPageAuditingJobsDetail GetJobsDetail();
// 获取API请求的透传ID
std::string GetRequestId();
```

返回结果说明

- 成功：解析 API 返回的 XML 内容中的审核任务结果到 WebPageAuditingJobsDetail 结构中，具体返回参数可查看 查询网页审核任务结果 文档。
- 失败：发生错误（例如 Bucket 不存在），错误信息则解析在 CosResult 结构体中。详情请参见 异常处理。

.NET(C#) SDK 快速入门

最近更新时间：2023-01-04 20:37:15

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 接口与参数文档请参见 SDK API。
- 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：永久密钥、持续更新的临时密钥、不变的临时密钥。

方式1：永久密钥

```
string secretId = "SECRET_ID"; //"云 API 密钥 SecretId";
string secretKey = "SECRET_KEY"; //"云 API 密钥 SecretKey";
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider cosCredentialProvider = new DefaultQCloudCredentialProvider(
secretId, secretKey, durationSecond);
```

方式2：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 : DefaultSessionQCloudCredentialProvider
{

// 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";
string tmpSecretKey = "SECRET_KEY"; //"临时密钥 SecretKey";
string tmpToken = "COS_TOKEN"; //"临时密钥 token";
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3：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 DefaultSessionQCloudCredentialProvider(
tmpSecretId, tmpSecretKey, tmpExpireTime, tmpToken);
```

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 / total));
};

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 / total));
};

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：2022-04-22 12:12:34

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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

```
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;
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, localFileName);
//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/product/436/44879
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：2022-04-22 12:12:38

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。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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, localFileName);
//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/<text>", null);

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

> 说明：
> 更多完整示例，请前往 GitHub 查看。

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：2022-04-22 12:12:43

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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://cloud.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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：2022-04-22 12:12:46

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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://cloud.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
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.CodecLongName);
// 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();
// .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：2022-04-22 12:12:50

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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://cloud.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
```

```
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，例如"/usr/local/"
GetSnapshotRequest request = new GetSnapshotRequest(bucket, key, time, destPath);
// 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：2022-10-14 15:39:52

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
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.SensitiveContentRecognition(request);
}

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：2022-10-14 15:40:04

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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.tencent.com/document/product/436/6224
.Build();
```

```
string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
SubmitVideoCensorJobRequest request = new SubmitVideoCensorJobRequest(bucket);
request.SetCensorObject("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
// 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
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，该参数用于指定视频截帧的配置。
request.SetSnapshotMode("Average"); // 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。
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，支持 float 格式，执行精度精确到毫秒
// 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.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
GetVideoCensorJobRequest request = new GetVideoCensorJobRequest(bucket, JobId);
// 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].SnapshotTime);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.Score);
```

```
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.Label);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].PornInfo.SubLabel);

Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.Score);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.Label);
Console.WriteLine(result.resultStruct.JobsDetail.Snapshot[i].AdsInfo.SubLabel);
}
}

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：2022-10-14 15:40:13

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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.tencent.com/document/product/436/6224
.Build();
```

```
string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
SubmitAudioCensorJobRequest request = new SubmitAudioCensorJobRequest(bucket);
request.SetCensorObject("audio.mp3"); // 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
// 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
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.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
GetAudioCensorJobRequest request = new GetAudioCensorJobRequest(bucket, JobId);
// 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.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Score);
```

```
Console.WriteLine(result.resultStruct.JobsDetail.Section[i].AdsInfo.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.Section[i].AdsInfo.Score);
}
}

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：2022-10-14 15:40:20

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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.tencent.com/document/product/436/6224
.Build();
```

```
string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
SubmitDocumentCensorJobRequest request = new SubmitDocumentCensorJobRequest(bucket);
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，不同类型以逗号分隔，例如：Porn,Ads
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.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
GetDocumentCensorJobRequest request = new GetDocumentCensorJobRequest(bucket, JobId);
// 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.PageNumber);
Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornInfo.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornInfo.SubLabel);
Console.WriteLine(result.resultStruct.JobsDetail.PageSegment.Results.PornInfo.Score);
}

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：2022-10-14 15:40:28

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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.tencent.com/document/product/436/6224
.Build();
```

```
string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
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文件的对象键
// 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
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.tencent.com/document/product/436/6224
.Build();

string secretId = "SECRET_ID"; // 云 API 密钥 SecretId, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
string secretKey = "SECRET_KEY"; // 云 API 密钥 SecretKey, 获取 API 密钥请参照 https://console.cloud.tencent.com/cam/capi
long durationSecond = 600; //每次请求签名有效时长，单位为秒
QCloudCredentialProvider qCloudCredentialProvider = new DefaultQCloudCredentialProvider(secretId,
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://console.cloud.tencent.com/developer
string bucket = "examplebucket-1250000000"; // 注意：此操作需要 bucket 开通内容审核相关功能
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.HitFlag);
Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Score);
Console.WriteLine(result.resultStruct.JobsDetail.Section[i].PornInfo.Keywords);
}
}

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：2022-04-22 12:13:16

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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表示开启二维码覆盖，默认值0
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：2023-01-04 20:37:47

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 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://examplebucket-1250000000.cos.<Region>.myqcloud.com
BucketURL *url.URL
// 访问 service API 的基础 URL（不包含 path 部分）: https://cos.<Region>.myqcloud.com
ServiceURL *url.URL
// 访问 Batch API 的基础 URL （不包含 path 部分）: https://<UIN>.cos-control.<Region>.myqcloud.com
BatchURL *url.URL
```

```
// 访问 CI 的基础 URL（不包含 path 部分）: https://examplebucket-1250000000.ci.<Region>.myqcloud.com
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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})
```

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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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://cloud.tencent.com/document/product/436/14048
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://cloud.tencent.com/document/product/436/14048
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://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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://cloud.tencent.com/document/product/436/14048
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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})

_, 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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})

s, _, err := c.Service.Get(context.Background())
if err != nil {
panic(err)
}
```

版权所有：腾讯云计算（北京）有限责任公司

```
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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})
// 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/test/objectPut.go` 中，对象键为 test/objectPut.go
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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})


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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})
// 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。https://console.cloud.tencent.com/cos5/bucket
// COS_REGION 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket, 关于地域的详情见 https://cloud.tencent.com/document/product/436/6224
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: "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://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
},
})
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：2022-04-22 12:13:32

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	否
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)
```

基础图片处理

最近更新时间：2022-08-10 09:54:51

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
亮度	对图片进行亮度调节
对比度	对图片进行对比度调节
锐化	对图片进行锐化
灰度图	将图片设置为灰度图
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
限制图片大小	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

基础图片处理

功能说明

支持灵活的图像编辑，例如图片旋转、裁剪、转码、缩放等。

下载时处理

方法原型

```
func (s *CIService) Get(ctx context.Context, key string, operation string, opt *ObjectGetOptions, id ...string) (*Response, error)

func (s *CIService) GetToFile(ctx context.Context, key, localpath, operation string, opt *ObjectGetOptions, id ...string) (*Response, error)
```

请求示例 (以缩放为例)

```
key := "test.jpg"
operation : = "imageMogr2/thumbnail/!50px"
```

```
// Case 1 从响应体中获取对象
resp, err := c.CI.Get(context.Background(), key, operation, 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(), key, filepath, operation, nil)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
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。
opt	对象下载参数，详见 下载对象
id	对象 VersionId

上传时处理

方法原型

```
func (s *CIService) PutFromFile(ctx context.Context, key string, filePath string, opt *ObjectPutOptions) (*ImageProcessResult, *Response, error)
```

请求示例（以缩放为例）

```
obj := "pic/upload/deer.jpg"
localpath := "./deer.jpg"
operation ： = "imageMogr2/thumbnail/!50px"
pic := &cos.PicOperations{
IsPicInfo: 1,
Rules: []cos.PicOperationsRules{
{
FileId: "thumbnail_50.jpg",
Rule: operation,
},
},
}
opt := &cos.ObjectPutOptions{
nil,
&cos.ObjectPutHeaderOptions{
XOptionHeader: &http.Header{},
},
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
res, _, err := c.CI.PutFromFile(context.Background(), obj, localpath, opt)
```

参数说明

参数名称	参数描述

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
filePath	本地文件的路径
opt	对象上传参数，详见 上传对象

云上数据处理

方法原型

```
func (s *CIService) ImageProcess(ctx context.Context, key string, opt *ImageProcessOptions) (*ImageProcessResult, *Response, error)
```

请求示例（以缩放为例）

```
obj := "pic/upload/deer.jpg"
operation ： = "imageMogr2/thumbnail/!50px"
pic := &cos.PicOperations{
IsPicInfo: 1,
Rules: []cos.PicOperationsRules{
{
FileId: "thumbnail_50.jpg",
Rule: operation,
},
},
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
res, _, err := c.CI.ImageProcess(context.Background(), obj, pic)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
opt	图片处理参数

图片样式

最近更新时间：2022-08-10 09:54:58

简介

本文档提供关于图片样式的 API 概览以及 SDK 示例代码。

API	操作描述
增加样式	该接口用于对某一个存储桶设置样式功能，设置样式后，可以方便地对图片进行统一的处理。
查询样式	用于查询该 Bucket 下已有的样式。若不使用请求体，则查询该 Bucket 下的全部样式，若使用请求体，则查询请求体中携带的样式名信息。
删除样式	用于删除某一特定样式

增加样式

功能说明

该接口用于对某一个存储桶设置样式功能，设置样式后，可以方便地对图片进行统一的处理。

方法原型

```
func (s *CIService) AddStyle(ctx context.Context, opt *AddStyleOptions) (*Response, error)
```

请求示例

```
style := &cos.AddStyleOptions{
StyleName: "grayscale_1",
StyleBody: "imageMogr2/grayscale/1",
}
_, err := c.CI.AddStyle(context.Background(), style)
```

参数说明

```
type AddStyleOptions struct {
XMLName xml.Name `xml:"AddStyle"`
StyleName string `xml:"StyleName,omitempty"`
StyleBody string `xml:"StyleBody,omitempty"`
}
```

参数名称	参数描述	类型	是否必填
StyleName	样式名称	string	是
StyleBody	样式详情	string	是

查询样式

功能说明

用于查询该 Bucket 下已有的样式。若不使用请求体，则查询该 Bucket 下的全部样式，若使用请求体，则查询请求体中携带的样式名信息。

方法原型

```
func (s *CIService) GetStyle(ctx context.Context, opt *GetStyleOptions) (*GetStyleResult, *Response, error)
```

请求示例

```
// 查询某个样式
opt := &cos.GetStyleOptions{StyleName: "rotate_90"}
res, _, err := c.CI.GetStyle(context.Background(), opt)
// 查询所有样式
res, _, err = c.CI.GetStyle(context.Background(), nil)
```

参数说明

```
type GetStyleOptions struct {
XMLName xml.Name `xml:"GetStyle"`
StyleName string `xml:"StyleName,omitempty"`
}

type GetStyleResult struct {
XMLName xml.Name `xml:"StyleList"`
StyleRule []StyleRule `xml:"StyleRule,omitempty"`
}

type StyleRule struct {
StyleName string `xml:"StyleName,omitempty"`
StyleBody string `xml:"StyleBody,omitempty"`
}
```

删除样式

功能说明

用于删除某一特定样式。

方法原型

```
func (s *CIService) DeleteStyle(ctx context.Context, opt *DeleteStyleOptions) (*Response, error)
```

请求示例

```
style := &cos.DeleteStyleOptions{
StyleName: "grayscale_1",
}
_, err := c.CI.DeleteStyle(context.Background(), style)
```

参数说明

```
type DeleteStyleOptions struct {
XMLName xml.Name `xml:"DeleteStyle"`
StyleName string `xml:"StyleName,omitempty"`
}
```

参数名称	参数描述	类型	是否必填
StyleName	样式名称	string	是

图片高级压缩

最近更新时间：2022-08-10 09:55:05

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 WebP、 AVIF 、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

高效地将图片转换为 WebP、 AVIF 、TPG 或 HEIF 等高压缩比格式

方法原型

```
func (s *CIService) GetToFile(ctx context.Context, key, localpath, operation string, opt *ObjectGetOptions, id ...string) (*Response, error)
```

请求示例

```
// 将原图转换为 WebP 格式
name := "test.png"
filepath := "test.webp"
_, err := c.CI.GetToFile(context.Background(), key, filepath, "imageMogr2/format/webp", nil)
if err != nil {
// ERROR
}

// 将原图转换为 AVIF 格式
filepath = "test.avif"
_, err = c.CI.GetToFile(context.Background(), key, filepath, "imageMogr2/format/avif", nil)
if err != nil {
// ERROR
}

// 将原图转换为 TPG 格式
name := "test.png"
filepath := "test.tpg"
_, err := c.CI.GetToFile(context.Background(), key, filepath, "imageMogr2/format/tpg", nil)
if err != nil {
// ERROR
}

// 将原图转换为 HEIF 格式
filepath = "test.heif"
_, err = c.CI.GetToFile(context.Background(), key, filepath, "imageMogr2/format/heif", nil)
if err != nil {
// ERROR
}
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
localpath	图片保存路径
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。
opt	对象下载参数，详见 下载对象
id	对象 VersionId

Guetzli 压缩

最近更新时间：2022-08-10 09:55:10

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, 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}
```

版权所有：腾讯云计算（北京）有限责任公司

```
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

功能说明

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：2022-04-22 12:13:48

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([]byte("testwatermark")),
},
},
}
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/text/"+base64.StdEncoding.EncodeToString([]byte("testwatermark")), nil)
```

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([]byte("testwatermark")),
},
},
}
opt.XOptionHeader.Add("Pic-Operations", cos.EncodePicOperations(pic))
name := "test2.jpg"
_, err := c.Object.PutFromFile(context.Background(), name, filepath, opt)
```

版权所有：腾讯云计算（北京）有限责任公司

数据万象服务

最近更新时间：2022-08-10 14:42:05

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, error)
```

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)
```

版权所有：腾讯云计算（北京）有限责任公司

请求示例

```
_, err := c.CI.CloseCIService(context.Background())
```

原图保护

最近更新时间：2022-08-10 14:42:10

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, *Response, error)
```

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())
```

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：2022-04-22 12:13:53

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 *DescribeMediaProcessBucketsOptions) (*DescribeMediaProcessBucketsResult, *Response, error)
```

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：2022-04-22 12:13:57

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 *ObjectGetOptions, id ...string) (*GetMediaInfoResult, *Response, error)
```

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

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

节点名称（关键字）	父节点	描述	类型
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
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

版权所有：腾讯云计算（北京）有限责任公司

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

媒体截图接口

最近更新时间：2022-04-22 12:14:02

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 *GetSnapshotOptions, id ...string) (*Response, error)
```

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

参数名称	参数描述	是否必填	类型
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：2022-04-22 12:14:08

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 *DescribeDocProcessBucketsOptions) (*DescribeDocProcessBucketsResult, *Response, error)
```

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：2022-04-22 12:14:12

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 *DocPreviewOptions) (*Response, error)
```

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
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

版权所有：腾讯云计算（北京）有限责任公司

异步处理队列接口

最近更新时间：2022-09-05 16:56:42

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 *DescribeDocProcessQueuesOptions) (*DescribeDocProcessQueuesResult, *Response, error)
```

请求示例

```
DescribeQueueOpt := &cos.DescribeDocProcessQueuesOptions{
QueueIds: "p111a8dd208104ce3b11c78398f658ca8,p4318f85d2aa14c43b1dba6f9b78be9b3,aacb2bb066e9c4478834d4196e76c49d3",
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
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 *UpdateDocProcessQueueOptions) (*UpdateDocProcessQueueResult, *Response, error)
```

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)
```

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：2022-04-22 12:14:19

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 *CreateDocProcessJobsOptions) (*CreateDocProcessJobsResult, *Response, error)
```

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/aHR0cDovL3Rlc3QwMDUtMTI1MTcwNDcwOC5jb3MuYXAtY2hvbmdxaW5nLm15cWNsb3VkLmNvbS8xLmpwZw==/gravity/southeast",
},
},
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${Page}表示多个输出文件，序号与预览页码一致，例如输入 abc_${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

参数名称	描述	类型
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) (*DescribeDocProcessJobResult, *Response, error)
```

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 *DescribeDocProcessJobsOptions) (*DescribeDocProcessJobsResult, *Response, error)
```

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：2022-08-04 10:53:49

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 *DocPreviewHTMLOptions) (*Response, error)
```

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"`
```

版权所有：腾讯云计算（北京）有限责任公司

```
HtmlRotate string `url:"htmlrotate,omitempty"`
HtmlHorizontal string `url:"htmlhorizontal,omitempty"`
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 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：2022-08-16 18:10:58

简介

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 string) (*ImageRecognitionResult, *Response, error)
// 支持指定更多参数
func (s *CIService) ImageAuditing(ctx context.Context, name string, opt *ImageRecognitionOptions) (*ImageRecognitionResult, *Response, erro
r)
```

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 *BatchImageAuditingOptions) (*BatchImageAuditingJobResult, *Response, error)
```

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

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) (*GetImageAuditingJobResult, *Response, error)
```

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：2022-10-14 15:40:49

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 *PutVideoAuditingJobOptions) (*PutVideoAuditingJobResult, *Response, error)
```

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，通过对截图逐一审核而实现的，该参数用于指定视频截帧的配置。	Struct
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型
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) (*GetVideoAuditingJobResult, *Response, error)
```

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：2022-08-16 18:11:11

简介

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 *PutAudioAuditingJobOptions) (*PutAudioAuditingJobResult, *Response, error)
```

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
```

版权所有：腾讯云计算（北京）有限责任公司

```
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) (*GetAudioAuditingJobResult, *Response, error)
```

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：2022-10-14 15:41:16

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 *PutTextAuditingJobOptions) (*PutTextAuditingJobResult, *Response, error)
```

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）、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。	String
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) (*GetTextAuditingJobResult, *Response, error)
```

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 := "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。

文档审核

最近更新时间：2022-10-14 15:41:41

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
创建任务	用于提交一个文档审核任务
查询任务	用于查询指定的文档审核任务

创建任务

PutDocumentAuditingJob 接口用来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。

方法原型

```
func (s *CIService) PutDocumentAuditingJob(ctx context.Context, opt *PutDocumentAuditingJobOptions) (*PutDocumentAuditingJobResult, *Response, error)
```

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.PutDocumentAuditingJobOptions{
InputUrl: "http://www.example.com/doctest.docx",
InputType: "docx",
Conf: &cos.DocumentAuditingJobConf{
DetectType: "Porn,Ads",
},
}
res, _, err := c.CI.PutDocumentAuditingJob(context.Background(), opt)
```

参数说明

```
type PutDocumentAuditingJobOptions struct {
InputObject string
InputUrl string
InputType string
InputDataId string
InputUserInfo *UserExtraInfo
Conf *DocumentAuditingJobConf
}
type UserExtraInfo struct {
TokenId string
Nickname string
DeviceId string
AppId string
Room string
IP string
Type string
}
type DocumentAuditingJobConf struct {
DetectType string
Callback string
BizType string
}
```

参数名称	参数描述	类型
InputObject	存储在 COS 存储桶中的文件名称，例如在目录 test 中的文件test.doc，则文件名称为 test/test.doc。Object 和 Url 只能选择其中一种。	String
InputUrl	文档文件的链接地址，例如 http://www.example.com/doctest.doc，Object 和 Url 只能选择其中一种。	String
InputType	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
InputDataId	文档标识，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

结果说明

调用 PutDocumentAuditingJob 函数，会解析 api 返回的 xml 内容到 PutDocumentAuditingJobResult 结构，具体返回参数可查看 提交文档审核任务 文档。

查询任务

GetDocumentAuditingJob 接口用来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。

方法原型

```
func (s *CIService) GetDocumentAuditingJob(ctx context.Context, jobid string) (*GetDocumentAuditingJobResult, *Response, error)
```

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 := "sdce25f391a72e11eb99f********"
res, _, err := c.CI.GetDocumentAuditingJob(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 GetDocumentAuditingJob 函数，会解析 api 返回的 xml 内容到 GetDocumentAuditingJobResult 结构，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：2022-10-14 15:41:52

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
创建任务	用于提交一个网页审核任务
查询任务	用于查询指定的网页审核任务

创建任务

PutWebpageAuditingJob 接口用于提交一个网页审核任务。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) PutWebpageAuditingJob(ctx context.Context, opt *PutWebpageAuditingJobOptions) (*PutWebpageAuditingJobResult, *Response, error)
```

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.PutWebpageAuditingJobOptions{
InputUrl: "http://www.test.com",
Conf: &cos.WebpageAuditingJobConf{
DetectType: "Porn,Ads",
},
}
res, _, err := c.CI.PutWebpageAuditingJob(context.Background(), opt)
```

参数说明

```
type PutWebpageAuditingJobOptions struct {
InputUrl string
```

```
InputDataId string
InputUserInfo *UserExtraInfo
Conf *WebpageAuditingJobConf
}
type UserExtraInfo struct {
TokenId string
Nickname string
DeviceId string
AppId string
Room string
IP string
Type string
}
type WebpageAuditingJobConf struct {
DetectType string
Callback string
ReturnHighlightHtml bool
}
```

参数名称	参数描述	类型
InputUrl	网页文件的链接地址，例如 http://www.test.com。	String
InputDataId	网页标识，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
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	回调地址，以 http:// 或者 https:// 开头的地址。	String
ReturnHighlightHtml	指定是否需要高亮展示网页内的违规文本，并返回高亮展示的 html 链接。取值为 true 和 false，默认为 false。	Bool

结果说明

调用 PutWebpageAuditingJob 函数，会解析 API 返回的 xml 内容到 PutWebpageAuditingJobResult 结构，具体返回参数可查看 提交网页审核任务 文档。

查询任务

GetWebpageAuditingJob 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。

方法原型

```
func (s *CIService) GetWebpageAuditingJob(ctx context.Context, jobid string) (*GetWebpageAuditingJobResult, *Response, error)
```

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 := "shce25f391a72e11eb99f********"
res, _, err := c.CI.GetWebpageAuditingJob(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 GetWebpageAuditingJob 函数，会解析 API 返回的 xml 内容到 GetWebpageAuditingJobResult 结构，具体返回参数可查看 查询网页审核任务结果 文档。

AI 内容识别
图片二维码

最近更新时间：2022-07-28 15:03:10

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 *ObjectGetOptions, id ...string) (*GetQRcodeResult, *Response, error)
```

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) (*GenerateQRcodeResult, *Response, error)

func (s *CIService) GenerateQRcodeToFile(ctx context.Context, filePath string, opt *GenerateQRcodeOptions) (*GenerateQRcodeResult, *Respo
nse, error)
```

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

图片标签

最近更新时间：2022-08-16 09:24:11

简介

本文档提供关于图片标签的 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

功能说明

获取图片中置信度较高的主题标签。

方法原型

```
func (s *CIService) PicTag(ctx context.Context, key string) (*PicTagResult, *Response, error)
```

请求示例

```
key := "pic/demo.jpg"
res, _, err := c.CI.PicTag(context.Background(), key)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg

结果说明

```
type PicTagResult struct {
XMLName xml.Name `xml:"RecognitionResult"`
Labels []PicTag `xml:"Labels,omitempty"`
}

type PicTag struct {
Confidence int `xml:"Confidence,omitempty"`
Name string `xml:"Name,omitempty"`
}
```

图片质量评估

最近更新时间：2022-08-16 09:24:15

简介

本文档提供关于图片质量评估的 API 概览以及 SDK 示例代码。

API	说明
图片质量评估	图片质量评估功能为同步请求方式，您可以通过本接口对图片文件进行检测，从多方面评估，最终给出综合可观的清晰度评分和主观的美观度评分。该接口属于 GET 请求。

图片质量评估

功能说明

评估图片的清晰度和主观的美观度。

方法原型

```
func (s *CIService) ImageQuality(ctx context.Context, obj string) (*ImageQualityResult, *Response, error)
```

请求示例

```
key := "pic/car.png"
res, _, err := c.CI.ImageQuality(context.Background(), key)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg

结果说明

```
type ImageQualityResult struct {
XMLName xml.Name `xml:"Response"`
LongImage bool `xml:"LongImage,omitempty"`
BlackAndWhite bool `xml:"BlackAndWhite,omitempty"`
SmallImage bool `xml:"SmallImage,omitempty"`
BigImage bool `xml:"BigImage,omitempty"`
PureImage bool `xml:"PureImage,omitempty"`
ClarityScore int `xml:"ClarityScore,omitempty"`
AestheticScore int `xml:"AestheticScore,omitempty"`
RequestId string `xml:"RequestId,omitempty"`
}
```

版权所有：腾讯云计算（北京）有限责任公司

人脸识别

最近更新时间：2022-08-16 09:24:19

简介

本文档提供关于人脸识别的 API 概览以及 SDK 示例代码。

API	操作描述
人脸检测	人脸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人脸位置。
人脸特效	人脸特效支持人脸美颜、人脸性别变换、人脸年龄变化、人像分割的特效功能，适用于社交娱乐、广告营销、互动传播等场景。

人脸检测

功能说明

检测图片中的人脸位置。

方法原型

```
func (s *CIService) DetectFace(ctx context.Context, key string, opt *DetectFaceOptions) (*DetectFaceResult, *Response, error)
```

请求示例

```
obj := "pic/face.jpeg"
key := &cos.DetectFaceOptions{
MaxFaceNum: 2,
}
res, _, err := c.CI.DetectFace(context.Background(), key, opt)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
opt	人脸检测相关选项

结果说明

```
type DetectFaceResult struct {
XMLName xml.Name `xml:"Response"`
ImageWidth int `xml:"ImageWidth,omitempty"`
ImageHeight int `xml:"ImageHeight,omitempty"`
FaceModelVersion string `xml:"FaceModelVersion,omitempty"`
RequestId string `xml:"RequestId,omitempty"`
FaceInfos []FaceInfos `xml:"FaceInfos,omitempty"`
}

type FaceInfos struct {
X int `xml:"X,omitempty"`
Y int `xml:"Y,omitempty"`
Width int `xml:"Width,omitempty"`
Height int `xml:"Height,omitempty"`
}
```

版权所有：腾讯云计算（北京）有限责任公司

人脸特效

功能说明

人脸特效支持人脸美颜、人脸性别变换、人脸年龄变化、人像分割的特效功能。

方法原型

```
func (s *CIService) FaceEffect(ctx context.Context, obj string, opt *FaceEffectOptions) (*FaceEffectResult, *Response, error)
```

请求示例

```
obj := "pic/self.jpeg"
opt := &cos.FaceEffectOptions{
Type: "face-segmentation",
Whitening: 50,
}
res, _, err := c.CI.FaceEffect(context.Background(), obj, opt)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
opt	人脸特效相关选项

结果说明

```
type FaceEffectResult struct {
XMLName xml.Name `xml:"Response"`
ResultImage string `xml:"ResultImage,omitempty"`
ResultMask string `xml:"ResultMask,omitempty"`
}
```

版权所有：腾讯云计算（北京）有限责任公司

人脸核身

最近更新时间：2022-08-16 09:24:26

简介

本文档提供关于人脸核身的 API 概览以及 SDK 示例代码。

API	操作描述
身份证识别	支持中国大陆居民二代身份证正反面所有字段的识别，包括姓名、性别、民族、出生日期、住址、公民身份证号、签发机关、有效期限；具备身份证照片、人像照片的裁剪功能和翻拍、PS、复印件告警功能，以及边框和框内遮挡告警、临时身份证告警和身份证有效期不合法告警等扩展功能。
获取动作顺序	使用动作活体检测模式前，需调用本接口获取动作顺序。
获取数字验证码	使用数字活体检测模式前，需调用本接口获取数字验证码。
活体人脸核身	集成了活体检测和跟权威库进行比对的能力，传入一段视频和姓名、身份证号信息即可进行验证。对录制的自拍视频进行活体检测，从而确认当前用户为真人，可防止照片、视频、静态3D建模等各种不同类型的攻击。检测为真人后，再判断该视频中的人与权威库的证件照是否属于同一个人，实现用户身份信息核实。

身份证识别

功能说明

中国大陆居民二代身份证正反面所有字段的识别。

方法原型

```
func (s *CIService) IdCardOCRWhenCloud(ctx context.Context, key string, query *IdCardOCROptions) (*IdCardOCRResult, *Response, error) // 云上数据处理
func (s *CIService) IdCardOCRWhenUpload(ctx context.Context, key, filePath string, query *IdCardOCROptions, header *ObjectPutOptions) (*IdCardOCRResult, *Response, error) // 上传时处理
```

请求示例1：云上数据处理

```
key := "pic/idcard_1.png"
query := &cos.IdCardOCROptions{
Config: &cos.IdCardOCROptionsConfig{
CropPortrait: true,
CropIdCard: true,
CopyWarn: true,
BorderCheckWarn: true,
ReshootWarn: true,
DetectPsWarn: true,
TempIdWarn: true,
InvalidDateWarn: true,
Quality: true,
MultiCardDetect: true,
},
}
res, _, err := c.CI.IdCardOCRWhenCloud(context.Background(), key, query)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
query	身份证识别相关选项

请求示例2：上传时处理

```
key := "pic/idcard_0.png"
filePath := "./idcard_0.png"
query := &cos.IdCardOCROptions{
Config: &cos.IdCardOCROptionsConfig{
CropPortrait: true,
CropIdCard: true,
CopyWarn: true,
BorderCheckWarn: true,
ReshootWarn: true,
DetectPsWarn: true,
TempIdWarn: true,
InvalidDateWarn: true,
Quality: true,
MultiCardDetect: true,
},
}

res, _, err := c.CI.IdCardOCRWhenUpload(context.Background(), key, filePath, query, nil)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
filePath	本地文件路径
query	身份证识别相关选项
header	对象上传参数，详见 上传对象

结果说明

```
type IdCardOCRResult struct {
XMLName xml.Name `xml:"Response"`
IdInfo *IdCardInfo `xml:"IdInfo,omitempty"`
AdvancedInfo *IdCardAdvancedInfo `xml:"AdvancedInfo,omitempty"`
}

type IdCardInfo struct {
Name string `xml:"Name,omitempty"`
Sex string `xml:"Sex,omitempty"`
Nation string `xml:"Nation,omitempty"`
Birth string `xml:"Birth,omitempty"`
Address string `xml:"Address,omitempty"`
IdNum string `xml:"IdNum,omitempty"`
Authority string `xml:"Authority,omitempty"`
```

版权所有：腾讯云计算（北京）有限责任公司

```
ValidDate string `xml:"ValidDate,omitempty"`
}

type IdCardAdvancedInfo struct {
IdCard string `xml:"IdCard,omitempty"`
Portrait string `xml:"Portrait,omitempty"`
Quality string `xml:"Quality,omitempty"`
BorderCodeValue string `xml:"BorderCodeValue,omitempty"`
WarnInfos []string `xml:"WarnInfos,omitempty"`
}
```

获取动作顺序

功能说明

获取动作顺序。

方法原型

```
func (s *CIService) GetActionSequence(ctx context.Context) (*GetActionSequenceResult, *Response, error)
```

请求示例

```
res, _, err := c.CI.GetActionSequence(context.Background())
```

参数说明

无

结果说明

```
type GetActionSequenceResult struct {
XMLName xml.Name `xml:"Response"`
ActionSequence string `xml:"ActionSequence,omitempty"`
}
```

获取数字验证码

功能说明

获取数字验证码。

方法原型

```
func (s *CIService) GetLiveCode(ctx context.Context) (*GetLiveCodeResult, *Response, error)
```

请求示例

```
res, _, err := c.CI.GetLiveCode(context.Background())
```

参数说明

无

结果说明

版权所有：腾讯云计算（北京）有限责任公司

```
type GetLiveCodeResult struct {
XMLName xml.Name `xml:"Response"`
LiveCode string `xml:"LiveCode,omitempty"`
}
```

活体人脸核身

功能说明

对录制的自拍视频进行活体检测。

方法原型

```
func (s *CIService) LivenessRecognitionWhenCloud(ctx context.Context, key string, query *LivenessRecognitionOptions) (*LivenessRecognitionResult, *Response, error) // 云上数据处理
func (s *CIService) LivenessRecognitionWhenUpload(ctx context.Context, key, filePath string, query *LivenessRecognitionOptions, header *ObjectPutOptions) (*LivenessRecognitionResult, *Response, error) // 上传时处理
```

请求示例1：云上数据处理

```
key := "pic/self.mp4"
query := &cos.LivenessRecognitionOptions{
IdCard: "111222xxxxxxxxxxx",
Name: "张三",
LivenessType: "SILENT",
BestFrameNum: 2,
}
res, _, err := c.CI.LivenessRecognitionWhenCloud(context.Background(), key, query)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
query	活体人脸检测相关选项

请求示例2：上传时处理

```
key := "pic/self.mp4"
filePath := "./self.mp4"
query := &cos.LivenessRecognitionOptions{
IdCard: "111222xxxxxxxxxxx",
Name: "张三",
LivenessType: "SILENT",
BestFrameNum: 2,
}
res, _, err := c.CI.LivenessRecognitionWhenUpload(context.Background(), key, filePath, query, nil)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg

参数名称	参数描述
filePath	本地文件路径
query	活体人脸检测相关选项
header	对象上传参数，详见 上传对象

结果说明

```
type IdCardOCRResult struct {
XMLName xml.Name `xml:"Response"`
IdInfo *IdCardInfo `xml:"IdInfo,omitempty"`
AdvancedInfo *IdCardAdvancedInfo `xml:"AdvancedInfo,omitempty"`
}

type IdCardInfo struct {
Name string `xml:"Name,omitempty"`
Sex string `xml:"Sex,omitempty"`
Nation string `xml:"Nation,omitempty"`
Birth string `xml:"Birth,omitempty"`
Address string `xml:"Address,omitempty"`
IdNum string `xml:"IdNum,omitempty"`
Authority string `xml:"Authority,omitempty"`
ValidDate string `xml:"ValidDate,omitempty"`
}

type IdCardAdvancedInfo struct {
IdCard string `xml:"IdCard,omitempty"`
Portrait string `xml:"Portrait,omitempty"`
Quality string `xml:"Quality,omitempty"`
BorderCodeValue string `xml:"BorderCodeValue,omitempty"`
WarnInfos []string `xml:"WarnInfos,omitempty"`
}
```

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：2022-08-16 09:24:31

简介

本文档提供关于汽车识别的 API 概览以及 SDK 示例代码。

API	说明
车辆车牌检测	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。

车辆车牌检测

功能说明

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。

方法原型

```
func (s *CIService) DetectCar(ctx context.Context, key string) (*DetectCarResult, *Response, error)
```

请求示例

```
key := "pic/car.png"
res, _, err := c.CI.DetectCar(context.Background(), key)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg

结果说明

```
type DetectCarResult struct {
XMLName xml.Name `xml:"Response"`
RequestId string `xml:"RequestId,omitempty"`
CarTags []CarTags `xml:"CarTags,omitempty"`
}

type CarTags struct {
Serial string `xml:"Serial,omitempty"`
Brand string `xml:"Brand,omitempty"`
Type string `xml:"Type,omitempty"`
Color string `xml:"Color,omitempty"`
Confidence int `xml:"Confidence,omitempty"`
Year int `xml:"Year,omitempty"`
CarLocation []CarLocation `xml:"CarLocation,omitempty"`
PlateContent []PlateContent `xml:"PlateContent,omitempty"`
}

type CarLocation struct {
X int `xml:"X,omitempty"`
Y int `xml:"Y,omitempty"`
```

版权所有：腾讯云计算（北京）有限责任公司

```
}

type PlateContent struct {
Plate string `xml:"Plate,omitempty"`
Color string `xml:"Color,omitempty"`
Type string `xml:"Type,omitempty"`
PlateLocation *PlateLocation `xml:"PlateLocation,omitempty"`
}

type PlateLocation struct {
X int `xml:"X,omitempty"`
Y int `xml:"Y,omitempty"`
}
```

图片文字识别（OCR）

最近更新时间：2022-08-16 09:24:36

简介

本文档提供关于通用文字识别的 API 概览以及 SDK 示例代码。

API	说明
通用文字识别	通用文字识别功能（Optical Character Recognition，OCR）基于行业前沿的深度学习技术，将图片上的文字内容，智能识别为可编辑的文本，可应用于随手拍扫描、纸质文档电子化、电商广告审核等多种场景，大幅提升信息处理效率。

通用文字识别

功能说明

通用文字识别。

方法原型

```
func (s *CIService) OcrRecognition(ctx context.Context, key string, opt *OcrRecognitionOptions) (*OcrRecognitionResult, *Response, error)
```

请求示例

```
obj := "pic/ocr.png"
opt := &cos.OcrRecognitionOptions{
Type: "general",
LanguageType: "zh",
Isword: true,
EnableWordPolygon: true,
}
res, _, err := c.CI.OcrRecognition(context.Background(), obj, opt)
```

参数说明

参数名称	参数描述
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/pic/pic.jpg中，对象键为 pic/pic.jpg
opt	通用文字识别的参数

结果说明

```
type OcrRecognitionResult struct {
XMLName xml.Name `xml:"Response"`
TextDetections []TextDetections `xml:"TextDetections,omitempty"`
Language string `xml:"Language,omitempty"`
Angel float64 `xml:"Angel,omitempty"`
PdfPageSize int `xml:"PdfPageSize,omitempty"`
RequestId string `xml:"RequestId,omitempty"`
}

type TextDetections struct {
DetectedText string `xml:"DetectedText,omitempty"`
Confidence int `xml:"Confidence,omitempty"`
Polygon []Polygon `xml:"Polygon,omitempty"`
```

版权所有：腾讯云计算（北京）有限责任公司

```
ItemPolygon []ItemPolygon `xml:"ItemPolygon,omitempty"`
Words []Words `xml:"Words,omitempty"`
WordPolygon []WordPolygon `xml:"WordPolygon,omitempty"`
}

type Polygon struct {
X int `xml:"X,omitempty"`
Y int `xml:"Y,omitempty"`
}

type ItemPolygon struct {
X int `xml:"X,omitempty"`
Y int `xml:"Y,omitempty"`
Width int `xml:"Width,omitempty"`
Height int `xml:"Height,omitempty"`
}

type Words struct {
Confidence int `xml:"Confidence,omitempty"`
Character string `xml:"Character,omitempty"`
WordCoordPoint *WordCoordPoint `xml:"WordCoordPoint,omitempty"`
}

type WordCoordPoint struct {
WordCoordinate []Polygon `xml:"WordCoordinate,omitempty"`
}

type WordPolygon struct {
LeftTop *Polygon `xml:"LeftTop,omitempty"`
RightTop *Polygon `xml:"RightTop,omitempty"`
RightBottom *Polygon `xml:"RightBottom,omitempty"`
LeftBottom *Polygon `xml:"LeftBottom,omitempty"`
}
```

版权所有：腾讯云计算（北京）有限责任公司

iOS SDK
快速入门

最近更新时间：2022-02-23 17:23:51

简介

为了开发者更方便快速地使用数据万象的基础图片处理功能和 CDN 的云闪图片分发功能，我们提供以下5个模块，开发者可根据具体需求进行选择。

模块	功能
CloudInfinite（默认模块）	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。
Loader	使用 CIImageLoadRequest 实例，请求网络图片并返回图片 data 数据。
TPG	解码 TPG 格式图片并显示，可用于显示普通图片，也可用于 TPG 图。
AVIF	解码 AVIF 格式图片并显示，可用于显示普通图片，也可用于 AVIF 图（v1.3.8及以上）。
SDWebImage-CloudInfinite	依赖于 SDWebImage、CloudInfinite 模块，提供了数据万象基础图片处理功能。

> 说明：
>
> - 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 接口与参数文档请参见 SDK API。
- 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。

步骤1：安装 SDK CloudInfinite 模块

CloudInfinite 模块主要功能：

- 包含数据万象的基础图片处理操作，分别有图片缩放、剪裁、旋转、格式转换、质量变换、高斯模糊、锐化、水印、获取图片主题色等。
- 将每一个操作封装为一个 TransformItem，而且支持各个基础操作组合使用，并将组合好的操作集合构建出可以直接进行网络请求的 URL。

使用 Cocoapods 集成

> 说明：
>
> 推荐使用 cocoapod 方式，下面涉及集成相关模块步骤一致，不再重复说明。

1. 在您工程 Podfile 文件中添加模块：

```
pod 'CloudInfinite'
```

2. 在终端执行安装命令：

```
pod install
```

步骤2：开始使用

导入头文件

Objective-C

```
#import <CloudInfinite.h>
```

swift

在桥接文件中加入（项目名称-Bridging-Header）。

```
#import <CloudInfinite.h>
```

构建 CIImageLoadRequest 实例

1. 在使用 CloudInfinite 构建具有数据万象功能的图片 URL 时，首先需要实例化 CloudInfinite 类。

Objective-C

```
CloudInfinite * cloudInfinite = [CloudInfinite new];
```

swift

```
let cloudInfinite = CloudInfinite();
```

2. 实例化图片转换类 CITransformation，并设置相关操作，这里以 TPG 为例。更多功能请参见 基础图片处理。

Objective-C

```
CITransformation * transform = [CITransformation new];
[transform setFormatWith:CIImageTypeTPG options:CILoadTypeUrlFooter];
```

swift

```
let transform = CITransformation();
transform.setFormatWith(CIImageFormat.typeTPG, options: CILoadTypeEnum.urlFooter);
```

3. 使用 CloudInfinite 实例构建具有数据万象功能的图片 URL。
   - 同步方式构建

Objective-C

```
CIImageLoadRequest * imageloadRequest = [cloudInfinite requestWithBaseUrl:@"图片链接" transform:transform];
// 图片url
NSURL * imageURL = imageloadRequest.url;
// header参数
NSString * heaer = imageloadRequest.header;
```

swift

```
let imageloadRequest = cloudInfinite.request(withBaseUrl: "图片链接", transform: transform);
// 图片url
let imageURL = imageloadRequest.url;
// header参数
let heaer = imageloadRequest.header;
```

- 异步方式构建

Objective-C

```
[cloudInfinite requestWithBaseUrl:@"图片链接" transform:transform request:^(CIImageLoadRequest * _Nonnull request) {
// 图片url
request.url;
// header参数
request.header;
}];
```

swift

```
cloudInfinite.request(withBaseUrl: "", transform: transform) { (request) in
// 图片url
request.url;
// header 参数
request.header;
};
```

> 说明：
>
> - 仅在进行图片格式转换并且 options 设置为 loadtypeacceptheader 时，CIImageLoadRequest 的 header 参数为有效值。
> - 在使用自定义 loader 请求图片时，如果需要使用到 header，则 header 需要拼接为如下：

```
NSDictionary * header = @{@"accept":[NSString stringWithFormat:@"image/%@",request.header]};
```

步骤3：加载图片

构建成功 CIImageLoadRequest 实例后，使用 CloudInfinite/Loader 模块进行图片加载。集成 CloudInfinite/Loader 模块，如下所示：

```
pod 'CloudInfinite/Loader'
```

请求图片 Data 数据

请求图片 NSData 数据，适用于请求 TPG 图片（需集成 TPG 解码模块）或者需要对图片二进制数据进行额外处理的图片。

Objective-C

```
// request 为上一步构建成功的 CIImageLoadRequest 实例；
[[CIImageLoader shareLoader] loadData:request loadComplete:^(NSData * _Nullable data, NSError * _Nullable error) {
// data：请求到的图片 data 数据；
}];
```

swift

```
// request 为上一步构建成功的 CIImageLoadRequest 实例；
CIImageLoader.share().loadData(request) { (data, error) in
// data：请求到的图片 data 数据；
};
```

加载普通图片

Objective-C

```
// 传入图片控件 imageView 以及 构建成功的 CIImageLoadRequest 实例；
[[CIImageLoader shareLoader]display:imageView loadRequest:request placeHolder:nil loadComplete:^(NSData * _Nullable data, NSError * _Nullable error) {
// data：请求到的图片 data 数据；
}];
```

swift

```
// 传入图片控件 imageView 以及 构建成功的 CIImageLoadRequest 实例；
CIImageLoader.share().display(imageView, loadRequest: request, placeHolder: nil) { (data, error) in
// data：请求到的图片 data 数据；
};
```

基础图片处理

最近更新时间：2021-12-03 18:30:28

简介

数据万象提供两种方式进行图片处理：图片自适应处理和 CloudInfinite 模块处理。

图片自适应处理

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）和裁剪（根据图片控件尺寸裁剪），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。

```
CIResponsiveTransformation * sTransform = [[CIResponsiveTransformation alloc]initWithView:imageView
scaleType:ScaleTypeAUTOFit]
```

CloudInfinite 模块处理

CloudInfinite 模块中的 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

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形
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例；
[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.CIGravityCenter);
```

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

版权所有：腾讯云计算（北京）有限责任公司

```
// 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80后，得到处理结果图的图片质量为64（80x80%）
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.com/google.jpg" gravity:0 dx:0 dy:0 blogo:0];
```

swift

```
transform.setWaterMarkWithImageUrl("", gravity: CloudInfiniteGravity.CIGravityCenter, dx: 0, dy: 0, blogo: CIWaterImageMarkBlogoEnum.none)
```

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 gravity:CIGravitySouth dx:100 dy:100 batch:YES degree:45];
```

swift

```
transform.setWaterMarkText("腾讯云数据万象", font: nil, textColor: nil, dissolve: 90, gravity: CloudInfiniteGravity.CIGravityCenter, dx: 100, dy: 100, batch: true, degree: 45)
```

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。

版权所有：腾讯云计算（北京）有限责任公司

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()
```

加载 TPG 图片

最近更新时间：2021-12-03 18:30:16

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:^(CIImageLoadRequest * _Nonnull request) {
// 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.urlFooter);

// 构建图片 CIImageLoadRequest
cloudInfinite.request(withBaseUrl: "图片链接", transform: transform) { (request) in
// 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 data，UIImage * _Nullable image, NSError * _Nullable err
or) {
}];
```

swift

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

加载 AVIF 图片

最近更新时间：2021-12-03 18:26:06

本文提供加载网络以及使用 AVIF 模块两种方式加载 AVIF 图片。

方式一：加载网络 AVIF 图片

> ⚠ 注意：
> CloudInfinite SDK 版本需要大于等于 v1.3.8。

1. 首先集成 CloudInfinite。

```
pod 'CloudInfinite'
```

2. 在 CloudInfinite 模块中构建出请求 AVIF 格式图片的链接，然后与 SDWebImage 配合使用，加载网络 AVIF 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:CIImageTypeAVIF options:CILoadTypeUrlFooter];
// 构建图片 CIImageLoadRequest
[cloudInfinite requestWithBaseUrl:@"图片链接" transform:transform request:^(CIImageLoadRequest * _Nonnull request) {
// 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.typeAVIF, options: CILoadTypeEnum.urlFooter);

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

方式二：使用 AVIF 模块加载 AVIF 图片

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

1. 首先集成 AVIF 模块。

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

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

Objective-C

```
[self.avifImageView setAvifImageWithData:data loadComplete:^(NSData * _Nullable data，UIImage * _Nullable image, NSError * _Nullable err
or) {
}];
```

swift

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

使用 SDWebImage 加载图片

最近更新时间：2021-12-03 18:30:05

本文主要介绍如何使用 SDWebImage 加载图片。

基础图片处理

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

1. 在使用数据万象基础图片处理操作时，需要集成 CloudInfinite/SDWebImage-CloudInfinite 模块。

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

2. 使用 UIImageView+CI，模仿 SDWebImage 调用风格，封装了一组可以传入 transform 的方法。

Objective-C

```
// 实例化 CITransformation 类并添加需要使用的操作；
CITransformation * transform = [CITransformation new];
// 加入缩放功能
[transform setZoomWithPercent:50 scaleType:ScalePercentTypeOnlyWidth];
// 加入圆角功能
[transform setCutWithRRadius:100];
// 根据实际需求加入相应的功能
...
// 使用 UIImageView+CI 类种方法，加载图片
[self.imageView sd_CI_setImageWithURL:[NSURL URLWithString:@"图片链接"] transformation:transform];
```

swift

```
// 实例化 CITransformation 类并添加需要使用的操作；
let transform = CITransformation();
// 加入缩放功能
transform.setZoomWithPercent(50, scale: ScalePercentType.onlyWidth)
// 加入圆角功能
transform.setCutWithRRadius(100)
// 根据实际需求加入相应的功能
...
// 使用 UIImageView+CI 类种方法，加载图片
imageView.sd_CI_setImage(with: NSURL.fileURL(withPath: "图片链接"), transformation: transform)
```

使用数据万象 TPG 功能

与 SDWebImage 配合使用数据万象 TPG 功能，支持 TPG 动图加载，无需额外处理。在使用 TPG 功能时，SDWebImage-CloudInfinite 需要依赖 CloudInfinite/TPG 模块。

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

SDWebImage-CloudInfinite 提供了两种加载 TPG 图片的方式：

方式一：调用 UIImageView+CI 加载 TPG

Objective-C

```
// 构建 CITransformation实例
CITransformation * tran = [CITransformation new];

// 设置TPG格式以及传参方式
[tran setFormatWith:CIImageTypeTPG options:CILoadTypeUrlFooter];
```

```
// 调用UIImageView+CI 类种方法，加载图片
[self.imageView sd_CI_setImageWithURL:[NSURL URLWithString:@"图片链接"] transformation:transform];
```

swift

```
// 构建 CITransformation 实例
let transform = CITransformation();

// 设置 TPG 格式以及传参方式
transform.setFormatWith(CIImageFormat.typeTPG, options: CILoadTypeEnum.urlFooter);

// 调用 UIImageView+CI 类种方法，加载图片
imageView.sd_CI_setImage(with: NSURL.init(string: "图片链接"), transformation: transform)
```

方式二：全局配置加载 TPG

适用场景

1. 如果整个项目都需要使用 TPG，或为已有项目接入 TPG。
2. 或者某些固定模式的图片链接需要使用 TPG。

使用方法

Objective-C

```
// 在项目启动时给 CIDownloaderConfig 添加需要使用 TPG 图片的链接正则表达式；满足这个正则的图片链接都使用 TPG 格式加载；
// 所有图片都使用 TPG 加载
[[CIDownloaderConfig sharedConfig] addTPGRegularExpress:@"http(s)?:.*" paramsType:CILoadTypeUrlFooter];

// 如果有的链接不需要使用 TPG，给 CIDownloaderConfig 添加排除的正则；
// 请求图片主题色排除
[[CIDownloaderConfig sharedConfig] addExcloudeTPGRegularExpress:@"http(s)?:.*imageAve"];
```

swift

```
// 在项目启动时给 CIDownloaderConfig 添加需要使用 TPG 图片的链接正则表达式；满足这个正则的图片链接都使用 TPG 格式加载；
// 所有图片都使用 TPG 加载
CIDownloaderConfig.shared().addTPGRegularExpress("http(s)?:.*", paramsType: CILoadTypeEnum.urlFooter)

// 如果有的链接不需要使用 TPG，给 CIDownloaderConfig 添加排除的正则；
// 请求图片主题色排除
CIDownloaderConfig.shared().addExcloudeTPGRegularExpress("http(s)?:.*imageAve")
```

使用数据万象 WEBP 功能

与 SDWebImage 配合使用数据万象 WEBP 功能，支持 WEBP 动图加载，无需额外处理。

准备工作

在使用 WEBP 功能时，SDWebImage-CloudInfinite 需要依赖 SDWebImageWebPCoder 库。

```
pod 'SDWebImageWebPCoder'
```

加载 WEBP 图片

Objective-C

```
// 实例化 CITransformation 类
CITransformation * tran = [CITransformation new];
// 设置转换为 webp 格式
[tran setFormatWith:CIImageTypeWEBP options:CILoadTypeUrlFooter];
// 加载图片
[self.imageView sd_CI_setImageWithURL:[NSURL URLWithString:@"图片链接"] transformation:tran];
```

swift

```
// 构建 CITransformation 实例
let transform = CITransformation();

// 设置 TPG 格式以及传参方式
transform.setFormatWith(CIImageFormat.typeWEBP, options: CILoadTypeEnum.urlFooter);

// 调用 UIImageView+CI 类种方法，加载图片
imageView.sd_CI_setImage(with: NSURL.init(string: "图片链接"), transformation: transform)
```

使用数据万象 AVIF 功能

与 SDWebImage 配合使用数据万象 AVIF 功能，支持 AVIF 动图加载，无需额外处理。在使用 AVIF 功能时，SDWebImage-CloudInfinite 需要依赖 CloudInfinite/AVIF 模块。

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

调用 UIImageView+CI 加载 AVIF

Objective-C

```
// 构建 CITransformation实例
CITransformation * tran = [CITransformation new];

// 设置AVIF格式以及传参方式
[tran setFormatWith:CIImageTypeAVIF options:CILoadTypeUrlFooter];

// 调用UIImageView+CI 类种方法，加载图片
[self.imageView sd_CI_setImageWithURL:[NSURL URLWithString:@"图片链接"] transformation:transform];
```

swift

```
// 构建 CITransformation 实例
let transform = CITransformation();

// 设置 AVIF 格式以及传参方式
transform.setFormatWith(CIImageFormat.typeAVIF, options: CILoadTypeEnum.urlFooter);

// 调用 UIImageView+CI 类种方法，加载图片
imageView.sd_CI_setImage(with: NSURL.init(string: "图片链接"), transformation: transform)
```

COS iOS SDK
快速入门

最近更新时间：2023-01-04 20:38:35

相关资源

- SDK 源码地址请参考： XML iOS SDK。
- 示例 Demo 可参考：XML iOS SDK Demo。
- SDK 接口与参数文档请参见 SDK API 参考。
- 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：iOS 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。如果您仍在使用 JSON 版本 SDK，请 升级到 XML iOS SDK。

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。
3. 您需要一个可以获取腾讯云临时密钥的远程地址，关于临时密钥的有关说明请参考 移动应用直传实践。

第一步：安装 SDK

方式一：使用 Cocoapods 集成（推荐）

标准版 SDK

在您工程的 Podfile 文件中使用：

```
pod 'QCloudCOSXML'
```

精简版 SDK

如果您仅仅使用到上传和下载功能，并且对 SDK 体积要求较高，可以使用我们的精简版 SDK。

精简版 SDK 是通过 Cocoapods 的 Subspec 功能实现的，因此目前只支持通过自动集成的方式。在您工程的 Podfile 文件中使用：

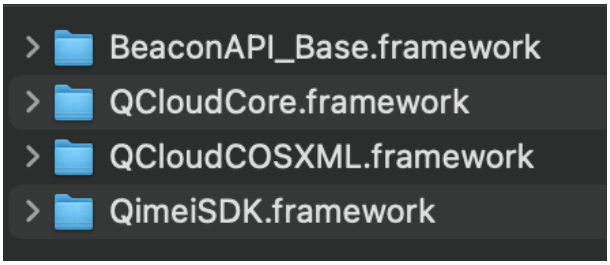
```
pod 'QCloudCOSXML/Transfer'
```

方式二：手动集成

您可以通过 快速下载地址 下载最新的正式包，也可以在 SDK Releases 里面找到我们所有历史版本的正式包。

1. 导入二进制库

将 QCloudCOSXML.framework、QCloudCore.framework 和 BeaconAPI_Base.framework 以及 QimeiSDK.framework 拖入到工程中。



并添加以下依赖库：

- CoreTelephony
- Foundation
- SystemConfiguration
- libc++.tbd

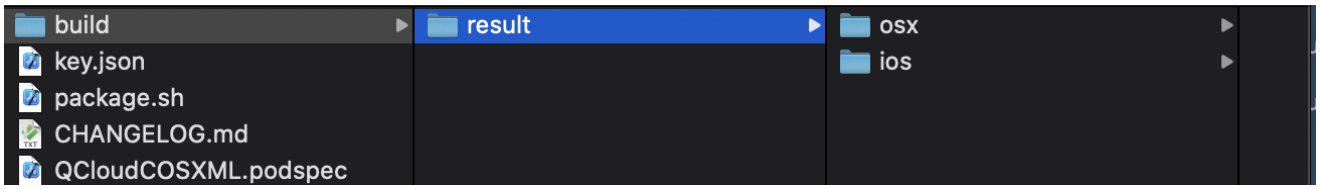
2. 工程配置

在 Build Settings 中设置 Other Linker Flags，加入以下参数：

```
-ObjC
-all_load
```



> 说明：
> SDK 提供了打包脚本，支持根据业务需求自行打包（该打包脚本依赖于 cocoapods，请先确保您的开发环境安装了 Cocoapods），打包步骤如下：
>
> 1. 下载源码：git clone https://github.com/tencentyun/qcloud-sdk-ios。
> 2. 运行打包脚本：source package.sh。
> 3. 将打包产物拖到工程中，然后按照上面手动集成的方式操作即可。
>   - iOS：将 ios 文件夹下的产物拖入项目。
>   - macOS：将 osx 文件夹下的产物拖入项目。



第二步：开始使用

1. 导入头文件

Objective-c

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

swift

```
import QCloudCOSXML
```

对于精简版 SDK，请导入：

Objective-c

```
#import <QCloudCOSXML/QCloudCOSXMLTransfer.h>
```

swift

```
import QCloudCOSXMLTransfer
```

2. 初始化 COS 服务并实现签名协议

> ⚠ 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。

方式一：获取临时密钥对请求授权（推荐）

SDK 在发出请求时，需要获取临时密钥计算签名，因此需要您实现 QCloudSignatureProvider 协议，在该协议中获取密钥后将密钥通过参数 continueBlock 回调给 SDK。

建议把初始化过程放在 AppDelegate 或者程序单例中。

具体步骤请参考下面的完整示例代码。

Objective-c

```
//AppDelegate.m
//AppDelegate 需遵循 QCloudSignatureProvider
@interface AppDelegate()<QCloudSignatureProvider>
@end

@implementation AppDelegate

- (BOOL)application:(UIApplication * )application
didFinishLaunchingWithOptions:(NSDictionary * )launchOptions {

QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
QCloudCOSXMLEndPoint* endpoint = [[QCloudCOSXMLEndPoint alloc] init];

// 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
// COS支持的所有region列表参见https://www.qcloud.com/document/product/436/6224
endpoint.regionName = @"COS_REGION";
// 使用 HTTPS
endpoint.useHTTPS = true;
configuration.endpoint = endpoint;
// 密钥提供者为自己
configuration.signatureProvider = self;
// 初始化 COS 服务示例
[QCloudCOSXMLService registerDefaultCOSXMLWithConfiguration:configuration];
[QCloudCOSTransferMangerService registerDefaultCOSTransferMangerWithConfiguration:
configuration];
return YES;
}


// 获取签名的方法入口，这里演示了获取临时密钥并计算签名的过程
// 您也可以自定义计算签名的过程
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
```

```
//这里同步从后台服务器获取临时密钥，强烈建议将获取临时密钥的逻辑放在这里，最大程度上保证密钥的可用性
//...
QCloudCredential* credential = [QCloudCredential new];

// 临时密钥 SecretId
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = @"SECRETID";
// 临时密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = @"SECRETKEY";
// 临时密钥 Token
// 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
credential.token = @"TOKEN";
/** 强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确(参数startTime和expiredTime单位为秒)
*/
credential.startDate = [NSDate dateWithTimeIntervalSince1970:startTime]; // 单位是秒
credential.expirationDate = [NSDate dateWithTimeIntervalSince1970:expiredTime]];// 单位是秒

QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc]
initWithCredential:credential];
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
QCloudSignature *signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}


@end
```

Swift

```
//AppDelegate.swift
//AppDelegate 需遵循 QCloudSignatureProvider

class AppDelegate: UIResponder, UIApplicationDelegate,
QCloudSignatureProvider {

func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
let config = QCloudServiceConfiguration.init();

let endpoint = QCloudCOSXMLEndPoint.init();

// 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
// COS支持的所有region列表参见https://www.qcloud.com/document/product/436/6224
endpoint.regionName = "COS_REGION";
// 使用 HTTPS
endpoint.useHTTPS = true;
config.endpoint = endpoint;
// 密钥提供者为自己
config.signatureProvider = self;

// 初始化 COS 服务示例
QCloudCOSXMLService.registerDefaultCOSXML(with: config);
QCloudCOSTransferMangerService.registerDefaultCOSTransferManger(
```

```
with: config);
return true
}


// 获取签名的方法入口，这里演示了获取临时密钥并计算签名的过程
// 您也可以自定义计算签名的过程
func signature(with fileds: QCloudSignatureFields!,
request: QCloudBizHTTPRequest!,
urlRequest urlRequst: NSMutableURLRequest!,
compelete continueBlock: QCloudHTTPAuthentationContinueBlock!) {

//这里同步从后台服务器获取临时密钥
//...

let credential = QCloudCredential.init();
// 临时密钥 SecretId
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = "SECRETID";
// 临时密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = "SECRETKEY";
// 临时密钥 Token
// 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
credential.token = "TOKEN";
/** 强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确(参数startTime和expiredTime单位为秒)
*/
credential.startDate = Date.init(timeIntervalSince1970: TimeInterval(startTime)!) DateFormatter().date(from: "startTime");
credential.expirationDate = Date.init(timeIntervalSince1970: TimeInterval(expiredTime)!)

let creator = QCloudAuthentationV5Creator.init(credential: credential);
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
let signature = creator?.signature(forData: urlRequst);
continueBlock(signature,nil);

}
}
```

> ⚠ 注意：
>
> - APPID 是您在成功申请腾讯云账户后所得到的账号，由系统自动分配，具有固定性和唯一性，可在 账号信息 中查看。腾讯云账号的 APPID，是与账号 ID 有唯一对应关系的应用 ID。
> - SecretId 和 SecretKey 合称为云 API 密钥，是用户访问腾讯云 API 进行身份验证时需要用到的安全凭证，可在 API 密钥管理 中获取。SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥。SecretId 用于标识 API 调用者身份。一个 APPID 可以创建多个云 API 密钥。

SDK 提供了一个 QCloudCredentailFenceQueue 的脚手架，实现对临时密钥的缓存与复用。脚手架在密钥过期之后会重新调用该协议的方法来重新获取新的密钥，直到该密钥过期时间大于设备的当前时间。

> ⚠ 注意：
>
> 脚手架只是根据密钥的过期时间是否大于设备的当前时间来判断密钥是否能复用，如果您申请密钥时设置了复杂的策略，则不建议使用脚手架工具。

建议把初始化过程放在 AppDelegate 或者程序单例中。使用脚手架您需要实现以下两个协议：

- QCloudSignatureProvider
- QCloudCredentailFenceQueueDelegate

完整示例代码请参考如下：

Objective-c

```
//AppDelegate.m
//AppDelegate 需遵循 QCloudSignatureProvider 与
//QCloudCredentailFenceQueueDelegate 协议

@interface AppDelegate()<QCloudSignatureProvider, QCloudCredentailFenceQueueDelegate>

// 一个脚手架实例
@property (nonatomic) QCloudCredentailFenceQueue* credentialFenceQueue;

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication * )application
didFinishLaunchingWithOptions:(NSDictionary * )launchOptions {
QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
QCloudCOSXMLEndPoint* endpoint = [[QCloudCOSXMLEndPoint alloc] init];

// 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
// COS支持的所有region列表参见https://www.qcloud.com/document/product/436/6224
endpoint.regionName = @"COS_REGION";
// 使用 HTTPS
endpoint.useHTTPS = true;
configuration.endpoint = endpoint;
// 密钥提供者为自己
configuration.signatureProvider = self;
// 初始化 COS 服务示例
[QCloudCOSXMLService registerDefaultCOSXMLWithConfiguration:configuration];
[QCloudCOSTransferMangerService registerDefaultCOSTransferMangerWithConfiguration:
configuration];

// 初始化临时密钥脚手架
self.credentialFenceQueue = [QCloudCredentailFenceQueue new];
self.credentialFenceQueue.delegate = self;

return YES;
}

- (void) fenceQueue:(QCloudCredentailFenceQueue * )queue requestCreatorWithContinue:(QCloudCredentailFenceQueueContinue)continueBloc
k
{
//这里同步从◊后台服务器获取临时密钥，强烈建议将获取临时密钥的逻辑放在这里，最大程度上保证密钥的可用性
//...

QCloudCredential* credential = [QCloudCredential new];
// 临时密钥 SecretId
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = @"SECRETID";
// 临时密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = @"SECRETKEY";
// 临时密钥 Token
```

```
// 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
credential.token = @"TOKEN";
/** 强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确(参数startTime和expiredTime单位为秒)
*/
credential.startDate = [NSDate dateWithTimeIntervalSince1970:startTime]; // 单位是秒
credential.expirationDate = [NSDate dateWithTimeIntervalSince1970:expiredTime]];// 单位是秒

QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc]
initWithCredential:credential];
continueBlock(creator, nil);
}

// 获取签名的方法入口，这里演示了获取临时密钥并计算签名的过程
// 您也可以自定义计算签名的过程
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
[self.credentialFenceQueue performAction:^(QCloudAuthentationCreator *creator,
NSError *error) {
if (error) {
continueBlock(nil, error);
} else {
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
}];
}


@end
```

Swift

```
//AppDelegate.swift
//AppDelegate 需遵循 QCloudSignatureProvider 与
//QCloudCredentailFenceQueueDelegate 协议

class AppDelegate: UIResponder, UIApplicationDelegate,
QCloudSignatureProvider, QCloudCredentailFenceQueueDelegate {

var credentialFenceQueue:QCloudCredentailFenceQueue?;

func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
let config = QCloudServiceConfiguration.init();

let endpoint = QCloudCOSXMLEndPoint.init();

// 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
// COS支持的所有region列表参见https://www.qcloud.com/document/product/436/6224
endpoint.regionName = "COS_REGION";
```

```
// 使用 HTTPS
endpoint.useHTTPS = true;
config.endpoint = endpoint;
// 密钥提供者为自己
config.signatureProvider = self;

// 初始化 COS 服务示例
QCloudCOSXMLService.registerDefaultCOSXML(with: config);
QCloudCOSTransferMangerService.registerDefaultCOSTransferManger(
with: config);

// 初始化临时密钥脚手架
self.credentialFenceQueue = QCloudCredentailFenceQueue.init();
self.credentialFenceQueue?.delegate = self;

return true
}

func fenceQueue(_ queue: QCloudCredentailFenceQueue!,
requestCreatorWithContinue continueBlock:
QCloudCredentailFenceQueueContinue!) {
//这里同步从后台服务器获取临时密钥
//...

let credential = QCloudCredential.init();
// 临时密钥 SecretId
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = "SECRETID";
// 临时密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = "SECRETKEY";
// 临时密钥 Token
// 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
credential.token = "TOKEN";
/** 强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确(参数startTime和expiredTime单位为秒)
*/
credential.startDate = Date.init(timeIntervalSince1970: TimeInterval(startTime)!) DateFormatter().date(from: "startTime");
credential.expirationDate = Date.init(timeIntervalSince1970: TimeInterval(expiredTime)!)

let auth = QCloudAuthentationV5Creator.init(credential: credential);
continueBlock(auth,nil);
}

// 获取签名的方法入口，这里演示了获取临时密钥并计算签名的过程
// 您也可以自定义计算签名的过程
func signature(with fileds: QCloudSignatureFields!,
request: QCloudBizHTTPRequest!,
urlRequest urlRequst: NSMutableURLRequest!,
compelete continueBlock: QCloudHTTPAuthentationContinueBlock!) {
self.credentialFenceQueue?.performAction({ (creator, error) in
if error != nil {
continueBlock(nil,error!);
}else{
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
let signature = creator?.signature(forData: urlRequst);
continueBlock(signature,nil);
```

```
}
})
}
}
```

⚠ 注意：

- 关于存储桶不同地域的简称请参考 地域和访问域名。
- 建议您使用 HTTPS 来请求数据，但如果您希望使用 HTTP 协议，为了确保在 iOS 9.0 以上的系统上可以运行，您需要为应用开启允许通过 HTTP 传输。详细的指引请参考 Apple 官方的说明文档 Preventing Insecure Network Connections。

如果您的 QCloudServiceConfiguration 发生改变，可以通过以下方法注册一个新的实例：

```
+ (QCloudCOSTransferMangerService*) registerCOSTransferMangerWithConfiguration:(QCloudServiceConfig
```

方式二：使用永久密钥进行本地调试

您可以使用腾讯云的永久密钥来进行开发阶段的本地调试。**由于该方式存在泄漏密钥的风险，请务必在上线前替换为临时密钥的方式。**

在使用永久密钥时，您可以不实现 QCloudCredentailFenceQueueDelegate 协议。

Objective-C

```
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{

QCloudCredential* credential = [QCloudCredential new];

// 永久密钥 secretID
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = @"SECRETID";
// 永久密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = @"SECRETKEY";
// 使用永久密钥计算签名
QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc]
initWithCredential:credential];
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
```

Swift

```
func signature(with fileds: QCloudSignatureFields!,
request: QCloudBizHTTPRequest!,
urlRequest urlRequst: NSMutableURLRequest!,
compelete continueBlock: QCloudHTTPAuthentationContinueBlock!) {
let credential = QCloudCredential.init();

// 永久密钥 secretID
// sercret_id替换为用户的 SecretId，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretID = "SECRETID";
```

```
// 永久密钥 SecretKey
// sercret_key替换为用户的 SecretKey，登录访问管理控制台查看密钥，https://console.cloud.tencent.com/cam/capi
credential.secretKey = "SECRETKEY";

// 使用永久密钥计算签名
let auth = QCloudAuthentationV5Creator.init(credential: credential);
// 注意 这里不要对urlRequst 进行copy以及mutableCopy操作
let signature = auth?.signature(forData: urlRequst)
continueBlock(signature,nil);
}
```

方式三：使用后台计算的签名对请求授权

在签名过程放在后台时，您可以不实现 QCloudCredentailFenceQueueDelegate 协议。

Objective-C

```
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
// 签名过期时间
NSDate *expiration = [[[NSDateFormatter alloc] init]
dateFromString:@"expiredTime"];
QCloudSignature *sign = [[QCloudSignature alloc] initWithSignature:
@"后台计算好的签名" expiration:expiration];
continueBlock(signature, nil);
}
```

Swift

```
func signature(with fileds: QCloudSignatureFields!,
request: QCloudBizHTTPRequest!,
urlRequest urlRequst: NSMutableURLRequest!,
compelete continueBlock: QCloudHTTPAuthentationContinueBlock!) {
// 签名过期时间
let expiration = DateFormatter().date(from: "expiredTime");
let sign = QCloudSignature.init(signature: "后台计算好的签名",
expiration: expiration);
continueBlock(signature,nil);
}
```

第三步：访问 COS 服务

上传对象

SDK 支持上传本地文件与二进制数据 NSData。下面以上传本地文件为例。

Objective-C

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];
// 本地文件路径
NSURL* url = [NSURL fileURLWithPath:@"文件的URL"];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
```

```
put.object = @"exampleobject";
//需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = url;
//监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];

//监听上传结果
[put setFinishBlock:^(id outputObject, NSError *error) {
//可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary * result = (NSDictionary *)outputObject;
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见**生成预签名链接**文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

Swift

```
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject";
//需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = NSURL.fileURL(withPath: "Local File Path") as AnyObject;

//监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if error != nil{
print(error!);
}else{
print(result!);
}
}

//监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};
//设置上传参数
put.initMultipleUploadFinishBlock = {(multipleUploadInitResult, resumeData) in
```

```
//在初始化分块上传完成以后会回调该 block，在这里可以获取 resumeData
//并且可以通过 resumeData 生成一个分块上传的请求
let resumeUploadRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>
.init(request: resumeData as Data?);
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见**生成预签名链接**文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

下载对象

Objective-C

```
QCloudCOSXMLDownloadObjectRequest * request = [QCloudCOSXMLDownloadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

//设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:@"Local File Path"];

//监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
//outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

//监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {
// bytesDownload 本次要下载的字节数（一个大文件可能要分多次发送）
// totalBytesDownload 已下载的字节数
// totalBytesExpectedToDownload 本次要下载的总字节数（即一个文件大小）
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] DownloadObject:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = "examplebucket-1250000000";
```

```
// 对象键
request.object = "exampleobject";

//设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

//监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in
// bytesDownload 本次要下载的字节数（一个大文件可能要分多次发送）
// totalBytesDownload 已下载的字节数
// totalBytesExpectedToDownload 本次要下载的总字节数（即一个文件大小）
}

//监听下载结果
request.finishBlock = { (copyResult, error) in
if error != nil{
print(error!);
}else{
print(copyResult!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

快速体验

最近更新时间：2022-08-25 09:05:29

简介

本文档将介绍如何配置客户端和运行示例 Demo。

相关资源

本文涉及的所有工具和 Demo 都存放在 Github 仓库，用户可前往获取。

搭建用户客户端

配置客户端

修改 QCloudCOSXMLDemo/QCloudCOSXMLDemo/key.json 文件，填入 APPID，secretID，secretKey 等参数值，然后执行以下命令：

```
pod install
```

> 说明：
> APPID，secretID，secretKey 可前往 API 密钥管理 页面获取。

执行命令完成后，打开 QCloudCOSXMLDemo.xcworkspace 即可进入 Demo 体验。

运行示例 Demo

查询存储桶列表

启动示例 App 后，将展示当前用户已创建的存储桶，如下图所示。



创建存储桶

点击右上角**新建桶**，在配置页面输入桶名称并选择存储桶的所属 地域，如下图所示。



查询对象列表

选择某个存储桶，进入存储桶详情，将看到该存储桶内所有文件以及文件夹，如下图所示。



上传文件

在文件列表页面点击右上角的**上传**，然后选择照片进行上传，支持设置文件的访问权限和传输状态控制，如下图所示。



下载或删除文件

选择并点击文件，然后点击文件的下载或删除按钮，对文件进行下载或删除操作，如下图所示。


Carrier
4:26 PM
我的存储桶
文件列表
上传
Ffff/
Jjj/
536291（1）.mp4ss
创建时间：2019-08-20 09:55:53
大小:12MB
下载
删除
无更多数据

存储桶操作

最近更新时间：2022-08-25 09:05:33

简介

本文档提供关于存储桶基本操作的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Service（List Buckets）	查询存储桶列表	查询指定账号下所有的存储桶列表
PUT Bucket	创建存储桶	在指定账号下创建一个存储桶
HEAD Bucket	检索存储桶及其权限	检索存储桶是否存在且是否有权限访问
DELETE Bucket	删除存储桶	删除指定账号下的空存储桶

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询存储桶列表

功能说明

用于查询指定账号下所有存储桶列表。

示例代码

Objective-C

```
// 获取所属账户的所有存储空间列表的方法
QCloudGetServiceRequest* request = [[QCloudGetServiceRequest alloc] init];
[request setFinishBlock:^(QCloudListAllMyBucketsResult* result,
NSError* error) {

// 从 result 中获取返回信息 存储桶列表
NSArray<QCloudBucket*> *buckets = result.buckets;

// bucket owner的信息
QCloudOwner *owner = result.owner;
}];
[[QCloudCOSXMLService defaultCOSXML] GetService:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
// 获取所属账户的所有存储空间列表的方法.
let getServiceReq = QCloudGetServiceRequest.init();
getServiceReq.setFinish{(result,error) in
if let result = result {
let buckets = result.buckets
let owner = result.owner
} else {
print(error!);
}
```

```
}
QCloudCOSXMLService.defaultCOSXML().getService(getServiceReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

创建存储桶

功能说明

创建一个存储桶（PUT Bucket）。

示例代码

Objective-C

```
// 创建存储桶
QCloudPutBucketRequest* request = [QCloudPutBucketRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError* error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] PutBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
// 创建存储桶
let putBucketReq = QCloudPutBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketReq.bucket = "examplebucket-1250000000";
putBucketReq.finishBlock = {(result,error) in
// 可以从 result 中获取服务器返回的 header 信息
if error != nil {
print(error!);
} else {
print(result!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucket(putBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

检索存储桶及其权限

功能说明

HEAD Bucket 请求可以确认该存储桶是否存在，是否有权限访问。有以下几种情况：

- 存储桶存在且有读取权限，返回 HTTP 状态码为200。
- 无存储桶读取权限，返回 HTTP 状态码为403。
- 存储桶不存在，返回 HTTP 状态码为404。

示例代码

Objective-C

```
QCloudHeadBucketRequest* request = [QCloudHeadBucketRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError* error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary * result = (NSDictionary *)outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML] HeadBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let headBucketReq = QCloudHeadBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
headBucketReq.bucket = "examplebucket-1250000000";

headBucketReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().headBucket(headBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

判断存储桶是否存在

功能说明

您可以通过 SDK 提供的快捷接口来判断 Bucket 是否存在

示例代码

Objective-C

```
// 存储桶名称，格式为 BucketName-APPID
[[QCloudCOSXMLService defaultCOSXML] doesBucketExist: @"examplebucket-1250000000"];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
// 存储桶名称，格式为 BucketName-APPID
QCloudCOSXMLService.defaultCOSXML().doesBucketExist("examplebucket-1250000000");
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除存储桶

功能说明

删除指定的存储桶（DELETE Bucket）。

> 注意：
> 删除存储桶前，请确保存储桶内的数据和未完成上传的分块数据已全部清空，否则会无法删除存储桶。

示例代码

Objective-C

```
QCloudDeleteBucketRequest* request = [[QCloudDeleteBucketRequest alloc ] init];

// 存储桶名称，命名格式：BucketName-APPID
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject,NSError*error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let deleteBucketReq = QCloudDeleteBucketRequest.init();

// 存储桶名称，命名格式：BucketName-APPID
deleteBucketReq.bucket = "examplebucket-1250000000";

deleteBucketReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
```

```
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteBucket(deleteBucketReq);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

对象操作
上传对象

最近更新时间：2022-08-25 09:05:38

简介

本文档提供关于上传对象的 API 概览以及 SDK 示例代码。

简单操作

API	操作名	操作描述
PUT Object	简单上传对象	上传一个对象至存储桶

分块操作

API	操作名	操作描述
List Multipart Uploads	查询分块上传	查询正在进行中的分块上传信息
Initiate Multipart Upload	初始化分块上传	初始化分块上传操作
Upload Part	上传分块	分块上传对象
List Parts	查询已上传块	查询特定分块上传操作中的已上传的块
Complete Multipart Upload	完成分块上传	完成整个文件的分块上传
Abort Multipart Upload	终止分块上传	终止一个分块上传操作并删除已上传的块

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

高级接口（推荐）

上传对象

高级接口封装了简单上传、分块上传接口，根据文件大小智能的选择上传方式，同时支持续传功能。

示例代码一: 上传本地文件

Objective-C

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];
/** 本地文件路径，请确保 URL 是以 file:// 开头，格式如下：
1. [NSURL URLWithString:@"file:////var/mobile/Containers/Data/Application/DBPF7490-D5U8-4ABF-A0AF-CC49D6A60AEB/Documents/exampleobject"]
2. [NSURL fileURLWithPath:@"/var/mobile/Containers/Data/Application/DBPF7490-D5U8-4ABF-A0AF-CC49D6A60AEB/Documents/exampleobject"]
*/
NSURL* url = [NSURL fileURLWithPath:@"文件的URL"];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = @"exampleobject";
// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = url;
// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
```

```
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];
// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * fileUrl = result.location;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[put setInitMultipleUploadFinishBlock:^(QCloudInitiateMultipartUploadResult *
multipleUploadInitResult,
QCloudCOSXMLUploadObjectResumeData resumeData) {
// 在初始化分块上传完成以后会回调该 block，在这里可以获取 resumeData，uploadid
NSString* uploadId = multipleUploadInitResult.uploadId;
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

Swift

```
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject";

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = NSURL.fileURL(withPath: "Local File Path") as AnyObject;

// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag
// 文件下载链接
let location = result.location;
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}
```

```
// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};
// 设置上传参数
put.initMultipleUploadFinishBlock = {(multipleUploadInitResult, resumeData) in
// 在初始化分块上传完成以后会回调该 block，在这里可以获取 resumeData,以及 uploadId
if let multipleUploadInitResult = multipleUploadInitResult {
let uploadId = multipleUploadInitResult.uploadId
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码二: 上传二进制数据

Objective-C

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = @"exampleobject";

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = [@"My Example Content" dataUsingEncoding:NSUTF8StringEncoding];

// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 新增字节数
// totalBytesSent 本次上传的总字节数
// totalBytesExpectedToSend 本地上传的目标字节数
}];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * fileUrl = result.location;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

Swift

```
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject";

// 需要上传的对象内容
let dataBody:NSData = "wrwrwrwrwrw".data(using: .utf8)! as NSData;
put.body = dataBody;

// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag

// 文件下载链接
let location = result.location;

// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}

// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in

// bytesSent 新增字节数
// totalBytesSent 本次上传的总字节数
// totalBytesExpectedToSend 本地上传的目标字节数
};

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

示例代码三: 上传暂停、继续与取消

Objective-C

对于上传任务，可以通过以下方式暂停：

```
NSError *error;
NSData *resmeData = [put cancelByProductingResumeData:&error];
```

暂停之后，可以通过以下方式续传：

```
QCloudCOSXMLUploadObjectRequest *resumeRequest = [QCloudCOSXMLUploadObjectRequest requestWithRequestData:resmeData];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:resumeRequest];
```

也通过以下方式取消上传：

```
//丢弃该上传
[put abort:^(id outputObject, NSError *error) {

}];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

对于上传任务，可以通过以下方式暂停：

```
var error : NSError?;
var uploadResumeData:Data = put.cancel(byProductingResumeData:&error) as Data;
```

暂停之后，可以通过以下方式续传：

```
var resumeRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>.init(request: uploadResumeData);
QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(resumeRequest);
```

也通过以下方式取消上传：

```
//丢弃该上传
put.abort { (outputObject, error) in

}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码四: 批量上传

Objective-C

```
for (int i = 0; i<20; i++) {
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
put.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 "video/xxx/movie.mp4"
put.object = [NSString stringWithFormat:@"exampleobject-%d",i];

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = [@"My Example Content" dataUsingEncoding:NSUTF8StringEncoding];

// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * fileUrl = result.location;

// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
}
```

Swift

```
for i in 1...10 {
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject-".appendingFormat("%d", i);

// 需要上传的对象内容
let dataBody:NSData = "wrwrwrwrwrw".data(using: .utf8)! as NSData;
put.body = dataBody;

// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag

// 文件的下载链接
let location = result.location;

// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
```

```
} else {
print(error!);
}
}

// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in

// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
}
```

示例代码五: 自定义简单上传与分块上传的阈值

Objective-C

```
for (int i = 0; i<20; i++) {
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = [NSString stringWithFormat:@"exampleobject-%d",i];

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = [@"My Example Content" dataUsingEncoding:NSUTF8StringEncoding];
// 自定义简单上传与分块上传的阈值：默认当文件大于1MB时启用分块上传
put.mutilThreshold = 10 *1024*1024;
// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * location = result.location;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
}
```

Swift

```
for i in 1...10 {
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject-".appendingFormat("%d", i);

// 需要上传的对象内容
let dataBody:NSData = "wrwrwrwrwrw".data(using: .utf8)! as NSData;
put.body = dataBody;
// 自定义简单上传与分块上传的阈值：默认当文件大于1MB时启用分块上传
put.mutilThreshold = 10 *1024*1024;
// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag
// 文件下载链接
let location = result.location;
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}

// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in

// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
}
```

示例代码六: 自定义分块大小

Objective-C

```
for (int i = 0; i<20; i++) {
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = [NSString stringWithFormat:@"exampleobject-%d",i];

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
```

```
put.body = [@"My Example Content" dataUsingEncoding:NSUTF8StringEncoding];
// 自定义分片大小，默认为1MB
put.sliceSize = 10 *1024*1024;
// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * location = result.location;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
}
```

Swift

```
for i in 1...10 {
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject-".appendingFormat("%d", i);

// 需要上传的对象内容
let dataBody:NSData = "wrwrwrwrwrw".data(using: .utf8)! as NSData;
put.body = dataBody;
// 自定义分片大小，默认为1MB
put.sliceSize = 10 *1024*1024
// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag

// 文件下载链接
let location = result.location;

// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}
```

```
// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
}
```

示例代码七: 上传时限速

> ⚠ 注意：
> 需要 COS iOS SDK v5.8.0 及以上版本。

Objective-C

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = [NSString stringWithFormat:@"exampleobject-%d",i];

// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
put.body = [@"My Example Content" dataUsingEncoding:NSUTF8StringEncoding];

// 使用trafficLimit参数设置上传限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
put.trafficLimit = 819200;
// 监听上传进度
[put setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
}];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * location = result.location;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

Swift

```
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();
```

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "exampleobject-".appendingFormat("%d", i);

// 需要上传的对象内容
let dataBody:NSData = "wrwrwrwrwrw".data(using: .utf8)! as NSData;
put.body = dataBody;
// 使用trafficLimit参数设置上传限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
put.trafficLimit = 819200;
// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件的 etag
let eTag = result.eTag

// 文件下载链接
let location = result.location;
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}

// 监听上传进度
put.sendProcessBlock = { (bytesSent, totalBytesSent,
totalBytesExpectedToSend) in
// bytesSent 本次要发送的字节数（一个大文件可能要分多次发送）
// totalBytesSent 已发送的字节数
// totalBytesExpectedToSend 本次上传要发送的总字节数（即一个文件大小）
};

QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
```

示例代码八: 创建目录

Objective-C

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.bucket = @"examplebucket-1250000000";
// 要创建的目录的名称
put.object = @"dir1";

// 需要上传的对象内容，如需创建目录使用空串生成NSData即可
put.body = [@"" dataUsingEncoding:NSUTF8StringEncoding];

// 监听上传结果
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError *error) {
// 在上传结果 result.location 中获取上传文件的下载链接
NSString * location = result.location;
}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

Swift

```
let put:QCloudCOSXMLUploadObjectRequest = QCloudCOSXMLUploadObjectRequest<AnyObject>();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = "dir/";

// 需要上传的对象内容
let dataBody:NSData = "".data(using: .utf8)! as NSData;
put.body = dataBody;

// 监听上传结果
put.setFinish { (result, error) in
// 获取上传结果
if let result = result {
// 文件下载链接
let location = result.location;

} else {
print(error!);
}
}
QCloudCOSTransferMangerService.defaultCOSTransferManager().uploadObject(put);
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

简单操作

简单上传对象

功能说明

PUT Object 接口可以上传一个对象至指定存储桶中，该操作需要请求者对存储桶有 WRITE 权限。最大支持上传不超过5GB的对象，5GB以上对象请使用 分块上传 或 高级接口 上传。

注意：

Key（文件名）不能以 / 结尾，否则会被识别为文件夹。

示例代码

Objective-C

```
QCloudPutObjectRequest* put = [QCloudPutObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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 "video/xxx/movie.mp4"
put.object = @"exampleobject";

// 文件内容，可以传入NSData*或者NSURL*类型的变量
put.body = [@"testFileContent" dataUsingEncoding:NSUTF8StringEncoding];

[put setFinishBlock:^(id result, NSError *error) {
// result 包含响应的 header 信息
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];

[[QCloudCOSXMLService defaultCOSXML] PutObject:put];
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

Swift

```
let putObject = QCloudPutObjectRequest<AnyObject>.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putObject.bucket = "examplebucket-1250000000";
// 需要上传的对象内容。可以传入NSData*或者NSURL*类型的变量
let dataBody:NSData? = "wrwrwrwrwrw".data(using: .utf8) as NSData?;
putObject.body = dataBody!;

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
putObject.object = "exampleobject";
putObject.finishBlock = {(result,error) in
if let result = result {
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putObject(putObject);
```

说明：

- 更多完整示例，请前往 GitHub 查看。
- 上传之后，您可以用同样的 Key 生成文件下载链接，具体使用方法见 **生成预签名链接** 文档。但注意如果您的文件是私有读权限，那么下载链接只有一定的有效期。

分块操作

这里说明下分块上传的流程。

分块上传的流程

1. 初始化分块上传（Initiate Multipart Upload），得到 UploadId
2. 使用 UploadId 上传分块（Upload Part）
3. 完成分块上传（Complete Multipart Upload）

分块继续上传的流程

1. 如果没有记录 UploadId，查询分块上传任务（List Multipart Uploads），得到对应文件的 UploadId
2. 使用 UploadId 列出已上传的分块（List Parts）
3. 使用 UploadId 上传剩余的分块（Upload Part）
4. 完成分块上传（Complete Multipart Upload）

终止分块上传的流程

1. 如果没有记录 UploadId，查询分块上传任务（List Multipart Uploads），得到对应文件的 UploadId
2. 终止分块上传并删除已上传分块（Abort Multipart Upload）

查询分块上传

功能说明

查询指定存储桶中正在进行的分块上传（List Multipart Uploads）。

示例代码

Objective-C

```
QCloudListBucketMultipartUploadsRequest* uploads = [QCloudListBucketMultipartUploadsRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
uploads.bucket = @"examplebucket-1250000000";
// 设置最大返回的 multipart 数量，合法取值从 1 到 1000
uploads.maxUploads = 100;
[uploads setFinishBlock:^(QCloudListMultipartUploadsResult* result,
NSError *error) {
// 可以从 result 中返回分块信息
// 进行中的分块上传对象
NSArray<QCloudListMultipartUploadContent*> *uploads = result.uploads;
}];
[[QCloudCOSXMLService defaultCOSXML] ListBucketMultipartUploads:uploads];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let listParts = QCloudListBucketMultipartUploadsRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
listParts.bucket = "examplebucket-1250000000";

// 设置最大返回的 multipart 数量，合法取值从 1 到 1000
listParts.maxUploads = 100;

listParts.setFinish { (result, error) in
if let result = result {
```

```
// 未完成的所有分块上传任务
let uploads = result.uploads;
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().listBucketMultipartUploads(listParts);
```

说明：
更多完整示例，请前往 GitHub 查看。

初始化分块上传

功能说明

初始化 Multipart Upload 上传操作，获取对应的 uploadId（Initiate Multipart Upload）。

示例代码

Objective-C

```
QCloudInitiateMultipartUploadRequest* initRequest = [QCloudInitiateMultipartUploadRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
initRequest.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 "video/xxx/movie.mp4"
initRequest.object = @"exampleobject";
// 将作为对象的元数据返回
initRequest.cacheControl = @"cacheControl";
initRequest.contentDisposition = @"contentDisposition";
// 定义 Object 的 ACL 属性。有效值：private，public-read-write，public-read；默认值：private
initRequest.accessControlList = @"public";
// 赋予被授权者读的权限。
initRequest.grantRead = @"grantRead";
// 赋予被授权者全部权限
initRequest.grantFullControl = @"grantFullControl";
[initRequest setFinishBlock:^(QCloudInitiateMultipartUploadResult* outputObject,
NSError *error) {
// 获取分块上传的 uploadId，后续的上传都需要这个 ID，请保存以备后续使用
self->uploadId = outputObject.uploadId;

}];

[[QCloudCOSXMLService defaultCOSXML] InitiateMultipartUpload:initRequest];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let initRequest = QCloudInitiateMultipartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
initRequest.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 "video/xxx/movie.mp4"
```

```
initRequest.object = "exampleobject";

initRequest.setFinish { (result, error) in
if let result = result {
// 获取分块上传的 uploadId，后续的上传都需要这个 ID，请保存以备后续使用
self.uploadId = result.uploadId;
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().initiateMultipartUpload(initRequest);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

上传分块

分块上传对象（Upload Part）。

示例代码

Objective-C

```
QCloudUploadPartRequest* request = [QCloudUploadPartRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";
// 块编号
request.partNumber = 1;
// 标识本次分块上传的 ID；使用 Initiate Multipart Upload 接口初始化分块上传时会得到一个 uploadId
request.uploadId = uploadId;
// 上传的数据：支持 NSData*，NSURL(本地 URL) 和 QCloudFileOffsetBody * 三种类型
request.body = [@"testFileContent" dataUsingEncoding:NSUTF8StringEncoding];
[request setSendProcessBlock:^(int64_t bytesSent,
int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
// 上传进度信息
// bytesSent 新增字节数
// totalBytesSent 本次上传的总字节数
// totalBytesExpectedToSend 本地上传的目标字节数
}];
[request setFinishBlock:^(QCloudUploadPartResult* outputObject, NSError *error) {
QCloudMultipartInfo *part = [QCloudMultipartInfo new];
// 获取所上传分块的 etag
part.eTag = outputObject.eTag;
part.partNumber = @"1";
// 保存起来用于最好完成上传时使用
self.parts = @[part];
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];

}];

[[QCloudCOSXMLService defaultCOSXML] UploadPart:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let uploadPart = QCloudUploadPartRequest<AnyObject>.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
uploadPart.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 "video/xxx/movie.mp4"
uploadPart.object = "exampleobject";
uploadPart.partNumber = 1;

// 标识本次分块上传的 ID
if let uploadId = self.uploadId {
uploadPart.uploadId = uploadId;
}

// 示例文件内容
let dataBody:NSData? = "wrwrwrwrwrwwrwrwrwrwwwrwrw"
.data(using: .utf8) as NSData?;

uploadPart.body = dataBody!;
uploadPart.setFinish { (result, error) in
if let result = result {
let mutipartInfo = QCloudMultipartInfo.init();
// 获取分块的 etag
mutipartInfo.eTag = result.eTag;
mutipartInfo.partNumber = "1";
// 保存起来用于最后完成上传时使用
self.parts = [mutipartInfo];
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}
uploadPart.sendProcessBlock = {(bytesSent,totalBytesSent,
totalBytesExpectedToSend) in
// 上传进度信息
// bytesSent 新增字节数
// totalBytesSent 本次上传的总字节数
// totalBytesExpectedToSend 本地上传的目标字节数

}
QCloudCOSXMLService.defaultCOSXML().uploadPart(uploadPart);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询已上传的分块

功能说明

查询特定分块上传操作中的已上传的块（List Parts）。

示例代码

Objective-C

```
QCloudListMultipartRequest* request = [QCloudListMultipartRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 在初始化分块上传的响应中，会返回一个唯一的描述符（upload ID）
request.uploadId = uploadId;

[request setFinishBlock:^(QCloudListPartsResult * _Nonnull result,
NSError * _Nonnull error) {

// 从 result 中获取已上传分块信息
// 用来表示每一个块的信息
NSArray<QCloudMultipartUploadPart*> *parts = result.parts;
}];

[[QCloudCOSXMLService defaultCOSXML] ListMultipart:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudListMultipartRequest.init();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
req.object = "exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

// 在初始化分块上传的响应中，会返回一个唯一的描述符（upload ID）
if let uploadId = self.uploadId {
req.uploadId = uploadId;
}
req.setFinish { (result, error) in
if let result = result {
// 所有已完成的分片
let parts = result.parts
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().listMultipart(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

完成分块上传

功能说明

完成整个文件的分块上传（Complete Multipart Upload）。

示例代码

Objective-C

```
QCloudCompleteMultipartUploadRequest *completeRequst = [QCloudCompleteMultipartUploadRequest new];
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
completeRequst.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
completeRequst.bucket = @"examplebucket-1250000000";
// 本次要查询的分块上传的 uploadId，可从初始化分块上传的请求结果 QCloudInitiateMultipartUploadResult 中得到
completeRequst.uploadId = uploadId;
// 已上传分块的信息
QCloudCompleteMultipartUploadInfo *partInfo = [QCloudCompleteMultipartUploadInfo new];
NSMutableArray * parts = [self.parts mutableCopy];
// 对已上传的块进行排序
[parts sortUsingComparator:^NSComparisonResult(QCloudMultipartInfo* _Nonnull obj1,
QCloudMultipartInfo* _Nonnull obj2) {
int a = obj1.partNumber.intValue;
int b = obj2.partNumber.intValue;

if (a < b) {
return NSOrderedAscending;
} else {
return NSOrderedDescending;
}
}];
partInfo.parts = [parts copy];
completeRequst.parts = partInfo;

[completeRequst setFinishBlock:^(QCloudUploadObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// 从 result 中获取上传结果
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];

[[QCloudCOSXMLService defaultCOSXML] CompleteMultipartUpload:completeRequst];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let complete = QCloudCompleteMultipartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

版权所有：腾讯云计算（北京）有限责任公司

```
complete.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 "video/xxx/movie.mp4"
complete.object = "exampleobject";

// 本次要查询的分块上传的 uploadId，可从初始化分块上传的请求结果
// QCloudInitiateMultipartUploadResult 中得到
complete.uploadId = "exampleUploadId";
if let uploadId = self.uploadId {
complete.uploadId = uploadId;
}

// 已上传分块的信息
let completeInfo = QCloudCompleteMultipartUploadInfo.init();
if self.parts == nil {
print("没有要完成的分块");
return;
}
if self.parts != nil {
completeInfo.parts = self.parts ?? [];
}

complete.parts = completeInfo;
complete.setFinish { (result, error) in
if let result = result {
// 文件的 eTag
let eTag = result.eTag
// 不带签名的文件链接
let location = result.location

// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().completeMultipartUpload(complete);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

终止分块上传

功能说明

终止一个分块上传操作并删除已上传的块（Abort Multipart Upload）。

示例代码

Objective-C

```
QCloudAbortMultipfartUploadRequest *abortRequest = [QCloudAbortMultipfartUploadRequest new];
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
abortRequest.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
abortRequest.bucket = @"examplebucket-1250000000";
```

```
// 本次要终止的分块上传的 uploadId
// 可从初始化分块上传的请求结果 QCloudInitiateMultipartUploadResult 中得到
abortRequest.uploadId = @"exampleUploadId";
[abortRequest setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary * result = (NSDictionary *)outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML]AbortMultipfartUpload:abortRequest];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let abort = QCloudAbortMultipfartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
abort.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 "video/xxx/movie.mp4"
abort.object = "exampleobject";

// 本次要查询的分块上传的 uploadId，可从初始化分块上传的请求结果
// QCloudInitiateMultipartUploadResult 中得到
abort.uploadId = self.uploadId!;

abort.finishBlock = {(result,error)in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().abortMultipfartUpload(abort);
```

说明：
更多完整示例，请前往 GitHub 查看。

下载对象

最近更新时间：2022-08-25 09:05:42

简介

本文档提供关于对象的下载操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Object	下载对象	下载一个对象至本地

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

高级接口（推荐）

下载对象

高级接口支持暂停、恢复以及取消下载请求，同时支持断点下载功能。

示例代码一：下载单个对象

Objective-C

```
QCloudCOSXMLDownloadObjectRequest * request = [QCloudCOSXMLDownloadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:@"Local File Path"];

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] DownloadObject:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}

// 监听下载结果
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：下载暂停、继续与取消

Objective-C

对于下载任务，可以通过以下方式暂停：

```
[request cancel];
```

暂停之后，可以通过以下方式续传：

```
// 本地已下载的文件大小
int64_t localCacheDownloadOffset = 0;
```

```
request.localCacheDownloadOffset = localCacheDownloadOffset;
```

也通过以下方式取消下载：

```
[request cancel];
```

> **说明：**
> **更多完整示例，请前往 GitHub 查看。**

Swift

对于下载任务，可以通过以下方式暂停：

```
request.cancel();
```

暂停之后，可以通过以下方式续传：

```
// 本地已下载的文件大小

let localCacheDownloadOffset = 100;
request.localCacheDownloadOffset = Int64(localCacheDownloadOffset);
```

也通过以下方式取消下载：

```
request.cancel();
```

> **说明：**
> **更多完整示例，请前往 GitHub 查看。**

示例代码三：断点下载

Objective-C

```
QCloudCOSXMLDownloadObjectRequest *getObjectRequest = [[QCloudCOSXMLDownloadObjectRequest alloc] init];
//支持断点下载，默认不支持
getObjectRequest.resumableDownload = true;
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getObjectRequest.bucket = transferTestBucket.name;
// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
getObjectRequest.downloadingURL = [NSURL URLWithString:QCloudTempFilePathWithExtension(@"downding")];
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
getObjectRequest.object = put.object;
// 监听下载结果
[getObjectRequest setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[getObjectRequest setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {
```

```
// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSTransferMangerService costransfermangerServiceForKey:kHTTPServiceKey] DownloadObject:getObjectRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

request.resumableDownload = true;
// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}

// 监听下载结果
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
}
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码四：批量下载

Objective-C

```
for (int i = 0; i<20; i++) {
QCloudCOSXMLDownloadObjectRequest * request = [QCloudCOSXMLDownloadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:@"Local File Path"];

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] DownloadObject:request];
}
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
for i in 1...10 {
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
```

```
request.localCacheDownloadOffset = 100;

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}

// 监听下载结果
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
}
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

示例代码四：下载文件夹及其文件

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";
// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

//要下载的文件夹在cos上的完整路径

request.prefix = @"cos_path";


[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
if(!error){
for (QCloudBucketContents *content in result.contents) {
QCloudCOSXMLDownloadObjectRequest * request = [QCloudCOSXMLDownloadObjectRequest new];

// 存储桶名称，格式为 BucketName-APPID
request.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"
request.object = content.key;

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:[@"Local File Path" stringByAppendingFormat:@"/%@",content.key]];
```

```
// 监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] DownloadObject:request];
}


}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();

// 存储桶名称，格式为 BucketName-APPID
getBucketReq.bucket = "examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;

//要下载的文件夹在cos上的完整路径
getBucketReq.prefix = "cos_path";

getBucketReq.setFinish { (result, error) in
if let result = result {
let contents = result.contents;
for content in contents {
let info = QCloudDeleteObjectInfo.init();
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，格式为 BucketName-APPID
request.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"
request.object = content.key;
```

```
// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中

request.downloadingURL = NSURL.fileURL(withPath: "Local File Path" ) as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}

// 监听下载结果
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
}
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

示例代码五：下载时限速

> **注意：**
> 需要 COS iOS SDK v5.9.5 及以上版本。

Objective-C

```
QCloudCOSXMLDownloadObjectRequest * request = [QCloudCOSXMLDownloadObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:@"Local File Path"];
```

```
// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 使用trafficLimit参数设置下载限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
request.trafficLimit = 819200;

// 监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] DownloadObject:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudCOSXMLDownloadObjectRequest = QCloudCOSXMLDownloadObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 使用trafficLimit参数设置下载限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
request.trafficLimit = 819200;

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
```

```
}

// 监听下载结果
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSTransferMangerService.defaultCOSTransferManager().downloadObject(request);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

简单操作

下载对象

功能说明

下载一个 Object（文件/对象）至本地（GET Object）。

示例代码一：下载对象

Objective-C

```
QCloudGetObjectRequest* request = [QCloudGetObjectRequest new];

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL URLWithString:QCloudTempFilePathWithExtension(@"downding")];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary* info = (NSDictionary *) outputObject;
// 服务端文件的crc64
result[@"x-cos-hash-crc64ecma"]
// 下载到本地的文件crc64 若返回的crc64与本地计算一致，则表示下载的文件与远端文件完整一致；
uint64_t localCrc64 = [本地文件data qcloud_crc64];
}];
[request setDownProcessBlock:^(int64_t bytesDownload, int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// 下载过程中的进度
// bytesDownload 一次下载的字节数，
// totalBytesDownload 总过接受的字节数
// totalBytesExpectedToDownload 文件一共多少字节

}];
```

```
[[QCloudCOSXMLService defaultCOSXML] GetObject:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getObject = QCloudGetObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getObject.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 "video/xxx/movie.mp4"
getObject.object = "exampleobject";
// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
getObject.downloadingURL = URL.init(string: NSTemporaryDirectory())!
.appendingPathComponent(getObject.object);
getObject.finishBlock = {(result,error) in
if let result = result {
// 服务端文件的crc64
result["x-cos-hash-crc64ecma"]
// 下载到本地的文件crc64 若返回的crc64与本地计算一致，则表示下载的文件与远端文件完整一致；
uint64_t localCrc64 = 本地文件data.qcloud_crc64();
} else {
print(error!);
}
};
getObject.downProcessBlock = {(bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in
// bytesDownload 一次下载的字节数，
// totalBytesDownload 总过接受的字节数
// totalBytesExpectedToDownload 文件一共多少字节
}
QCloudCOSXMLService.defaultCOSXML().getObject(getObject);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：下载时限速

> 注意：
> 需要 COS iOS SDK v5.9.5 及以上版本。

Objective-C

```
QCloudGetObjectRequest* request = [QCloudGetObjectRequest new];

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL URLWithString:QCloudTempFilePathWithExtension(@"downding")];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
```

版权所有：腾讯云计算（北京）有限责任公司

```
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 使用trafficLimit参数设置下载限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
request.trafficLimit = 819200;

[request setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary* info = (NSDictionary *) outputObject;
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];
[request setDownProcessBlock:^(int64_t bytesDownload, int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// 下载过程中的进度
// bytesDownload 一次下载的字节数，
// totalBytesDownload 总过接受的字节数
// totalBytesExpectedToDownload 文件一共多少字节

}];

[[QCloudCOSXMLService defaultCOSXML] GetObject:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getObject = QCloudGetObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getObject.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 "video/xxx/movie.mp4"
getObject.object = "exampleobject";
// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
getObject.downloadingURL = URL.init(string: NSTemporaryDirectory())!
.appendingPathComponent(getObject.object);

// 使用trafficLimit参数设置下载限速，单位为bit/s，限速值设置范围为819200 - 838860800，即100KB/s - 100MB/s。
getObject.trafficLimit = 819200;

getObject.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];
} else {
print(error!);
}
};
```

```
getObject.downProcessBlock = {(bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in
// bytesDownload 一次下载的字节数，
// totalBytesDownload 总过接受的字节数
// totalBytesExpectedToDownload 文件一共多少字节
}
QCloudCOSXMLService.defaultCOSXML().getObject(getObject);
```

说明：

更多完整示例，请前往 GitHub 查看。

列出对象

最近更新时间：2022-08-25 09:05:46

简介

本文档提供关于列出对象操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GET Bucket（List Objects）	查询对象列表	查询存储桶下的部分或者全部对象
GET Bucket Object Versions	查询对象及其历史版本列表	查询存储桶下的部分或者全部对象及其历史版本信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询对象列表

功能说明

查询存储桶下的部分或者全部对象。

示例代码一: 获取第一页数据

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

// 前缀匹配，用来规定返回的文件前缀地址
request.prefix = @"dir1/";

[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
// result 返回具体信息
// QCloudListBucketResult.contents 桶内文件数组
// QCloudListBucketResult.commonPrefixes 桶内文件夹数组
if (result.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
self->prevPageResult = result;
}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();
```

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketReq.bucket = "examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;

// 前缀匹配
getBucketReq.prefix = "dir/";

getBucketReq.setFinish { (result, error) in
if let result = result {
// 文件列表
let contents = result.contents

if (result.isTruncated) {
// 数据被截断，需要请求下一页数据
self.prevPageResult = result;
}
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：请求下一页数据

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// prevPageResult 是上一页的返回结果
// 分页参数 默认以UTF-8二进制顺序列出条目，所有列出条目从marker开始
request.marker = prevPageResult.nextMarker;

// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
// result 返回具体信息
// QCloudListBucketResult.contents 桶内文件数组
// QCloudListBucketResult.commonPrefixes 桶内文件夹数组
if (result.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
self->prevPageResult = result;
}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketReq.bucket = "examplebucket-1250000000";

// 分页参数 默认以UTF-8二进制顺序列出条目，所有列出条目从marker开始
if let result = self.prevPageResult {
getBucketReq.marker = result.marker
}

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;
// 前缀匹配
getBucketReq.prefix = "dir/";

getBucketReq.setFinish { (result, error) in
if let result = result {
// 文件列表
let contents = result.contents

if (result.isTruncated) {
// 数据被截断，需要请求下一页数据
self.prevPageResult = result;
}
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码三：获取对象列表与子目录

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

// 前缀匹配，用来规定返回的文件前缀地址
request.prefix = @"dir1/";

// 定界符为一个符号，如果有 Prefix，则将 Prefix 到 delimiter 之间的相同路径归为一类，
```

```
// 定义为 Common Prefix，然后列出所有 Common Prefix。如果没有 Prefix，则从路径起点开始
// delimiter:路径分隔符 固定为 /
request.delimiter = @"/";

// prevPageResult 是上一页的返回结果
// 分页参数 默认以UTF-8二进制顺序列出条目，所有列出条目从marker开始
request.marker = prevPageResult.nextMarker;

[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
// result 返回具体信息
// QCloudListBucketResult.contents 桶内文件数组
// QCloudListBucketResult.commonPrefixes 桶内文件夹数组
if (result.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
self->prevPageResult = result;
}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketReq.bucket = "examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;

// 前缀匹配，用来规定返回的文件前缀地址
getBucketReq.prefix = "dir/";

// 定界符为一个符号，如果有 Prefix，则将 Prefix 到 delimiter 之间的相同路径归为一类，
// 定义为 Common Prefix，然后列出所有 Common Prefix。如果没有 Prefix，则从路径起点开始
// delimiter:路径分隔符 固定为 /
getBucketReq.delimiter = "/";

// 分页参数 默认以UTF-8二进制顺序列出条目，所有列出条目从marker开始
if let result = self.prevPageResult {
getBucketReq.marker = result.marker
}

getBucketReq.setFinish { (result, error) in
if let result = result {
// 文件列表
let contents = result.contents

if (result.isTruncated) {
// 数据被截断，需要请求下一页数据
self.prevPageResult = result;
```

```
}
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询对象历史版本列表

功能说明

查询开启版本控制的存储桶下的部分或者全部对象。

示例代码：获取对象历史版本列表第一页数据

```
QCloudListObjectVersionsRequest* listObjectVersionsRequest = [[QCloudListObjectVersionsRequest alloc] init];

// 存储桶名称
listObjectVersionsRequest.bucket = @"bucketname";

// 一页请求数据条目数，默认 1000
listObjectVersionsRequest.maxKeys = 100;

//从当前key列出剩余的条目
listObjectVersionsRequest.keyMarker = prevPageResult.nextKeyMarker;
//从当前key的某个版本列出剩余的条目
listObjectVersionsRequest.versionIdMarker = prevPageResult.nextVersionIDMarkder;
[listObjectVersionsRequest setFinishBlock:^(QCloudListVersionsResult * _Nonnull result,
NSError * _Nonnull error) {

// 已删除的文件
NSArray<QCloudDeleteMarker*> *deleteMarker = result.deleteMarker;

// 对象版本条目
NSArray<QCloudVersionContent*> *versionContent = result.versionContent;

if (result.isTruncated) {
// 表示数据被截断，需要拉取下一页数据
self->prevPageResult = result;
}


}];

[[QCloudCOSXMLService defaultCOSXML] ListObjectVersions:listObjectVersionsRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

复制与移动对象

最近更新时间：2022-08-25 09:05:50

简介

本文档提供关于复制、移动对象的 API 概览以及 SDK 示例代码。

简单操作

API	操作名	操作描述
PUT Object - Copy	设置对象复制（修改对象属性）	复制文件到目标路径
DELETE Object	删除单个对象	在存储桶中删除指定对象

分块操作

API	操作名	操作描述
List Multipart Uploads	查询分块上传/复制	查询正在进行中的分块上传/复制信息
Initiate Multipart Upload	初始化分块上传/复制	初始化分块上传/复制操作
Upload Part - Copy	复制分块	将其他对象复制为一个分块
List Parts	查询已上传/复制块	查询特定分块操作中的已上传/复制的块
Complete Multipart Upload	完成分块上传/复制	完成整个文件的分块上传/复制
Abort Multipart Upload	终止分块上传/复制	终止一个分块操作并删除已上传/复制的块

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

高级接口（推荐）

复制对象

高级接口封装了简单复制、分块复制接口的异步请求，并支持暂停、恢复以及取消复制请求。

示例代码

Objective-C

```
QCloudCOSXMLCopyObjectRequest* request = [[QCloudCOSXMLCopyObjectRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 文件来源存储桶，需要是公有读或者在当前账号有权限
request.sourceBucket = @"sourcebucket-1250000000";

// 源文件名称
request.sourceObject = @"sourceObject";

// 源文件的 APPID
request.sourceAPPID = @"1250000000";
```

```
// 来源的地域
request.sourceRegion= @"COS_REGION";

[request setFinishBlock:^(QCloudCopyObjectResult* result, NSError* error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
}];

// 注意如果是跨地域复制，这里使用的 transferManager 所在的 region 必须为目标桶所在的 region
[[QCloudCOSTransferMangerService defaultCOSTransferManager] CopyObject:request];

// 取消copy
// 若需要取消copy 调用cancel方法
[request cancel];
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let copyRequest = QCloudCOSXMLCopyObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
copyRequest.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 "video/xxx/movie.mp4"
copyRequest.object = "exampleobject";

// 文件来源存储桶，需要是公有读或者在当前账号有权限
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
copyRequest.sourceBucket = "sourcebucket-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 "video/xxx/movie.mp4"
copyRequest.sourceObject = "sourceObject";

// 源文件的 APPID
copyRequest.sourceAPPID = "1250000000";

// 来源的地域
copyRequest.sourceRegion = "COS_REGION";

copyRequest.setFinish { (copyResult, error) in
if let copyResult = copyResult {
// 文件的 etag
let eTag = copyResult.eTag
} else {
print(error!);
}

}
// 注意如果是跨地域复制，这里使用的 transferManager 所在的 region 必须为目标桶所在的 region
QCloudCOSTransferMangerService.defaultCOSTransferManager().copyObject(copyRequest);

// 取消copy
```

```
// 若需要取消copy 调用cancel方法
copyRequest.cancel();
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

移动对象

移动对象主要包括两个操作：复制源对象到目标位置，删除源对象。

由于 COS 通过存储桶名称（Bucket）和对象键（ObjectKey）来标识对象。移动对象也就意味着修改这个对象的标识，COS Java SDK 目前没有提供修改对象唯一标识名的单独接口，但是可以通过组合**复制对象**加上**删除对象**的基本操作，来达到修改对象标识的目的，从而实现移动对象。

例如将 mybucket-1250000000 这个存储桶中的 picture.jpg 这个对象移动到同个存储桶的 doc 路径下。首先可以复制 picture.jpg 对象到存储桶的 doc 路径下，即对象键设定为 doc/picture.jpg，复制完成后删除 picture.jpg 这个对象，来实现“移动”的效果。

同样的，如果想将 mybucket-1250000000 这个存储桶里的 picture.jpg 这个对象移动到 myanothorbucket-1250000000 这个存储桶里，可以先将对象复制到 myanothorbucket-1250000000 存储桶，然后删除原来的对象。

示例代码

Objective-C

```
QCloudCOSXMLCopyObjectRequest* request = [[QCloudCOSXMLCopyObjectRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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"
request.object = @"exampleobject";

// 文件来源存储桶，需要是公有读或者在当前账号有权限
request.sourceBucket = @"sourcebucket-1250000000";

// 源文件名称
request.sourceObject = @"sourceObject";

// 源文件的 APPID
request.sourceAPPID = @"1250000000";

// 来源的地域
request.sourceRegion= @"COS_REGION";

[request setFinishBlock:^(QCloudCopyObjectResult* result, NSError* error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
if(!error){
QCloudDeleteObjectRequest* deleteObjectRequest = [QCloudDeleteObjectRequest new];

// 文件来源存储桶，需要是公有读或者在当前账号有权限
deleteObjectRequest.bucket = @"sourcebucket-1250000000";

// 源文件名称，是源对象在 COS 上的完整路径，如果带目录的话，格式为 "dir1/object1"
deleteObjectRequest.object = @"sourceObject";

[deleteObjectRequest setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
```

```
[[QCloudCOSXMLService defaultCOSXML] DeleteObject:deleteObjectRequest];
}
}];

// 注意如果是跨地域移动，这里使用的 transferManager 所在的 region 必须为目标桶所在的 region
[[QCloudCOSTransferMangerService defaultCOSTransferManager] CopyObject:request];
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let copyRequest = QCloudCOSXMLCopyObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
copyRequest.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"
copyRequest.object = "exampleobject";

// 文件来源存储桶，需要是公有读或者在当前账号有权限
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
copyRequest.sourceBucket = "sourcebucket-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"
copyRequest.sourceObject = "sourceObject";

// 源文件的 APPID
copyRequest.sourceAPPID = "1250000000";

// 来源的地域
copyRequest.sourceRegion = "COS_REGION";

copyRequest.setFinish { (copyResult, error) in
if let copyResult = copyResult {
// 文件的 etag
let deleteObject = QCloudDeleteObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
deleteObject.bucket = "sourcebucket-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"
deleteObject.object = "sourceObject";

deleteObject.finishBlock = {(result,error)in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteObject(deleteObject);
} else {
```

```
print(error!);
}

}
// 注意如果是跨地域移动，这里使用的 transferManager 所在的 region 必须为目标桶所在的 region
QCloudCOSTransferMangerService.defaultCOSTransferManager().copyObject(copyRequest);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

简单操作

复制对象（修改属性）

复制文件到目标路径（PUT Object-Copy）。

示例代码一：复制对象时保留对象属性

Objective-C

```
QCloudPutObjectCopyRequest* request = [[QCloudPutObjectCopyRequest alloc] init];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";
// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致，即用户试图修改元数据时，必须为 Replaced
request.metadataDirective = @"Copy";
// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
request.accessControlList = @"default";
// 源对象所在的路径
request.objectCopySource =
@"sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";
// 指定源文件的 versionID，只有开启或开启后暂停的存储桶，才会响应此参数
request.versionID = @"objectVersion1";
[request setFinishBlock:^(QCloudCopyObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// result 返回具体信息

}];
[[QCloudCOSXMLService defaultCOSXML] PutObjectCopy:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let putObjectCopy = QCloudPutObjectCopyRequest.init();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putObjectCopy.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 "video/xxx/movie.mp4"
```

```
putObjectCopy.object = "exampleobject";
// 源对象所在的路径
putObjectCopy.objectCopySource = "sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";
// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致，即用户试图修改元数据时，必须为 Replaced
putObjectCopy.metadataDirective = "Copy";
// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
putObjectCopy.accessControlList = "default";
// 指定源文件的 versionID，只有开启或开启后暂停的存储桶，才会响应此参数
putObjectCopy.versionID = "versionID";
putObjectCopy.setFinish { (result, error) in
if let result = result {
let eTag = result.eTag
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putObjectCopy(putObjectCopy);
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码二：复制对象时替换对象属性

Objective-C

```
QCloudPutObjectCopyRequest* request = [[QCloudPutObjectCopyRequest alloc] init];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";
// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致，即用户试图修改元数据时，必须为 Replaced
request.metadataDirective = @"Replaced";
// 修改元数据
[request.customHeaders setValue:@"newValue" forKey:@"x-cos-meta-*"];
// 对象存储类型，枚举值请参见 存储类型概述 文档，例如 MAZ_STANDARD，MAZ_STANDARD_IA，
// STANDARD_IA，ARCHIVE。仅当对象不是标准存储（STANDARD）时才会返回该头部
// 修改存储类型
[request.customHeaders setValue:@"newValue" forKey:@"x-cos-storage-class"];
// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
// 修改acl
request.accessControlList = @"private";
// 源对象所在的路径
request.objectCopySource =
@"sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";
```

```
// 指定源文件的 versionID，只有开启或开启后暂停的存储桶，才会响应此参数
request.versionID = @"objectVersion1";

[request setFinishBlock:^(QCloudCopyObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// result 返回具体信息

}];
[[QCloudCOSXMLService defaultCOSXML] PutObjectCopy:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudPutObjectCopyRequest = QCloudPutObjectCopyRequest();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";
// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致，即用户试图修改元数据时，必须为 Replaced
request.metadataDirective = "Replaced";
// 修改元数据
request.customHeaders.setValue("newValue", forKey: "x-cos-meta-*");
// 对象存储类型，枚举值请参见 存储类型概述 文档，例如 MAZ_STANDARD，MAZ_STANDARD_IA，
// STANDARD_IA，ARCHIVE。仅当对象不是标准存储（STANDARD）时才会返回该头部
// 修改存储类型
request.customHeaders.setValue("newValue", forKey: "x-cos-storage-class");
// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
// 修改acl
request.accessControlList = "源文件acl";
// 源对象所在的路径
request.objectCopySource = "sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";
// 指定源文件的 versionID，只有开启或开启后暂停的存储桶，才会响应此参数
request.versionID = "versionID";
request.setFinish { (result, error) in
if let result = result {
let eTag = result.eTag
} else {
print(error!);
}

}
QCloudCOSXMLService.defaultCOSXML().putObjectCopy(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码三: 修改对象元数据

Objective-C

```
QCloudPutObjectCopyRequest* request = [[QCloudPutObjectCopyRequest alloc] init];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";
// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致，即用户试图修改元数据时，必须为 Replaced
request.metadataDirective = @"Replaced";
// 自定义对象header
[request.customHeaders setValue:@"newValue" forKey:@"x-cos-meta-*"];
// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
request.accessControlList = @"default";
// 源对象所在的路径
request.objectCopySource =
@"examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject";

[request setFinishBlock:^(QCloudCopyObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// result 返回具体信息

}];
[[QCloudCOSXMLService defaultCOSXML] PutObjectCopy:request];
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudPutObjectCopyRequest = QCloudPutObjectCopyRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

// 是否拷贝元数据，枚举值：Copy，Replaced，默认值 Copy。
// 假如标记为 Copy，忽略 Header 中的用户元数据信息直接复制
// 假如标记为 Replaced，按 Header 信息修改元数据。当目标路径和原路径一致
// 即用户试图修改元数据时，必须为 Replaced
request.metadataDirective = "Replaced";

// 自定义对象header
request.customHeaders.setValue("newValue", forKey: "x-cos-meta-*")

// 定义 Object 的 ACL 属性，有效值：private，public-read，default。
// 默认值：default（继承 Bucket 权限）。
```

```
// 注意：如果您无需进行 Object ACL 控制，请填 default
// 或者此项不进行设置，默认继承 Bucket 权限。
request.accessControlList = "default";

// 源对象所在的路径
request.objectCopySource =
"examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject";

request.setFinish { (result, error) in
if let result = result {
// 生成的新文件的 etag
let eTag = result.eTag
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().putObjectCopy(request);
```

说明：
更多完整示例，请前往 GitHub 查看。

示例代码四: 修改对象存储类型

Objective-C

```
QCloudPutObjectCopyRequest* request = [[QCloudPutObjectCopyRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 对象存储类型，枚举值请参见 存储类型概述 文档，例如 MAZ_STANDARD，MAZ_STANDARD_IA，
// STANDARD_IA，ARCHIVE。仅当对象不是标准存储（STANDARD）时才会返回该头部
[request.customHeaders setValue:@"ARCHIVE" forKey:@"x-cos-storage-class"];

// 源对象所在的路径
request.objectCopySource =
@"examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject";

// 指定源文件的 versionID，只有开启或开启后暂停的存储桶，才会响应此参数
request.versionID = @"";

[request setFinishBlock:^(QCloudCopyObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// result 返回具体信息

}];
[[QCloudCOSXMLService defaultCOSXML] PutObjectCopy:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudPutObjectCopyRequest = QCloudPutObjectCopyRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = "exampleobject";

// 对象存储类型，枚举值请参见 存储类型概述 文档，例如 MAZ_STANDARD，MAZ_STANDARD_IA，
// STANDARD_IA，ARCHIVE。仅当对象不是标准存储（STANDARD）时才会返回该头部
request.customHeaders.setValue("newValue", forKey: "x-cos-storage-class");
// 源对象所在的路径
request.objectCopySource =
"examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/exampleobject";

request.setFinish { (result, error) in
if let result = result {
// 生成的新文件的 etag
let eTag = result.eTag
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().putObjectCopy(request);
```

说明：
更多完整示例，请前往 GitHub 查看。

分块操作

这里主要介绍分块复制的流程。分块复制和分块上传在涉及接口及使用上具有一定重合度。

分块复制的流程

1. 初始化分块复制（Initiate Multipart Upload），得到 UploadId。
2. 使用 UploadId 复制分块（Upload Part - Copy）。
3. 完成分块复制（Complete Multipart Upload）。

分块复制的流程

1. 如果没有记录 UploadId，查询分块复制任务（List Multipart Uploads），得到对应文件的 UploadId。
2. 使用 UploadId 列出已复制的分块（List Parts）。
3. 使用 UploadId 复制剩余的分块（Upload Part Copy）。
4. 完成分块复制（Complete Multipart Upload）。

终止分块复制的流程

1. 如果没有记录 UploadId，查询分块复制任务（List Multipart Upload），得到对应文件的 UploadId
2. 终止分块复制并删除已复制分块（Abort Multipart Upload）

查询分块复制

功能说明

查询指定存储桶中正在进行的分块复制（List Multipart Uploads），该接口与分块上传的查询分块上传接口一致。

示例代码

Objective-C

```
QCloudListBucketMultipartUploadsRequest* uploads = [QCloudListBucketMultipartUploadsRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
uploads.bucket = @"examplebucket-1250000000";
// 设置最大返回的 multipart 数量，合法取值从 1 到 1000
uploads.maxUploads = 100;
[uploads setFinishBlock:^(QCloudListMultipartUploadsResult* result,
NSError *error) {
// 可以从 result 中返回分块信息
// 进行中的分块复制对象
NSArray<QCloudListMultipartUploadContent*> *uploads = result.uploads;
}];
[[QCloudCOSXMLService defaultCOSXML] ListBucketMultipartUploads:uploads];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let listParts = QCloudListBucketMultipartUploadsRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
listParts.bucket = "examplebucket-1250000000";

// 设置最大返回的 multipart 数量，合法取值从 1 到 1000
listParts.maxUploads = 100;

listParts.setFinish { (result, error) in
if let result = result {
// 未完成的所有分块复制任务
let uploads = result.uploads;
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().listBucketMultipartUploads(listParts);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

初始化分块复制

功能说明

初始化 Multipart Upload 复制操作，获取对应的 uploadId（Initiate Multipart Upload），该接口与分块上传的初始化分块上传接口一致。

示例代码

Objective-C

```
QCloudInitiateMultipartUploadRequest* initRequest = [QCloudInitiateMultipartUploadRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
initRequest.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 "video/xxx/movie.mp4"
initRequest.object = @"exampleobject";
// 将作为对象的元数据返回
initRequest.cacheControl = @"cacheControl";
initRequest.contentDisposition = @"contentDisposition";
// 定义 Object 的 ACL 属性。有效值：private，public-read-write，public-read；默认值：private
initRequest.accessControlList = @"public";
// 赋予被授权者读的权限。
initRequest.grantRead = @"grantRead";
// 赋予被授权者全部权限
initRequest.grantFullControl = @"grantFullControl";
[initRequest setFinishBlock:^(QCloudInitiateMultipartUploadResult* outputObject,
NSError *error) {
// 获取分块复制的 uploadId，后续的分块复制都需要这个 ID，请保存以备后续使用
self->uploadId = outputObject.uploadId;

}];

[[QCloudCOSXMLService defaultCOSXML] InitiateMultipartUpload:initRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let initRequest = QCloudInitiateMultipartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
initRequest.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 "video/xxx/movie.mp4"
initRequest.object = "exampleobject";

initRequest.setFinish { (result, error) in
if let result = result {
// 获取分块复制的 uploadId，后续的分块复制都需要这个 ID，请保存以备后续使用
self.uploadId = result.uploadId;
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().initiateMultipartUpload(initRequest);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

复制分块

功能说明

将其他对象复制为一个分块（Upload Part-Copy）。

示例代码

Objective-C

```
QCloudUploadPartCopyRequest* request = [[QCloudUploadPartCopyRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 源文件 URL 路径，可以通过 versionid 子资源指定历史版本
request.source = @"sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";

// 在初始化分块复制的响应中，会返回一个唯一的描述符（upload ID）
request.uploadID = uploadId;

// 标志当前分块的序号
request.partNumber = 1;

[request setFinishBlock:^(QCloudCopyObjectResult* result, NSError* error) {
QCloudMultipartInfo *part = [QCloudMultipartInfo new];

// 获取所复制分块的 etag
part.eTag = result.eTag;
part.partNumber = @"1";
// 保存起来用于最后完成复制时使用
self.parts=@[part];
}];

[[QCloudCOSXMLService defaultCOSXML]UploadPartCopy:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudUploadPartCopyRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.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 "video/xxx/movie.mp4"
req.object = "exampleobject";

// 源文件 URL 路径，可以通过 versionid 子资源指定历史版本
req.source = "sourcebucket-1250000000.cos.ap-guangzhou.myqcloud.com/sourceObject";
// 在初始化分块复制的响应中，会返回一个唯一的描述符（upload ID）
if let uploadId = self.uploadId {
req.uploadID = uploadId;
}

// 标志当前分块的序号
req.partNumber = 1;
req.setFinish { (result, error) in
if let result = result {
let mutipartInfo = QCloudMultipartInfo.init();
```

```
// 获取所复制分块的 etag
mutipartInfo.eTag = result.eTag;
mutipartInfo.partNumber = "1";
// 保存起来用于最后完成复制时使用
self.parts = [mutipartInfo];
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().uploadPartCopy(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询已复制的分块

功能说明

查询特定分块复制操作中的已复制的块（List Parts），该接口与分块上传的查询已上传的分块接口一致。

示例代码

Objective-C

```
QCloudListMultipartRequest* request = [QCloudListMultipartRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 在初始化分块复制的响应中，会返回一个唯一的描述符（upload ID）
request.uploadId = uploadId;

[request setFinishBlock:^(QCloudListPartsResult * _Nonnull result,
NSError * _Nonnull error) {

// 从 result 中获取已复制分块信息
// 用来表示每一个块的信息
NSArray<QCloudMultipartUploadPart*> *parts = result.parts;
}];

[[QCloudCOSXMLService defaultCOSXML] ListMultipart:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudListMultipartRequest.init();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
req.object = "exampleobject";
```

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

// 在初始化分块复制的响应中，会返回一个唯一的描述符（upload ID）
if let uploadId = self.uploadId {
req.uploadId = uploadId;
}
req.setFinish { (result, error) in
if let result = result {
// 所有已完成的分片
let parts = result.parts
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().listMultipart(req);
```

说明：
更多完整示例，请前往 GitHub 查看。

完成分块复制

功能说明

完成整个文件的分块复制（Complete Multipart Upload），该接口与分块上传的完成分块上传接口一致。

示例代码

Objective-C

```
QCloudCompleteMultipartUploadRequest *completeRequst = [QCloudCompleteMultipartUploadRequest new];
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
completeRequst.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
completeRequst.bucket = @"examplebucket-1250000000";
// 本次要查询的分块复制的 uploadId，可从初始化分块复制的请求结果 QCloudInitiateMultipartUploadResult 中得到
completeRequst.uploadId = uploadId;
// 已复制分块的信息
QCloudCompleteMultipartUploadInfo *partInfo = [QCloudCompleteMultipartUploadInfo new];
NSMutableArray * parts = [self.parts mutableCopy];
// 对已复制的块进行排序
[parts sortUsingComparator:^NSComparisonResult(QCloudMultipartInfo* _Nonnull obj1,
QCloudMultipartInfo* _Nonnull obj2) {
int a = obj1.partNumber.intValue;
int b = obj2.partNumber.intValue;

if (a < b) {
return NSOrderedAscending;
} else {
return NSOrderedDescending;
}
}];
partInfo.parts = [parts copy];
completeRequst.parts = partInfo;
```

```
[completeRequst setFinishBlock:^(QCloudUploadObjectResult * _Nonnull result,
NSError * _Nonnull error) {
// 从 result 中获取复制结果
}];

[[QCloudCOSXMLService defaultCOSXML] CompleteMultipartUpload:completeRequst];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let complete = QCloudCompleteMultipartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
complete.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 "video/xxx/movie.mp4"
complete.object = "exampleobject";

// 本次要查询的分块复制的 uploadId，可从初始化分块复制的请求结果
// QCloudInitiateMultipartUploadResult 中得到
complete.uploadId = "exampleUploadId";
if let uploadId = self.uploadId {
complete.uploadId = uploadId;
}

// 已复制分块的信息
let completeInfo = QCloudCompleteMultipartUploadInfo.init();
if self.parts == nil {
print("没有要完成的分块");
return;
}
if self.parts != nil {
completeInfo.parts = self.parts ?? [];
}

complete.parts = completeInfo;
complete.setFinish { (result, error) in
// 从 result 中获取复制结果
}
QCloudCOSXMLService.defaultCOSXML().completeMultipartUpload(complete);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

终止分块复制

功能说明

终止一个分块复制操作并删除已复制的块（Abort Multipart Upload），该接口与分块上传的终止分块上传接口一致。

示例代码

Objective-C

```
QCloudAbortMultipfartUploadRequest *abortRequest = [QCloudAbortMultipfartUploadRequest new];
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
abortRequest.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
abortRequest.bucket = @"examplebucket-1250000000";
// 本次要终止的分块复制的 uploadId
// 可从初始化分块复制的请求结果 QCloudInitiateMultipartUploadResult 中得到
abortRequest.uploadId = @"exampleUploadId";
[abortRequest setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary * result = (NSDictionary *)outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML]AbortMultipfartUpload:abortRequest];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let abort = QCloudAbortMultipfartUploadRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
abort.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 "video/xxx/movie.mp4"
abort.object = "exampleobject";

// 本次要查询的分块复制的 uploadId，可从初始化分块复制的请求结果
// QCloudInitiateMultipartUploadResult 中得到
abort.uploadId = self.uploadId!;

abort.finishBlock = {(result,error)in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().abortMultipfartUpload(abort);
```

说明：
更多完整示例，请前往 GitHub 查看。

检索对象内容

最近更新时间：2022-08-25 09:05:56

简介

本文档提供关于检索对象内容操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
SELECT Object Content	检索对象内容	从指定对象（CSV 格式或者 JSON 格式）中检索内容

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

检索对象内容

功能说明

COS Select 支持检索以下格式的对象数据：

- CSV 格式：对象以 CSV 格式存储，并以固定的分隔符划分。
- JSON 格式：对象以 JSON 格式存储，可以是 JSON 文件或者 JSON 列表。

> 注意：
>
> - 使用 COS Select，您必须具有 cos:GetObject 的授权。
> - CSV、JSON 对象需要以 UTF-8 格式编码。
> - COS Select 支持检索 GZIP 或者 BZIP2 压缩的 CSV、JSON 对象。
> - COS Select 支持检索 SSE-COS 加密的 CSV、JSON 对象。

示例代码

```
QCloudSelectObjectContentRequest *request = [QCloudSelectObjectContentRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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"
request.object = @"exampleobject";
request.selectType = @"2";
// 选择文件 配置
QCloudSelectObjectContentConfig *config= [QCloudSelectObjectContentConfig new];
/**SQL 表达式，代表您需要发起的检索操作。例如SELECT s._1 FROM COSObject s。
这个表达式可以从 CSV 格式的对象中检索第一列内容。有关 SQL 表达式的详细介绍，
请参见 (Select)[https://cloud.tencent.com/document/product/436/37636] 命令
*/
config.express = @"select * from COSObject";
/**
表达式类型，该项为扩展项，目前只支持 SQL 表达式，仅支持 SQL 参数
*/
config.expressionType = QCloudExpressionTypeSQL;
/**
描述待检索对象的格式
*/
QCloudInputSerialization *inputS = [QCloudInputSerialization new];
inputS.compressionType = QCloudCOSXMLCompressionTypeNONE;
/**
描述在JSON对象格式下所需的文件参数。
```

```
*/
QCloudSerializationJSON *inputJson = [QCloudSerializationJSON new];
/**
SON 文件的类型：
DOCUMENT 表示 JSON 文件仅包含一个独立的 JSON 对象，且该对象可以被切割成多行
LINES 表示 JSON 对象中的每一行包含了一个独立的 JSON 对象
合法值为 DOCUMENT 、LINES
*/
inputJson.type = QCloudInputJSONFileTypeDocument;
inputS.serializationJSON = inputJson;
config.inputSerialization = inputS;
/**
描述检索结果的输出格式
*/
QCloudOutputSerialization *outputS = [QCloudOutputSerialization new];

QCloudSerializationJSON *outputJson = [QCloudSerializationJSON new];
/**
将输出结果中的记录分隔为不同行的字符，默认通过\n进行分隔。您可以指定任意8进制字符，
如逗号、分号、Tab 等。该参数最多支持2个字节，即您可以输入\r\n这类格式的分隔符。默认值为\n
*/
outputJson.outputRecordDelimiter = @"\n";
/**
描述在JSON对象格式下所需的文件参数。
*/
outputS.serializationJSON = outputJson;

config.outputSerialization = outputS;
/**
是否需要返回查询进度 QueryProgress 信息，如果选中 COS Select 将周期性返回查询进度
*/
QCloudRequestProgress *requestProgress = [QCloudRequestProgress new];
requestProgress.enabled = @"FALSE";
config.requestProgress =requestProgress;
request.selectObjectContentConfig = config;
/**
文件存储在本地的路径
*/
request.downloadingURL = [NSURL fileURLWithPath:QCloudFileInSubPath(@"test", @"2.json")];
[request setDownProcessBlock:^(int64_t bytesDownload, int64_t totalBytesDownload, int64_t totalBytesExpectedToDownload) {
NSLog(@"       DOWN [Total]%lld [Downloaded]%lld [Download]%lld", totalBytesExpectedToDownload, totalBytesDownload, bytesDownload);
}];

[request setFinishBlock:^(id _Nonnull result, NSError * _Nonnull error) {
NSLog(@"result = %@",result);
}];
[[QCloudCOSXMLService defaultCOSXML] SelectObjectContent:request];
```

说明：

更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudSelectObjectContentRequest.init();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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"
request.object = "exampleobject";
//代表这一接口的版本信息
request.selectType = "2";
// 选择文件 配置
let config = QCloudSelectObjectContentConfig();
/**SQL 表达式，代表您需要发起的检索操作。例如SELECT s._1 FROM COSObject s。
这个表达式可以从 CSV 格式的对象中检索第一列内容。有关 SQL 表达式的详细介绍，
请参见 (Select)[https://cloud.tencent.com/document/product/436/37636] 命令
*/

config.express = "Select * from COSObject";
/**
表达式类型，该项为扩展项，目前只支持 SQL 表达式，仅支持 SQL 参数
*/
config.expressionType = .SQL;

/**
描述待检索对象的格式
*/
let inputS = QCloudInputSerialization();
inputS.compressionType = .NONE;
/**
描述在JSON对象格式下所需的文件参数。
*/
let inputJson = QCloudSerializationJSON.init();
/**
SON 文件的类型：
DOCUMENT 表示 JSON 文件仅包含一个独立的 JSON 对象，且该对象可以被切割成多行
LINES 表示 JSON 对象中的每一行包含了一个独立的 JSON 对象
合法值为 DOCUMENT 、LINES
*/
inputJson.type = .document;

inputS.serializationJSON = inputJson;

config.inputSerialization = inputS;

/**
描述检索结果的输出格式
*/
let outputS = QCloudOutputSerialization.init();

let outputJson = QCloudSerializationJSON.init();
/**
将输出结果中的记录分隔为不同行的字符，默认通过\n进行分隔。您可以指定任意8进制字符，
如逗号、分号、Tab 等。该参数最多支持2个字节，即您可以输入\r\n这类格式的分隔符。默认值为\n
*/
outputJson.outputRecordDelimiter = "\n";
outputS.serializationJSON = outputJson;

config.outputSerialization = outputS;
```

```
/**
是否需要返回查询进度 QueryProgress 信息，如果选中 COS Select 将周期性返回查询进度
*/
let requestProgress = QCloudRequestProgress.init();
requestProgress.enabled = "FALSE";
config.requestProgress = requestProgress;
request.selectObjectContentConfig = config;
//文件在本地的存储路径
request.downloadingURL = NSURL.fileURL(withPath: QCloudFileInSubPath("test", "2.json"));
request.finishBlock = {(result,error) in
if error != nil{
print(error!)
}else{
print(result!);
}
}

QCloudCOSXMLService.defaultCOSXML().selectObjectContent(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

检查对象是否存在

最近更新时间：2022-08-25 09:05:59

简介

本文档提供快捷查询存储桶中某个对象是否存在的示例代码。示例代码实际调用了 HEAD Object COS API，是该接口的简化版。

HEAD Object 除了检查对象是否存在，主要功能为返回对象元数据。如需查看包含了 HEAD Object 完整功能的 SDK 接口，请参见 查询对象元数据。

API	操作名	操作描述
HEAD Object	查询对象元数据	查询对象的元数据信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询对象元数据

功能说明

检查存储桶中是否存在某个对象。

示例代码

Objective-C

```
//存储桶名称，格式为 BucketName-APPID
NSString *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 "video/xxx/movie.mp4"
NSString *object = @"exampleobject";
[[QCloudCOSXMLService defaultCOSXML] doesObjectExistWithBucket:bucket object:object];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
// 存储桶名称，格式为 BucketName-APPID
let 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 "video/xxx/movie.mp4"
let object = "exampleobject";

QCloudCOSXMLService.defaultCOSXML().doesObjectExist(withBucket: bucket, object: object);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

版权所有：腾讯云计算（北京）有限责任公司

删除对象

最近更新时间：2022-08-25 09:06:03

简介

本文档提供关于对象的删除操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DELETE Object	删除单个对象	在存储桶中删除指定对象
DELETE Multiple Objects	删除多个对象	在存储桶中批量删除对象

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

删除对象

功能说明

删除指定的对象（DELETE Object）。

示例代码一：删除单个对象

Objective-C

```
QCloudDeleteObjectRequest* deleteObjectRequest = [QCloudDeleteObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
deleteObjectRequest.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 "video/xxx/movie.mp4"
deleteObjectRequest.object = @"exampleobject";

[deleteObjectRequest setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] DeleteObject:deleteObjectRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let deleteObject = QCloudDeleteObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
deleteObject.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 "video/xxx/movie.mp4"
deleteObject.object = "exampleobject";
```

版权所有：腾讯云计算（北京）有限责任公司

```
deleteObject.finishBlock = {(result,error)in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteObject(deleteObject);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：删除目录

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";
// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

//要删除的目录名称：带'/'
request.prefix = @"prefix";

[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
if(!error){
NSMutableArray *deleteInfosArr = [NSMutableArray array];
for (QCloudBucketContents *content in result.contents) {
QCloudDeleteMultipleObjectRequest *delteRequest = [QCloudDeleteMultipleObjectRequest new];
delteRequest.bucket = request.bucket;

QCloudDeleteObjectInfo *object = [QCloudDeleteObjectInfo new];
object.key = content.key;
[deleteInfosArr addObject:object];
}

QCloudDeleteInfo *deleteInfos = [QCloudDeleteInfo new];
deleteInfos.objects = [deleteInfosArr copy];

QCloudDeleteMultipleObjectRequest *delteRequest = [QCloudDeleteMultipleObjectRequest new];
delteRequest.bucket = @"examplebucket-1250000000";
delteRequest.deleteObjects = deleteInfos;
[delteRequest setFinishBlock:^(QCloudDeleteResult *outputObject, NSError *error) {
NSLog(@"outputObject = %@",outputObject);
}];

[[QCloudCOSXMLService defaultCOSXML] DeleteMultipleObject:delteRequest];

}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketReq.bucket = "examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;

//要删除的目录名称：带'/'
getBucketReq.prefix = "dir/";

getBucketReq.setFinish { (result, error) in
if let result = result {
let contents = result.contents;
let infos = NSMutableArray.init();
for content in contents {
let info = QCloudDeleteObjectInfo.init();
info.key = content.key;
infos.add(info);
}
let mutipleDel = QCloudDeleteMultipleObjectRequest.init();
// 要删除的文件集合
let deleteInfos = QCloudDeleteInfo.init();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
mutipleDel.bucket = "examplebucket-1250000000";

deleteInfos.objects = infos as! [QCloudDeleteObjectInfo];

// 布尔值，这个值决定了是否启动 Quiet 模式：
// true：启动 Quiet 模式
// false：启动 Verbose 模式
// 默认值为 False
deleteInfos.quiet = false;

// 封装了需要批量删除的多个对象的信息
mutipleDel.deleteObjects = deleteInfos;

mutipleDel.setFinish { (result, error) in
if let result = result {
let deleted = result.deletedObjects
let failed = result.deletedFailedObjects
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteMultipleObject(mutipleDel);
} else {
print(error!);
}
```

```
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除多个对象

功能说明

批量删除多个对象（DELETE Multiple Objects）。

示例代码一：批量删除对象

Objective-C

```
QCloudDeleteMultipleObjectRequest* delteRequest = [QCloudDeleteMultipleObjectRequest new];
delteRequest.bucket = @"examplebucket-1250000000";

// 要删除的单个文件
QCloudDeleteObjectInfo* deletedObject0 = [QCloudDeleteObjectInfo new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
deletedObject0.key = @"exampleobject";

// 要删除的文件集合
QCloudDeleteInfo* deleteInfo = [QCloudDeleteInfo new];

// 布尔值，这个值决定了是否启动 Quiet 模式：
// true：启动 Quiet 模式
// false：启动 Verbose 模式
// 默认值为 False
deleteInfo.quiet = NO;

// 存放需要删除对象信息的数组
deleteInfo.objects = @[deletedObject0];

// 封装了需要批量删除的多个对象的信息
delteRequest.deleteObjects = deleteInfo;

[delteRequest setFinishBlock:^(QCloudDeleteResult* outputObject,
NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息

}];

[[QCloudCOSXMLService defaultCOSXML] DeleteMultipleObject:delteRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let mutipleDel = QCloudDeleteMultipleObjectRequest.init();
```

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
mutipleDel.bucket = "examplebucket-1250000000";

// 要删除的单个文件
let info1 = QCloudDeleteObjectInfo.init();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
info1.key = "exampleobject";

let info2 = QCloudDeleteObjectInfo.init();

// 要删除的文件集合
let deleteInfos = QCloudDeleteInfo.init();

// 存放需要删除对象信息的数组
deleteInfos.objects = [info1,info2];

// 布尔值，这个值决定了是否启动 Quiet 模式：
// true：启动 Quiet 模式
// false：启动 Verbose 模式
// 默认值为 False
deleteInfos.quiet = false;

// 封装了需要批量删除的多个对象的信息
mutipleDel.deleteObjects = deleteInfos;

mutipleDel.setFinish { (result, error) in
if let result = result {
let deleted = result.deletedObjects
let failed = result.deletedFailedObjects
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteMultipleObject(mutipleDel);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

示例代码二：删除带指定前缀的对象

Objective-C

```
QCloudGetBucketRequest* request = [QCloudGetBucketRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";
// 单次返回的最大条目数量，默认1000
request.maxKeys = 100;

//如果要删除指定前缀的文件:prefix为文件名前缀
request.prefix = @"prefix";

[request setFinishBlock:^(QCloudListBucketResult * result, NSError* error) {
if(!error){
NSMutableArray *deleteInfosArr = [NSMutableArray array];
```

```
for (QCloudBucketContents *content in result.contents) {
QCloudDeleteMultipleObjectRequest *delteRequest = [QCloudDeleteMultipleObjectRequest new];
delteRequest.bucket = request.bucket;

QCloudDeleteObjectInfo *object = [QCloudDeleteObjectInfo new];
object.key = content.key;
[deleteInfosArr addObject:object];
}

QCloudDeleteInfo *deleteInfos = [QCloudDeleteInfo new];
deleteInfos.objects = [deleteInfosArr copy];

QCloudDeleteMultipleObjectRequest *delteRequest = [QCloudDeleteMultipleObjectRequest new];
delteRequest.bucket = @"examplebucket-1250000000";
delteRequest.deleteObjects = deleteInfos;
[delteRequest setFinishBlock:^(QCloudDeleteResult *outputObject, NSError *error) {
NSLog(@"outputObject = %@",outputObject);
}];

[[QCloudCOSXMLService defaultCOSXML] DeleteMultipleObject:delteRequest];

}
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucket:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReq = QCloudGetBucketRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketReq.bucket = "examplebucket-1250000000";

// 单次返回的最大条目数量，默认1000
getBucketReq.maxKeys = 100;

//如果要删除指定前缀的文件:prefix为文件名前缀
getBucketReq.prefix = "dir/";


getBucketReq.setFinish { (result, error) in
if let result = result {
let contents = result.contents;
let infos = NSMutableArray.init();
for content in contents {
let info = QCloudDeleteObjectInfo.init();
info.key = content.key;
infos.add(info);
}
let mutipleDel = QCloudDeleteMultipleObjectRequest.init();
// 要删除的文件集合
```

```
let deleteInfos = QCloudDeleteInfo.init();
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
mutipleDel.bucket = "examplebucket-1250000000";

deleteInfos.objects = infos as! [QCloudDeleteObjectInfo];

// 布尔值，这个值决定了是否启动 Quiet 模式：
// true：启动 Quiet 模式
// false：启动 Verbose 模式
// 默认值为 False
deleteInfos.quiet = false;

// 封装了需要批量删除的多个对象的信息
mutipleDel.deleteObjects = deleteInfos;

mutipleDel.setFinish { (result, error) in
if let result = result {
let deleted = result.deletedObjects
let failed = result.deletedFailedObjects
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteMultipleObject(mutipleDel);
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucket(getBucketReq);
```

说明：

更多完整示例，请前往 GitHub 查看。

恢复归档对象

最近更新时间：2022-08-25 09:06:08

简介

本文档提供关于恢复归档对象操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
POST Object restore	恢复归档对象	将归档类型的对象取回访问

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

恢复归档对象

功能说明

将归档类型的对象取回访问（POST Object restore）。

示例代码

Objective-C

```
QCloudPostObjectRestoreRequest *req = [QCloudPostObjectRestoreRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.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 "video/xxx/movie.mp4"
req.object = @"exampleobject";

// 设置临时副本的过期时间
req.restoreRequest.days = 10;

// 复原的过程类型配置信息
req.restoreRequest.CASJobParameters.tier = QCloudCASTierStandard;

[req setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] PostObjectRestore:req];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let restore = QCloudPostObjectRestoreRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
restore.bucket = "examplebucket-1250000000";
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
restore.object = "exampleobject";

// 设置临时副本的过期时间
restore.restoreRequest.days = 10;

// 复原的过程类型配置信息
restore.restoreRequest.casJobParameters.tier = .standard;
restore.finishBlock = {(result,error)in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().postObjectRestore(restore);
```

说明：

更多完整示例，请前往 GitHub 查看。

查询对象元数据

最近更新时间：2022-08-25 09:06:11

简介

本文档提供关于查询对象元数据操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
HEAD Object	查询对象元数据	查询对象的元数据信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询对象元数据

功能说明

查询 Object 的 Meta 信息。

示例代码

Objective-C

```
QCloudHeadObjectRequest* headerRequest = [QCloudHeadObjectRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
headerRequest.object = @"exampleobject";

// versionId 当启用版本控制时，指定要查询的版本 ID，如不指定则查询对象的最新版本
headerRequest.versionID = @"versionID";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
headerRequest.bucket = @"examplebucket-1250000000";

[headerRequest setFinishBlock:^(NSDictionary* result, NSError *error) {
// result 返回具体信息
// 获取文件crc64
NSString * crc64 = [[outputObject __originHTTPURLResponse__].allHeaderFields valueForKey:@"x-cos-hash-crc64ecma"];
}];

[[QCloudCOSXMLService defaultCOSXML] HeadObject:headerRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let headObject = QCloudHeadObjectRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
headObject.bucket = "examplebucket-1250000000";

// versionId 当启用版本控制时，指定要查询的版本 ID，如不指定则查询对象的最新版本
headObject.versionID = "versionID";
```

```
// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
headObject.object = "exampleobject";
headObject.finishBlock = {(result,error) in
if let result = result {
// 获取文件crc64
let crc64 = result?.__originHTTPURLResponse__.allHeaderFields["x-cos-hash-crc64ecma"];

} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().headObject(headObject);
```

说明：

更多完整示例，请前往 GitHub 查看。

服务端加密

最近更新时间：2022-08-25 09:06:14

简介

本文档提供关于如何使用在上传对象时开启服务端加密。服务端加密的密钥分为三种：

- COS 托管加密密钥
- 客户提供的加密密钥
- KMS 托管加密密钥

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

使用 COS 托管加密密钥的服务端加密（SSE-COS）保护数据

功能说明

由腾讯云 COS 托管主密钥和管理数据。COS 会帮助您在数据写入数据中心时自动加密，并在您取用该数据时自动解密。目前支持使用 COS 主密钥对数据进行 AES-256 加密。

示例代码

Objective-C

```
QCloudCOSXMLUploadObjectRequest *request = [QCloudCOSXMLUploadObjectRequest new];
[request setCOSServerSideEncyption];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
request.setCOSServerSideEncyption();
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

使用客户提供的加密密钥的服务端加密（SSE-C）保护数据

功能说明

加密密钥由用户自己提供，用户在上传对象时，COS 将使用用户提供的加密密钥对用户的数据进行 AES-256 加密。

> 注意：
> - 该加密所运行的服务需要使用 HTTPS 请求。
> - 用户需要提供一个32字节的字符串作为密钥，支持数字、字母、字符的组合，不支持中文。
> - 如果上传文件时设置了密钥加密，那么在使用 GET（下载）、HEAD（查询）源对象时也需要在请求中带上相同的密钥，才能正常响应。

示例代码

Objective-C

```
QCloudCOSXMLUploadObjectRequest *request = [QCloudCOSXMLUploadObjectRequest new];
NSString *customKey = @"123456qwertyuioplkjhgfdsazxcvbnm";
```

```
[request setCOSServerSideEncyptionWithCustomerKey:customKey];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let customKey = "123456qwertyuioplkjhgfdsazxcvbnm";
request.setCOSServerSideEncyptionWithCustomerKey(customKey);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

使用 KMS 托管加密密钥的服务端加密（SSE-KMS）保护数据

功能说明

SSE-KMS 加密即使用 KMS 托管密钥的服务端加密。KMS 是腾讯云推出的一款安全管理类服务，使用经过第三方认证的硬件安全模块 HSM（Hardware Security Module）来生成和保护密钥。它能够帮助用户轻松创建和管理密钥，满足用户多应用多业务的密钥管理需求以及满足监管和合规要求。关于如何开通 KMS 服务请参考：服务端加密概述。

示例代码

Objective-C

```
QCloudCOSXMLUploadObjectRequest *request = [QCloudCOSXMLUploadObjectRequest new];
NSString *customKey = @"123456qwertyuioplkjhgfdsazxcvbnm";
NSString *arrJsonStr = @"{\"key\":\"value\"}";
[request setCOSServerSideEncyptionWithKMSCustomKey:customKey jsonStr:arrJsonStr];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let customKey = "123456qwertyuioplkjhgfdsazxcvbnm";
let arrJsonStr = "{\"key\":\"value\"}";
self.advancedRequest?.setCOSServerSideEncyptionWithKMSCustomKey(customKey, jsonStr: arrJsonStr);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

对象访问 URL

最近更新时间：2022-08-25 09:06:18

简介

本文档提供获取对象访问 URL 的代码示例。

> 说明：
>
> - 查询对象访问的 URL，该接口不会判断对象是否真实存在。
> - 如果您的文件是私有读权限，那么本接口生成的 URL 不能直接用于访问资源，您可以通过 生成预签名链接 文档来生成 URL。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

获取对象访问 URL

示例代码一：获取对象访问 URL

Objective-C

```
QCloudGetPresignedURLRequest* getPresignedURLRequest = [[QCloudGetPresignedURLRequest alloc] init];

// 生成的链接不携带签名
getPresignedURLRequest.isUseSignature = false:

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresignedURLRequest.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 "video/xxx/movie.mp4"
getPresignedURLRequest.object = @"exampleobject";

[getPresignedURLRequest setFinishBlock:^(QCloudGetPresignedURLResult * _Nonnull result,
NSError * _Nonnull error) {
// 预签名 URL
NSString* presignedURL = result.presienedURL;

}];

[[QCloudCOSXMLService defaultCOSXML] getPresignedURL:getPresignedURLRequest];
```

> 说明：
>
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getPresign = QCloudGetPresignedURLRequest.init();

// 生成的链接不携带签名
getPresign.isUseSignature = false;
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresign.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 "video/xxx/movie.mp4"
```

版权所有：腾讯云计算（北京）有限责任公司

```
getPresign.object = "exampleobject";

getPresign.setFinish { (result, error) in
if let result = result {
let url = result.presienedURL
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getPresignedURL(getPresign);
```

说明：
更多完整示例，请前往 GitHub 查看。

生成预签名链接

最近更新时间：2022-08-25 09:06:22

简介

本文档提供关于生成对象预签名链接的示例代码。

> 说明：
>
> - 建议用户使用临时密钥生成预签名，通过临时授权的方式进一步提高预签名上传、下载等请求的安全性。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来生成预签名，建议永久密钥的权限范围仅限于上传或下载操作，以规避风险。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

生成对象预签名链接

示例代码一：生成预签名上传链接

Objective-C

```
QCloudGetPresignedURLRequest* getPresignedURLRequest = [[QCloudGetPresignedURLRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresignedURLRequest.bucket = @"examplebucket-1250000000";

// 使用预签名 URL 的请求的 HTTP 方法。有效值（大小写敏感）为：@"GET"、@"PUT"、@"POST"、@"DELETE"
getPresignedURLRequest.HTTPMethod = @"PUT";

// 获取预签名函数，默认签入Header Host；您也可以选择不签入Header Host，但可能导致请求失败或安全漏洞
getPresignedURLRequest.signHost = YES;

// http 请求参数，传入的请求参数需与实际请求相同，能够防止用户篡改此HTTP请求的参数
getPresignedURLRequest.requestParameters = @{@"param1":@"value1",@"param1":@"value1"};

// http 请求头部，传入的请求头部需包含在实际请求中，能够防止用户篡改签入此处的HTTP请求头部
getPresignedURLRequest.requestHeaders = @{@"param1":@"value1",@"param1":@"value1"};

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
getPresignedURLRequest.object = @"exampleobject";

[getPresignedURLRequest setFinishBlock:^(QCloudGetPresignedURLResult * _Nonnull result,
NSError * _Nonnull error) {
// 预签名 URL
NSString* presignedURL = result.presienedURL;
// 使用预签名链接进行上传文件
NSMutableURLRequest *request = [[NSMutableURLRequest alloc] initWithURL:[NSURL URLWithString:result.presienedURL]];
// 指定HTTPMethod 为PUT
request.HTTPMethod = @"PUT";
// fromData 为需要上传的文件
[[[NSURLSession sharedSession]
uploadTaskWithRequest:request fromData:[@"testtest" dataUsingEncoding:NSUTF8StringEncoding] completionHandler:^(NSData * _Nullable d
ata, NSURLResponse * _Nullable response, NSError * _Nullable error) {
// response中查看上传结果
```

```
}]resume];

}];

[[QCloudCOSXMLService defaultCOSXML] getPresignedURL:getPresignedURLRequest];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getPresign = QCloudGetPresignedURLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresign.bucket = "examplebucket-1250000000" ;

// 使用预签名 URL 的请求的 HTTP 方法。有效值（大小写敏感）为：
// @"GET"、@"PUT"、@"POST"、@"DELETE"
getPresign.httpMethod = "PUT";

// 获取预签名函数，默认签入Header Host；您也可以选择不签入Header Host，但可能导致请求失败或安全漏洞
getPresign.signHost = YES;

// http 请求参数，传入的请求参数需与实际请求相同，能够防止用户篡改此HTTP请求的参数
getPresign.requestParameters = {"param1":"value1","param1":"value1"};

// http 请求头部，传入的请求头部需包含在实际请求中，能够防止用户篡改签入此处的HTTP请求头部
getPresign.requestHeaders = {"param1":"value1","param1":"value1"};

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
getPresign.object = "exampleobject";
getPresign.setFinish { (result, error) in
if let result = result {
let url = result.presienedURL
// 使用预签名链接进行上传文件
let request = NSMutableURLRequest.init(url: NSURL.init(string:url) as! URL);
// 指定HTTPMethod 为PUT
request.httpMethod = "PUT";
// fromData 为需要上传的文件
URLSession.shared.uploadTask(with: request, from: "testtest".data(using: String.Encoding.utf8)) { data, res, err in
// response中查看上传结果
}.resume()
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getPresignedURL(getPresign);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

示例代码二：生成预签名下载链接

Objective-C

```
QCloudGetPresignedURLRequest* getPresignedURLRequest = [[QCloudGetPresignedURLRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresignedURLRequest.bucket = @"examplebucket-1250000000";

// 使用预签名 URL 的请求的 HTTP 方法。有效值（大小写敏感）为：@"GET"、@"PUT"、@"POST"、@"DELETE"
getPresignedURLRequest.HTTPMethod = @"GET";

// 获取预签名函数，默认签入Header Host；您也可以选择不签入Header Host，但可能导致请求失败或安全漏洞
getPresignedURLRequest.signHost = YES;

// http 请求参数，传入的请求参数需与实际请求相同，能够防止用户篡改此HTTP请求的参数
getPresignedURLRequest.requestParameters = @{@"param1":@"value1",@"param1":@"value1"};

// http 请求头部，传入的请求头部需包含在实际请求中，能够防止用户篡改签入此处的HTTP请求头部
getPresignedURLRequest.requestHeaders = @{@"param1":@"value1",@"param1":@"value1"};

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
getPresignedURLRequest.object = @"exampleobject";

[getPresignedURLRequest setFinishBlock:^(QCloudGetPresignedURLResult * _Nonnull result,
NSError * _Nonnull error) {
// 预签名 URL
NSString* presignedURL = result.presienedURL;
// 使用预签名链接进行下载文件
NSMutableURLRequest *request = [[NSMutableURLRequest alloc] initWithURL:[NSURL URLWithString:presignedURL]];
// 指定HTTPMethod为GET
request.HTTPMethod = @"GET";
[[[NSURLSession sharedSession]
downloadTaskWithRequest:request
completionHandler:^(NSURL *_Nullable location, NSURLResponse *_Nullable response, NSError *_Nullable error) {
// location 下载成功后的本地文件路径
}] resume];
}];

[[QCloudCOSXMLService defaultCOSXML] getPresignedURL:getPresignedURLRequest];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getPresign = QCloudGetPresignedURLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getPresign.bucket = "examplebucket-1250000000" ;

// 使用预签名 URL 的请求的 HTTP 方法。有效值（大小写敏感）为：
// @"GET"、@"PUT"、@"POST"、@"DELETE"
getPresign.httpMethod = "GET";

// 获取预签名函数，默认签入Header Host；您也可以选择不签入Header Host，但可能导致请求失败或安全漏洞
getPresignedURLRequest.signHost = YES;
```

版权所有：腾讯云计算（北京）有限责任公司

```
// http 请求参数，传入的请求参数需与实际请求相同，能够防止用户篡改此HTTP请求的参数
getPresignedURLRequest.requestParameters = {"param1":"value1","param1":"value1"};

// http 请求头部，传入的请求头部需包含在实际请求中，能够防止用户篡改签入此处的HTTP请求头部
getPresignedURLRequest.requestHeaders = {"param1":"value1","param1":"value1"};

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
getPresign.object = "exampleobject";
getPresign.setFinish { (result, error) in
if let result = result {
let url = result.presienedURL
// 使用预签名链接进行下载文件
let request = NSMutableURLRequest.init(url: NSURL.init(string: url) as! URL);
// 指定HTTPMethod为GET
request.httpMethod = "GET";
URLSession.shared.downloadTask(with: request) { location, response, error in
// location 下载成功后的本地文件路径
}.resume();
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getPresignedURL(getPresign);
```

说明：
更多完整示例，请前往 GitHub 查看。

预请求跨域配置

最近更新时间：2022-08-25 09:06:25

简介

本文档提供关于预请求跨域配置操作相关的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
Options Object	预请求跨域配置	用预请求来确认是否可以发送真正的跨域请求

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

预请求跨域配置

功能说明

获取预请求跨域配置（Options Object）。

示例代码

Objective-C

```
QCloudOptionsObjectRequest* request = [[QCloudOptionsObjectRequest alloc] init];

// 存储桶名称，格式：BucketName-APPID
request.bucket =@"examplebucket-1250000000";

// 模拟跨域访问的请求来源域名，请求 method，请求 host
request.origin = @"http://cloud.tencent.com";
request.accessControlRequestMethod = @"GET";
request.accessControlRequestHeaders = @"host";

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
request.object = @"exampleobject";

[request setFinishBlock:^(id outputObject, NSError* error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary* info = (NSDictionary *) outputObject;

}];

[[QCloudCOSXMLService defaultCOSXML] OptionsObject:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let optionsObject = QCloudOptionsObjectRequest.init();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
optionsObject.object = "exampleobject";
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 模拟跨域访问的请求来源域名，请求 method，请求头部
optionsObject.origin = "http://www.qcloud.com";
optionsObject.accessControlRequestMethod = "GET";
optionsObject.accessControlRequestHeaders = "origin";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
optionsObject.bucket = "examplebucket-1250000000";

optionsObject.finishBlock = {(result,error) in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
}
}
QCloudCOSXMLService.defaultCOSXML().optionsObject(optionsObject);
```

说明：
更多完整示例，请前往 GitHub 查看。

异地容灾
版本控制

最近更新时间：2022-08-25 09:06:31

简介

本文档提供关于版本控制的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket versioning	设置版本控制	设置存储桶的版本控制功能
GET Bucket versioning	查询版本控制	查询存储桶的版本控制信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置版本控制

功能说明

设置指定存储桶的版本控制功能。开启版本控制功能后，只能暂停，不能关闭。

示例代码

Objective-C

```
// 开启版本控制
QCloudPutBucketVersioningRequest* request = [[QCloudPutBucketVersioningRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket =@"examplebucket-1250000000";

// 说明版本控制的具体信息
QCloudBucketVersioningConfiguration* versioningConfiguration =
[[QCloudBucketVersioningConfiguration alloc] init];

request.configuration = versioningConfiguration;

// 说明版本是否开启，枚举值：QCloudCOSBucketVersioningStatusEnabled、
// QCloudCOSBucketVersioningStatusSuspended
versioningConfiguration.status = QCloudCOSBucketVersioningStatusEnabled;

[request setFinishBlock:^(id outputObject, NSError* error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketVersioning:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
// 开启版本控制
let putBucketVersioning = QCloudPutBucketVersioningRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketVersioning.bucket = "examplebucket-1250000000";

// 说明版本控制的具体信息
let config = QCloudBucketVersioningConfiguration.init();

// 说明版本是否开启，枚举值：Suspended、Enabled
config.status = .enabled;

putBucketVersioning.configuration = config;

putBucketVersioning.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketVersioning(putBucketVersioning);
```

说明：
更多完整示例，请前往 GitHub 查看。

查询版本控制

功能说明

查询指定存储桶的版本控制信息。

- 获取存储桶版本控制的状态，需要有该存储桶的读权限。
- 有三种版本控制状态：未启用版本控制、启用版本控制和暂停版本控制。

示例代码

Objective-C

```
QCloudGetBucketVersioningRequest* request =
[[QCloudGetBucketVersioningRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(QCloudBucketVersioningConfiguration* result,
NSError* error) {
// 获取多版本状态
QCloudCOSBucketVersioningStatus * status = result.status;
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucketVersioning:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketVersioning = QCloudGetBucketVersioningRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketVersioning.bucket = "examplebucket-1250000000";

getBucketVersioning.setFinish { (config, error) in
if let config = config {
// 多版本状态
let status = config.status
} else {
print(error!);
}

}
QCloudCOSXMLService.defaultCOSXML().getBucketVersioning(getBucketVersioning);
```

说明：
更多完整示例，请前往 GitHub 查看。

跨地域复制

最近更新时间：2022-08-25 09:06:34

简介

本文档提供关于跨地域复制的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket replication	设置跨地域复制	设置存储桶的跨地域复制规则
GET Bucket replication	查询跨地域复制	查询存储桶的跨地域复制规则
DELETE Bucket replication	删除跨地域复制	删除存储桶的跨地域复制规则

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置跨地域复制

功能说明

设置指定存储桶的跨地域复制规则。

示例代码

Objective-C

```
QCloudPutBucketReplicationRequest* request = [[QCloudPutBucketReplicationRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

// 说明所有跨地域配置信息
QCloudBucketReplicationConfiguation* replConfiguration =
[[QCloudBucketReplicationConfiguation alloc] init];

// 发起者身份标示
replConfiguration.role = @"qcs::cam::uin/100000000001:uin/100000000001";

// 具体配置信息
QCloudBucketReplicationRule* rule = [[QCloudBucketReplicationRule alloc] init];

// 用来标注具体 Rule 的名称
rule.identifier = @"identifier";
rule.status = QCloudCOSXMLStatusEnabled;

// 资源标识符
QCloudBucketReplicationDestination* destination = [[QCloudBucketReplicationDestination alloc] init];
NSString* destinationBucket = @"destinationbucket-1250000000";

// 目标存储桶所在地域
NSString* region = @"ap-beijing";
destination.bucket = [NSString stringWithFormat:@"qcs::cos:%@::%@",region,destinationBucket];

// 目标存储桶信息
rule.destination = destination;
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 前缀匹配策略，不可重叠，重叠返回错误。前缀匹配根目录为空
rule.prefix = @"prefix1";
replConfiguration.rule = @[rule];
request.configuation = replConfiguration;

[request setFinishBlock:^(id outputObject, NSError* error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketRelication:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let putBucketReplication = QCloudPutBucketReplicationRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketReplication.bucket = "examplebucket-1250000000";

// 说明所有跨地域配置信息
let config = QCloudBucketReplicationConfiguation.init();
config.role = "qcs::cam::uin/100000000001:uin/100000000001";

// 发起者身份标示
let rule = QCloudBucketReplicationRule.init();

// 用来标注具体 Rule 的名称
rule.identifier = "rule1";
// 规则开启状态，可选 .enabled, .disabled
rule.status = .enabled;

// 目标存储桶信息
let destination = QCloudBucketReplicationDestination.init();
let destinationBucket = "destinationbucket-1250000000";
let region = "ap-beijing";
destination.bucket = "qcs::cos:\(region)::\(destinationBucket)";
rule.destination = destination;

// 前缀匹配策略，不可重叠，重叠返回错误。前缀匹配根目录为空
rule.prefix = "dir/";

config.rule = [rule];

putBucketReplication.configuation = config;

putBucketReplication.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
```

```
}
QCloudCOSXMLService.defaultCOSXML().putBucketRelication(putBucketReplication);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询跨地域复制

功能说明

查询指定存储桶的跨地域复制规则。

示例代码

Objective-C

```
QCloudGetBucketReplicationRequest* request = [[QCloudGetBucketReplicationRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(QCloudBucketReplicationConfiguation* result,
NSError* error) {
// 具体配置信息，最多支持 1000 个，所有策略只能指向一个目标存储桶
NSArray *rules = result.rule;
}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketReplication:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketReplication = QCloudGetBucketReplicationRequest.init();
getBucketReplication.bucket = "examplebucket-1250000000";
getBucketReplication.setFinish { (config, error) in
if let config = config {
// 配置信息列表
let rule = config.rule
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketReplication(getBucketReplication);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除跨地域复制

功能说明

删除指定存储桶的跨地域复制规则。

示例代码

Objective-C

```
QCloudDeleteBucketReplicationRequest* request =
[[QCloudDeleteBucketReplicationRequest alloc] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError* error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketReplication:request];
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let deleteBucketReplication = QCloudDeleteBucketReplicationRequest.init();
deleteBucketReplication.bucket = "examplebucket-1250000000";
deleteBucketReplication.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().deleteBucketReplication(deleteBucketReplication);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

数据管理
生命周期

最近更新时间：2022-08-25 09:06:39

简介

本文档提供关于生命周期的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket lifecycle	设置生命周期	设置存储桶的生命周期管理的配置
GET Bucket lifecycle	查询生命周期	查询存储桶生命周期管理的配置
DELETE Bucket lifecycle	删除生命周期	删除存储桶生命周期管理的配置

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置生命周期

功能说明

设置指定存储桶的生命周期配置信息。

示例代码

Objective-C

```
QCloudPutBucketLifecycleRequest* request = [QCloudPutBucketLifecycleRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";
__block QCloudLifecycleConfiguration* lifecycleConfiguration =
[[QCloudLifecycleConfiguration alloc] init];

// 规则描述
QCloudLifecycleRule* rule = [[QCloudLifecycleRule alloc] init];

// 用于唯一地标识规则
rule.identifier = @"identifier";

// 指明规则是否启用，枚举值：Enabled，Disabled
rule.status = QCloudLifecycleStatueEnabled;

// Filter 用于描述规则影响的 Object 集合
QCloudLifecycleRuleFilter* filter = [[QCloudLifecycleRuleFilter alloc] init];

// 指定规则所适用的前缀。匹配前缀的对象受该规则影响，Prefix 最多只能有一个
filter.prefix = @"prefix1";

// Filter 用于描述规则影响的 Object 集合
rule.filter = filter;

// 规则转换属性，对象何时转换为 Standard_IA 或 Archive
QCloudLifecycleTransition* transition = [[QCloudLifecycleTransition alloc] init];
```

```
// 指明规则对应的动作在对象最后的修改日期过后多少天操作：
transition.days = 100;

// 指定 Object 转储到的目标存储类型，枚举值： STANDARD_IA，ARCHIVE
transition.storageClass = QCloudCOSStorageStandardIA;
rule.transition = transition;
request.lifeCycle = lifecycleConfiguration;

// 生命周期配置
request.lifeCycle.rules = @[rule];
[request setFinishBlock:^(id outputObject, NSError* error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] PutBucketLifecycle:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let putBucketLifecycleReq = QCloudPutBucketLifecycleRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketLifecycleReq.bucket = "examplebucket-1250000000";

let config = QCloudLifecycleConfiguration.init();

// 规则描述
let rule = QCloudLifecycleRule.init();

// 用于唯一地标识规则
rule.identifier = "swift";

// 指明规则是否启用，枚举值：Enabled，Disabled
rule.status = .enabled;

// Filter 用于描述规则影响的 Object 集合
let fileter = QCloudLifecycleRuleFilter.init();

// 指定规则所适用的前缀。匹配前缀的对象受该规则影响，Prefix 最多只能有一个
fileter.prefix = "0";

// Filter 用于描述规则影响的 Object 集合
rule.filter = fileter;

// 规则转换属性，对象何时转换为 Standard_IA 或 Archive
let transition = QCloudLifecycleTransition.init();

// 指明规则对应的动作在对象最后的修改日期过后多少天操作：
transition.days = 100;

// 指定 Object 转储到的目标存储类型，枚举值： STANDARD_IA，ARCHIVE
```

```
transition.storageClass = .standardIA;

rule.transition = transition;

putBucketLifecycleReq.lifeCycle = config;

// 生命周期配置
putBucketLifecycleReq.lifeCycle.rules = [rule];

putBucketLifecycleReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketLifecycle(putBucketLifecycleReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询生命周期

功能说明

查询存储桶的生命周期管理配置。

示例代码

Objective-C

```
QCloudGetBucketLifecycleRequest* request = [QCloudGetBucketLifecycleRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";
[request setFinishBlock:^(QCloudLifecycleConfiguration* result,NSError* error) {
// 可以从 result 中获取返回信息
// result.rules 规则描述集合的数组

}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketLifecycle:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketLifeCycle = QCloudGetBucketLifecycleRequest.init();
getBucketLifeCycle.bucket = "examplebucket-1250000000";
getBucketLifeCycle.setFinish { (config, error) in
if let config = config {
// 生命周期规则
let rules = config.rules
} else {
print(error!);
```

```
}

};
QCloudCOSXMLService.defaultCOSXML().getBucketLifecycle(getBucketLifeCycle);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除生命周期

功能说明

删除存储桶生命周期管理的配置。

示例代码

Objective-C

```
QCloudDeleteBucketLifeCycleRequest* request =
[[QCloudDeleteBucketLifeCycleRequest alloc ] init];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(QCloudLifecycleConfiguration* deleteResult, NSError* error) {
// 返回删除结果
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketLifeCycle:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let deleteBucketLifeCycle = QCloudDeleteBucketLifeCycleRequest.init();
deleteBucketLifeCycle.bucket = "examplebucket-1250000000";
deleteBucketLifeCycle.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
};
QCloudCOSXMLService.defaultCOSXML().deleteBucketLifeCycle(deleteBucketLifeCycle);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

版权所有：腾讯云计算（北京）有限责任公司

日志管理

最近更新时间：2022-08-25 09:06:42

简介

本文档提供关于日志管理的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket logging	设置日志管理	为源存储桶开启日志记录
GET Bucket logging	查询日志管理	查询源存储桶的日志配置信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置日志管理

功能说明

PUT Bucket logging 用于为源存储桶开启日志记录，将源存储桶的访问日志保存到指定的目标存储桶中。

示例代码

Objective-C

```
QCloudPutBucketLoggingRequest *request = [QCloudPutBucketLoggingRequest new];

// 说明日志记录配置的状态，如果无子节点信息则意为关闭日志记录
QCloudBucketLoggingStatus *status = [QCloudBucketLoggingStatus new];

// 存储桶 logging 设置的具体信息，主要是目标存储桶
QCloudLoggingEnabled *loggingEnabled = [QCloudLoggingEnabled new];

// 存放日志的目标存储桶，可以是同一个存储桶（但不推荐），或同一账户下、同一地域的存储桶
loggingEnabled.targetBucket = @"examplebucket-1250000000";

// 日志存放在目标存储桶的指定路径
loggingEnabled.targetPrefix = @"mylogs";

status.loggingEnabled = loggingEnabled;
request.bucketLoggingStatus = status;

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketLogging:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudPutBucketLoggingRequest.init();

// 说明日志记录配置的状态，如果无子节点信息则意为关闭日志记录
let status = QCloudBucketLoggingStatus.init();

// 存储桶 logging 设置的具体信息，主要是目标存储桶
let loggingEnabled = QCloudLoggingEnabled.init();

// 存放日志的目标存储桶，可以是同一个存储桶（但不推荐），或同一账户下、同一地域的存储桶
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
loggingEnabled.targetBucket = "examplebucket-1250000000";

// 日志存放在目标存储桶的指定路径
loggingEnabled.targetPrefix = "logs/";

status.loggingEnabled = loggingEnabled;
req.bucketLoggingStatus = status;

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
req.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().putBucketLogging(req);
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

查询日志管理

功能说明

GET Bucket logging 用于查询指定存储桶的日志配置信息。

示例代码

Objective-C

```
QCloudGetBucketLoggingRequest *getReq = [QCloudGetBucketLoggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";

[getReq setFinishBlock:^(QCloudBucketLoggingStatus * _Nonnull result,
NSError * _Nonnull error) {
// 日志配置信息
QCloudLoggingEnabled *loggingEnabled = result.loggingEnabled;
}];
[[QCloudCOSXMLService defaultCOSXML]GetBucketLogging:getReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudGetBucketLoggingRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
req.setFinish { (result, error) in
if let result = result {
// 日志配置信息
let enabled = result.loggingEnabled
} else {
print(error!);
}
};
QCloudCOSXMLService.defaultCOSXML().getBucketLogging(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

存储桶标签

最近更新时间：2022-08-25 09:06:47

简介

本文档提供关于存储桶标签的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket tagging	设置存储桶标签	为已存在的存储桶设置标签
GET Bucket tagging	查询存储桶标签	查询指定存储桶下已有的存储桶标签
DELETE Bucket tagging	删除存储桶标签	删除指定的存储桶标签

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置存储桶标签

功能说明

PUT Bucket tagging 用于为已存在的存储桶设置标签。

示例代码

Objective-C

```
QCloudPutBucketTaggingRequest *putReq = [QCloudPutBucketTaggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putReq.bucket = @"examplebucket-1250000000";

// 标签集合
QCloudTagging *taggings = [QCloudTagging new];

QCloudTag *tag1 = [QCloudTag new];

// 标签的 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
tag1.key = @"age";

// 标签的 Value，长度不超过256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
tag1.value = @"20";
QCloudTag *tag2 = [QCloudTag new];
tag2.key = @"name";
tag2.value = @"karis";

// 标签集合，最多支持10个标签
QCloudTagSet *tagSet = [QCloudTagSet new];
tagSet.tag = @[tag1,tag2];
taggings.tagSet = tagSet;

// 标签集合
putReq.taggings = taggings;

[putReq setFinishBlock:^(id outputObject, NSError *error) {
```

```
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketTagging:putReq];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudPutBucketTaggingRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
let taggings = QCloudTagging.init();

// 标签集合
let tagSet = QCloudTagSet.init();
taggings.tagSet = tagSet;
let tag1 = QCloudTag.init();

// 标签的 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
tag1.key = "age";

// 标签的 Value，长度不超过256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
tag1.value = "20";

let tag2 = QCloudTag.init();
tag2.key = "name";
tag2.value = "karis";

// 标签集合，最多支持10个标签
tagSet.tag = [tag1,tag2];

// 标签集合
req.taggings = taggings;
req.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketTagging(req);
```

说明：
更多完整示例，请前往 GitHub 查看。

查询存储桶标签

功能说明

GET Bucket tagging 用于查询指定存储桶下已有的存储桶标签。

示例代码

Objective-C

```
QCloudGetBucketTaggingRequest *getReq = [QCloudGetBucketTaggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";

[getReq setFinishBlock:^(QCloudBucketTagging * result, NSError * error) {
// tag的集合
QCloudTagSet * tagSet = result.tagSet;
}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketTagging:getReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudGetBucketTaggingRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
req.setFinish { (result, error) in
if let result = result {
// 标签集合
let tagSet = result.tagSet
} else {
print(error!);
}
};
QCloudCOSXMLService.defaultCOSXML().getBucketTagging(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除存储桶标签

功能说明

DELETE Bucket tagging 用于删除指定存储桶下已有的存储桶标签。

示例代码

Objective-C

```
QCloudDeleteBucketTaggingRequest *delReq = [QCloudDeleteBucketTaggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
delReq.bucket = @"examplebucket-1250000000";

[delReq setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
```

```
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketTagging:delReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudDeleteBucketTaggingRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
req.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
};
QCloudCOSXMLService.defaultCOSXML().deleteBucketTagging(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

对象标签

最近更新时间：2022-08-25 09:06:50

简介

本文档提供关于对象标签的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Object tagging	设置对象标签	为已上传的对象设置标签
GET Object tagging	查询对象标签	查询指定对象下已有的对象标签
DELETE Object tagging	删除对象标签	删除指定对象下已有的对象标签

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置对象标签

功能说明

COS 支持为已存在的对象设置标签。通过为对象添加键值对作为对象标签，可以协助您分组管理已有的对象资源。

示例代码

Objective-C

```
QCloudPutObjectTaggingRequest *putReq = [QCloudPutObjectTaggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putReq.bucket = @"examplebucket-1250000000";

// 标签集合
QCloudTagging *taggings = [QCloudTagging new];

QCloudTag *tag1 = [QCloudTag new];

// 标签的 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
tag1.key = @"age";

// 标签的 Value，长度不超过256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
tag1.value = @"20";
QCloudTag *tag2 = [QCloudTag new];
tag2.key = @"name";
tag2.value = @"karis";

// 标签集合，最多支持10个标签
QCloudTagSet *tagSet = [QCloudTagSet new];
tagSet.tag = @[tag1,tag2];
taggings.tagSet = tagSet;

// 标签集合
putReq.taggings = taggings;
```

```
[putReq setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] PutObjectTagging:putReq];
```

Swift

```
let putReq = QCloudPutObjectTaggingRequest()

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putReq.bucket = "examplebucket-1250000000";

// 标签集合
let taggings = QCloudTagging();

let tag1 = QCloudTag();

// 标签的 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
tag1.key = "age";

// 标签的 Value，长度不超过256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
tag1.value = "20";
let tag2 = QCloudTag();
tag2.key = "name";
tag2.value = "karis";

// 标签集合，最多支持10个标签
let tagSet = QCloudTagSet();
tagSet.tag = [tag1,tag2];
taggings.tagSet = tagSet;

// 标签集合
putReq.taggings = taggings;

req.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putObjectTagging(putReq);
```

查询对象标签

功能说明

查询指定对象下已有的对象标签。

示例代码

Objective-C

```
QCloudGetObjectTaggingRequest *getReq = [QCloudGetObjectTaggingRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";

[getReq setFinishBlock:^(QCloudTagging * result, NSError * error) {

// tag的集合
QCloudTagSet * tagSet = result.tagSet;
}];
[[QCloudCOSXMLService defaultCOSXML] GetObjectTagging:getReq];
```

Swift

```
let getReq = QCloudGetObjectTaggingRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = "examplebucket-1250000000";

req.finishBlock = {(result,error) in
if let result = result {
// tag的集合
let tagSet = result.tagSet;
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().getObjectTagging(getReq);
```

删除对象标签

> ⚠ **注意：**
> COS iOS SDK 版本需要大于等于 v5.9.9。

功能说明

删除指定对象下已有的对象标签。

请求示例

Objective-C

```
QCloudDeleteObjectTaggingRequest *request = [QCloudDeleteObjectTaggingRequest new];

// 文件名
request.object = @"test.png";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";
request.versionId = @"versionId";

[request setFinishBlock:^(id * result, NSError * error) {

if(!error){
// 删除成功
```

```
}else{
// 删除失败
}
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteObjectTagging:request];
```

Swift

```
let request = QCloudDeleteObjectTaggingRequest();

// 文件名
request.object = "test.png";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = "examplebucket-1250000000";

request.versionId = "versionId";

req.finishBlock = {(result,error) in
if(!error){
// 删除成功
}else{
// 删除失败
}
}

QCloudCOSXMLService.defaultCOSXML().deleteObjectTagging(request);
```

静态网站

最近更新时间：2022-08-25 09:06:55

简介

本文档提供关于静态网站的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket website	设置静态网站	设置存储桶的静态网站配置
GET Bucket website	查询静态网站配置	查询存储桶的静态网站配置
DELETE Bucket website	删除静态网站配置	删除存储桶的静态网站配置

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置静态网站

功能说明

PUT Bucket website 用于为存储桶配置静态网站。

示例代码

Objective-C

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
NSString *bucket = @"examplebucket-1250000000";

NSString *indexDocumentSuffix = @"index.html";
NSString *errorDocKey = @"error.html";
NSString *derPro = @"https";
int errorCode = 451;
NSString * replaceKeyPrefixWith = @"404.html";
QCloudPutBucketWebsiteRequest *putReq = [QCloudPutBucketWebsiteRequest new];
putReq.bucket = bucket;

QCloudWebsiteConfiguration *config = [QCloudWebsiteConfiguration new];

QCloudWebsiteIndexDocument *indexDocument = [QCloudWebsiteIndexDocument new];

// 指定索引文档的对象键后缀。例如指定为index.html，那么当访问到存储桶的根目录时，会自动返回
// index.html 的内容，或者当访问到article/目录时，会自动返回 article/index.html的内容
indexDocument.suffix = indexDocumentSuffix;
// 索引文档配置
config.indexDocument = indexDocument;

// 错误文档配置
QCloudWebisteErrorDocument *errDocument = [QCloudWebisteErrorDocument new];
errDocument.key = errorDocKey;
// 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容
config.errorDocument = errDocument;

// 重定向所有请求配置
QCloudWebsiteRedirectAllRequestsTo *redir = [QCloudWebsiteRedirectAllRequestsTo new];
redir.protocol = derPro;
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 指定重定向所有请求的目标协议，只能设置为 https
config.redirectAllRequestsTo = redir;

// 单条重定向规则配置
QCloudWebsiteRoutingRule *rule = [QCloudWebsiteRoutingRule new];

// 重定向规则的条件配置
QCloudWebsiteCondition *contition = [QCloudWebsiteCondition new];
contition.httpErrorCodeReturnedEquals = errorCode;
rule.condition = contition;

// 重定向规则的具体重定向目标配置
QCloudWebsiteRedirect *webRe = [QCloudWebsiteRedirect new];
webRe.protocol = derPro;

// 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，
// 仅可在 Condition 为 KeyPrefixEquals 时设置
webRe.replaceKeyPrefixWith = replaceKeyPrefixWith;
rule.redirect = webRe;

QCloudWebsiteRoutingRules *routingRules = [QCloudWebsiteRoutingRules new];
routingRules.routingRule = @[rule];

// 重定向规则配置，最多设置100条 RoutingRule
config.rules = routingRules;
putReq.websiteConfiguration = config;

[putReq setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] PutBucketWebsite:putReq];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudPutBucketWebsiteRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

let indexDocumentSuffix = "index.html";
let errorDocKey = "error.html";
let errorCode = 451;
let replaceKeyPrefixWith = "404.html";

let config = QCloudWebsiteConfiguration.init();

let indexDocument = QCloudWebsiteIndexDocument.init();

// 指定索引文档的对象键后缀。例如指定为index.html，那么当访问到存储桶的根目录时，会自动返回
```

```
// index.html 的内容，或者当访问到article/目录时，会自动返回 article/index.html的内容
indexDocument.suffix = indexDocumentSuffix;

// 索引文档配置
config.indexDocument = indexDocument;

// 错误文档配置
let errDocument = QCloudWebisteErrorDocument.init();
errDocument.key = errorDocKey;

// 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容
config.errorDocument = errDocument;

// 重定向所有请求配置
let redir = QCloudWebsiteRedirectAllRequestsTo.init();

// 指定重定向所有请求的目标协议，只能设置为 https
redir.protocol = "https";
config.redirectAllRequestsTo = redir;

// 单条重定向规则配置
let rule = QCloudWebsiteRoutingRule.init();

// 重定向规则的条件配置
let contition = QCloudWebsiteCondition.init();
contition.httpErrorCodeReturnedEquals = Int32(errorCode);
rule.condition = contition;

// 重定向规则的具体重定向目标配置
let webRe = QCloudWebsiteRedirect.init();
webRe.protocol = "https";

// 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，
// 仅可在 Condition 为 KeyPrefixEquals 时设置
webRe.replaceKeyPrefixWith = replaceKeyPrefixWith;
rule.redirect = webRe;

let routingRules = QCloudWebsiteRoutingRules.init();
routingRules.routingRule = [rule];

// 重定向规则配置，最多设置100条 RoutingRule
config.rules = routingRules;
req.websiteConfiguration = config;

req.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketWebsite(req);
```

说明：
更多完整示例，请前往 GitHub 查看。

查询静态网站配置

功能说明

GET Bucket website 用于查询与存储桶关联的静态网站配置信息。

示例代码

Objective-C

```
QCloudGetBucketWebsiteRequest *getReq = [QCloudGetBucketWebsiteRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";
[getReq setFinishBlock:^(QCloudWebsiteConfiguration * result,
NSError * error) {

// 设置重定向规则，最多设置100条RoutingRule
QCloudWebsiteRoutingRules *rules =result.rules;

// 索引文档
QCloudWebsiteIndexDocument *indexDocument = result.indexDocument;

// 错误文档
QCloudWebisteErrorDocument *errorDocument = result.errorDocument;

// 重定向所有请求
QCloudWebsiteRedirectAllRequestsTo *redirectAllRequestsTo = result.redirectAllRequestsTo;

}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketWebsite:getReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudGetBucketWebsiteRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

req.setFinish {(result,error) in
if let result = result {
let rules = result.rules
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketWebsite(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除静态网站配置

功能说明

DELETE Bucket website 用于删除存储桶中的静态网站配置。

示例代码

Objective-C

```
QCloudDeleteBucketWebsiteRequest *delReq = [QCloudDeleteBucketWebsiteRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
delReq.bucket = @"examplebucket-1250000000";

[delReq setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketWebsite:delReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let delReq = QCloudDeleteBucketWebsiteRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
delReq.bucket = "examplebucket-1250000000";

delReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().deleteBucketWebsite(delReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

清单

最近更新时间：2022-08-25 09:07:00

简介

本文档提供关于清单的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket inventory	设置清单任务	设置存储桶的清单任务
GET Bucket inventory	查询清单任务	查询存储桶的清单任务
DELETE Bucket inventory	删除清单任务	删除存储桶的清单任务

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API 参考。

设置清单任务

功能说明

PUT Bucket inventory 用于在存储桶中创建清单任务。

示例代码

Objective-C

```
QCloudPutBucketInventoryRequest *putReq = [QCloudPutBucketInventoryRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putReq.bucket= @"examplebucket-1250000000";

// 清单任务的名称
putReq.inventoryID = @"list1";

// 用户在请求体中使用 XML 语言设置清单任务的具体配置信息。配置信息包括清单任务分析的对象，
// 分析的频次，分析的维度，分析结果的格式及存储的位置等信息。
QCloudInventoryConfiguration *config = [QCloudInventoryConfiguration new];

// 清单的名称，与请求参数中的 id 对应
config.identifier = @"list1";

// 清单是否启用的标识：
// 如果设置为 true，清单功能将生效
// 如果设置为 false，将不生成任何清单
config.isEnabled = @"True";

// 描述存放清单结果的信息
QCloudInventoryDestination *des = [QCloudInventoryDestination new];

QCloudInventoryBucketDestination *btDes =[QCloudInventoryBucketDestination new];

// 清单分析结果的文件形式，可选项为 CSV 格式
btDes.cs = @"CSV";

// 存储桶的所有者 ID
btDes.account = @"1278687956";
```

```
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
btDes.bucket = @"qcs::cos:ap-guangzhou::examplebucket-1250000000";

// 清单分析结果的前缀
btDes.prefix = @"list1";

// COS 托管密钥的加密方式
QCloudInventoryEncryption *enc = [QCloudInventoryEncryption new];
enc.ssecos = @"";

// 为清单结果提供服务端加密的选项
btDes.encryption = enc;

// 清单结果导出后存放的存储桶信息
des.bucketDestination = btDes;

// 描述存放清单结果的信息
config.destination = des;

// 配置清单任务周期
QCloudInventorySchedule *sc = [QCloudInventorySchedule new];

// 清单任务周期，可选项为按日或者按周，枚举值：Daily、Weekly
sc.frequency = @"Daily";
config.schedule = sc;
QCloudInventoryFilter *fileter = [QCloudInventoryFilter new];
fileter.prefix = @"myPrefix";
config.filter = fileter;
config.includedObjectVersions = QCloudCOSIncludedObjectVersionsAll;
QCloudInventoryOptionalFields *fields = [QCloudInventoryOptionalFields new];

fields.field = @[ @"Size",
@"LastModifiedDate",
@"ETag",
@"StorageClass",
@"IsMultipartUploaded",
@"ReplicationStatus"];

// 设置清单结果中应包含的分析项目
config.optionalFields = fields;
putReq.inventoryConfiguration = config;
[putReq setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary * result = (NSDictionary *)outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketInventory:putReq];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let putReq = QCloudPutBucketInventoryRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putReq.bucket = "examplebucket-1250000000";

// 清单任务的名称
putReq.inventoryID = "list1";

// 用户在请求体中使用 XML 语言设置清单任务的具体配置信息。配置信息包括清单任务分析的对象，
// 分析的频次，分析的维度，分析结果的格式及存储的位置等信息。
let config = QCloudInventoryConfiguration.init();

// 清单的名称，与请求参数中的 id 对应
config.identifier = "list1";

// 清单是否启用的标识：
// 如果设置为 true，清单功能将生效
// 如果设置为 false，将不生成任何清单
config.isEnabled = "True";

// 描述存放清单结果的信息
let des = QCloudInventoryDestination.init();
let btDes = QCloudInventoryBucketDestination.init();

// 清单分析结果的文件形式，可选项为 CSV 格式
btDes.cs = "CSV";

// 存储桶的所有者 ID
btDes.account = "1278687956";

// 清单分析结果的存储桶名
btDes.bucket = "qcs::cos:ap-guangzhou::examplebucket-1250000000";

// 清单分析结果的前缀
btDes.prefix = "list1";

// COS 托管密钥的加密方式
let enc = QCloudInventoryEncryption.init();
enc.ssecos = "";

// 为清单结果提供服务端加密的选项
btDes.encryption = enc;

// 清单结果导出后存放的存储桶信息
des.bucketDestination = btDes;

// 描述存放清单结果的信息
config.destination = des;

// 配置清单任务周期
let sc = QCloudInventorySchedule.init();

// 清单任务周期，可选项为按日或者按周，枚举值：Daily、Weekly
sc.frequency = "Daily";
config.schedule = sc;
```

```
let fileter = QCloudInventoryFilter.init();
fileter.prefix = "myPrefix";
config.filter = fileter;
config.includedObjectVersions = .all;
let fields = QCloudInventoryOptionalFields.init();
fields.field = [ "Size",
"LastModifiedDate",
"ETag",
"StorageClass",
"IsMultipartUploaded",
"ReplicationStatus"];
// 设置清单结果中应包含的分析项目
config.optionalFields = fields;
putReq.inventoryConfiguration = config;

putReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().putBucketInventory(putReq);
```

说明：
更多完整示例，请前往 GitHub 查看。

错误码说明

该请求可能会发生的一些常见的特殊错误如下：

错误码	描述	状态码
InvalidArgument	不合法的参数值	HTTP 400 Bad Request
TooManyConfigurations	清单数量已经达到1000条的上限	HTTP 400 Bad Request
AccessDenied	未授权的访问。您可能不具备访问该存储桶的权限	HTTP 403 Forbidden

查询清单任务

功能说明

GET Bucket inventory 用于查询存储桶中用户的清单任务信息。

示例代码

Objective-C

```
QCloudGetBucketInventoryRequest *getReq = [QCloudGetBucketInventoryRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";

// 清单任务的名称
getReq.inventoryID = @"list1";
[getReq setFinishBlock:^(QCloudInventoryConfiguration * _Nonnull result,
NSError * _Nonnull error) {
```

```
// result 包含清单的信息
}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketInventory:getReq];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudGetBucketInventoryRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";
// 清单任务的名称
req.inventoryID = "list1";
req.setFinish {(result,error) in
if let result = result {
// 任务信息
let enabled = result.isEnabled
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketInventory(req);
```

说明：
更多完整示例，请前往 GitHub 查看。

删除清单任务

功能说明

DELETE Bucket inventory 用于删除存储桶中指定的清单任务。

示例代码

Objective-C

```
QCloudDeleteBucketInventoryRequest *delReq = [QCloudDeleteBucketInventoryRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
delReq.bucket = @"examplebucket-1250000000";

// 清单任务的名称
delReq.inventoryID = @"list1";
[delReq setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary * result = (NSDictionary *)outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketInventory:delReq];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let delReq = QCloudDeleteBucketInventoryRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
delReq.bucket = "examplebucket-1250000000";

// 清单任务的名称
delReq.inventoryID = "list1";
delReq.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().deleteBucketInventory(delReq);
```

说明：
更多完整示例，请前往 GitHub 查看。

访问管理
跨域访问

最近更新时间：2022-08-25 09:07:05

简介

本文档提供关于跨域访问的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket cors	设置跨域配置	设置存储桶的跨域名访问权限
GET Bucket cors	查询跨域配置	查询存储桶的跨域名访问配置信息
DELETE Bucket cors	删除跨域配置	删除存储桶的跨域名访问配置信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置跨域配置

功能说明

设置指定存储桶的跨域名访问配置信息（PUT Bucket cors）。

示例代码

Objective-C

```
QCloudPutBucketCORSRequest* putCORS = [QCloudPutBucketCORSRequest new];
QCloudCORSConfiguration* cors = [QCloudCORSConfiguration new];

QCloudCORSRule* rule = [QCloudCORSRule new];

// 配置规则 ID
rule.identifier = @"rule1";

// 跨域请求可以使用的 HTTP 请求头部，支持通配符 *
rule.allowedHeader = @[@"origin",@"host",@"accept",
@"content-type",@"authorization"];
rule.exposeHeader = @"ETag";

// 跨域请求允许的 HTTP 操作，例如：GET，PUT，HEAD，POST，DELETE
rule.allowedMethod = @[@"GET",@"PUT",@"POST", @"DELETE", @"HEAD"];

// 跨域请求得到结果的有效期
rule.maxAgeSeconds = 3600;

// 允许的访问来源，支持通配符 *，格式为：协议://域名[:端口]
rule.allowedOrigin = @"http://cloud.tencent.com";

cors.rules = @[rule];
putCORS.corsConfiguration = cors;

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putCORS.bucket = @"examplebucket-1250000000";
```

```
[putCORS setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary * result = (NSDictionary *)outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] PutBucketCORS:putCORS];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let putBucketCorsReq = QCloudPutBucketCORSRequest.init();

let corsConfig = QCloudCORSConfiguration.init();

let rule = QCloudCORSRule.init();

// 配置规则的 ID
rule.identifier = "rule1";

// 跨域请求可以使用的 HTTP 请求头部，支持通配符 *
rule.allowedHeader = ["origin","host","accept","content-type","authorization"];
rule.exposeHeader = "Etag";

// 跨域请求允许的 HTTP 操作，例如：GET，PUT，HEAD，POST，DELETE
rule.allowedMethod = ["GET","PUT","POST", "DELETE", "HEAD"];

// 跨域请求得到结果的有效期
rule.maxAgeSeconds = 3600;

// 允许的访问来源，支持通配符 *，格式为：协议://域名[:端口]
rule.allowedOrigin = "*";

corsConfig.rules = [rule];
putBucketCorsReq.corsConfiguration = corsConfig;

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketCorsReq.bucket = "examplebucket-1250000000";
putBucketCorsReq.finishBlock = {(result,error) in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketCORS(putBucketCorsReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询跨域配置

功能说明

查询指定存储桶的跨域名访问配置信息（GET Bucket cors）。

示例代码

Objective-C

```
QCloudGetBucketCORSRequest* corsReqeust = [QCloudGetBucketCORSRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
corsReqeust.bucket = @"examplebucket-1250000000";

[corsReqeust setFinishBlock:^(QCloudCORSConfiguration * _Nonnull result,
NSError * _Nonnull error) {
// 跨域规则列表
NSArray<QCloudCORSRule*> *rules = result.rules;

}];

[[QCloudCOSXMLService defaultCOSXML] GetBucketCORS:corsReqeust];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketCorsRes = QCloudGetBucketCORSRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketCorsRes.bucket = "examplebucket-1250000000";
getBucketCorsRes.setFinish { (corsConfig, error) in
if let corsConfig = corsConfig {
// 跨域规则列表
let rules = corsConfig.rules
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketCORS(getBucketCorsRes);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

删除跨域配置

功能说明

删除指定存储桶的跨域名访问配置（DELETE Bucket cors）。

示例代码

Objective-C

```
QCloudDeleteBucketCORSRequest* deleteCORS = [QCloudDeleteBucketCORSRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
deleteCORS.bucket = @"examplebucket-1250000000";

[deleteCORS setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary* info = (NSDictionary *) outputObject;
}];
[[QCloudCOSXMLService defaultCOSXML] DeleteBucketCORS:deleteCORS];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let deleteBucketCorsRequest = QCloudDeleteBucketCORSRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
deleteBucketCorsRequest.bucket = "examplebucket-1250000000";

deleteBucketCorsRequest.finishBlock = {(result,error) in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().deleteBucketCORS(deleteBucketCorsRequest);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

自定义域名

最近更新时间：2022-08-25 09:07:09

简介

本文档提供关于自定义域名的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket domain	设置自定义域名	设置存储桶的自定义域名信息
GET Bucket domain	查询自定义域名	查询存储桶的自定义域名信息

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置自定义域名

功能说明

PUT Bucket domain 用于为存储桶配置自定义域名。

示例代码

Objective-C

```
QCloudPutBucketDomainRequest *req = [QCloudPutBucketDomainRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = @"examplebucket-1250000000";

QCloudDomainConfiguration *config = [QCloudDomainConfiguration new];
QCloudDomainRule *rule = [QCloudDomainRule new];

// 源站状态，可选 QCloudDomainStatueEnabled、 QCloudDomainStatueDisabled
rule.status = QCloudDomainStatueEnabled;
// 域名信息
rule.name = @"www.baidu.com";

// 替换已存在的配置、有效值CNAME/TXT 填写则强制校验域名所有权之后，再下发配置
rule.replace = QCloudCOSDomainReplaceTypeTxt;
rule.type = QCloudCOSDomainTypeRest;

// 规则描述集合的数组
config.rules = @[rule];

// 域名配置的规则
req.domain = config;

[req setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;

}];
[[QCloudCOSXMLService defaultCOSXML]PutBucketDomain:req];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudPutBucketDomainRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

let config = QCloudDomainConfiguration.init();
let rule = QCloudDomainRule.init();
// 开启状态，可选 .enabled, .disabled
rule.status = .enabled;
rule.name = "www.baidu.com";

// 替换已存在的配置、有效值CNAME/TXT 填写则强制校验域名所有权之后，再下发配置
rule.replace = .txt;
rule.type = .rest;

// 规则描述集合的数组
config.rules = [rule];

// 域名配置的规则
req.domain = config;
req.finishBlock = {(result,error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketDomain(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

返回错误码说明

该请求可能会发生的一些常见的特殊错误如下：

状态码	说明
HTTP 409 Conflict	该域名记录已存在，且请求中没有设置强制覆盖。或者该域名记录不存在，且请求中设置了强制覆盖
HTTP 451 Unavailable For Legal Reasons	该域名是中国境内域名，并且没有备案

查询自定义域名

功能说明

GET Bucket domain 用于查询存储桶的自定义域名信息。

示例代码

Objective-C

```
QCloudGetBucketDomainRequest *getReq = [QCloudGetBucketDomainRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getReq.bucket = @"examplebucket-1250000000";

[getReq setFinishBlock:^(QCloudDomainConfiguration * _Nonnull result,
NSError * _Nonnull error) {
// 规则描述集合的数组
NSArray *rules = result.rules;
}];
[[QCloudCOSXMLService defaultCOSXML]GetBucketDomain:getReq];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let req = QCloudGetBucketDomainRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
req.bucket = "examplebucket-1250000000";

req.finishBlock = {(result,error) in
if let result = result {
// result 包含源站信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketDomain(req);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

返回参数说明

参数名称	描述	类型
x-cos-domain-txt-verification	域名校验信息，该字段是一个 MD5 校验值，原串格式为：cos[Region][BucketName-APPID][BucketCreateTime]，其中 Region 为存储桶所在地域，BucketCreateTime 为存储桶 GMT 创建时间	String

访问控制

最近更新时间：2022-08-25 09:07:13

简介

本文档提供关于存储桶、对象的访问控制列表（ACL）的相关 API 概览以及 SDK 示例代码。

存储桶 ACL

API	操作名	操作描述
PUT Bucket acl	设置存储桶 ACL	设置指定存储桶的访问权限控制列表（ACL）
GET Bucket acl	查询存储桶 ACL	查询指定存储桶的访问权限控制列表（ACL）

对象 ACL

API	操作名	操作描述
PUT Object acl	设置对象 ACL	设置存储桶中某个对象的访问控制列表
GET Object acl	查询对象 ACL	查询对象的访问控制列表

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

存储桶 ACL

设置存储桶 ACL

功能说明

设置指定存储桶的访问权限控制列表（ACL）。

示例代码

Objective-C

```
QCloudPutBucketACLRequest* putACL = [QCloudPutBucketACLRequest new];

// 授予权限的账号 ID
NSString* uin = @"100000000001";
NSString *ownerIdentifier = [NSString stringWithFormat:@"qcs::cam::uin/%@:uin/%@"
, uin,uin];
NSString *grantString = [NSString stringWithFormat:@"id=\"%@\"",ownerIdentifier];

// 赋予被授权者读写权限
putACL.grantFullControl = grantString;

// 赋予被授权者读权限
putACL.grantRead = grantString;

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putACL.bucket = @"examplebucket-1250000000";

[putACL setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取服务器返回的 header 信息
NSDictionary * result = (NSDictionary *)outputObject;
```

```
}];
// 设置acl
[[QCloudCOSXMLService defaultCOSXML] PutBucketACL:putACL];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let putBucketACLReq = QCloudPutBucketACLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putBucketACLReq.bucket = "examplebucket-1250000000";

// 授予权限的账号 ID
let appTD = "100000000001";
let ownerIdentifier = "qcs::cam::uin/\(appTD):uin/\(appTD)";
let grantString = "id=\"\(ownerIdentifier)\"";
// 赋予被授权者写权限
putBucketACLReq.grantWrite = grantString;

// 赋予被授权者读权限
putBucketACLReq.grantRead = grantString;

// 赋予被授权者读写权限 grantFullControl == grantRead + grantWrite
putBucketACLReq.grantFullControl = grantString;

putBucketACLReq.finishBlock = {(result,error) in
if let result = result {
// 可以从 result 中获取服务器返回的 header 信息
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketACL(putBucketACLReq);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询存储桶 ACL

功能说明

查询指定存储桶的访问权限控制列表（ACL）。

示例代码

Objective-C

```
QCloudGetBucketACLRequest* getBucketACl = [QCloudGetBucketACLRequest new];

// 存储桶名称，格式：BucketName-APPID
getBucketACl.bucket = @"examplebucket-1250000000";

[getBucketACl setFinishBlock:^(QCloudACLPolicy * _Nonnull result,
```

```
NSError * _Nonnull error) {
// 被授权者与权限的信息
QCloudAccessControlList *acl = result.accessControlList;
}];

[[QCloudCOSXMLService defaultCOSXML] GetBucketACL:getBucketACl];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getBucketACLReq = QCloudGetBucketACLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getBucketACLReq.bucket = "examplebucket-1250000000";

getBucketACLReq.setFinish { (result, error) in
if let result = result {
// ACL 授权信息
let acl = result.accessControlList;
} else {
print(error!)
}
}
QCloudCOSXMLService.defaultCOSXML().getBucketACL(getBucketACLReq)
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

对象 ACL

设置对象 ACL

功能说明

设置存储桶中某个对象的访问控制列表（ACL）。

示例代码

Objective-C

```
QCloudPutObjectACLRequest* request = [QCloudPutObjectACLRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

NSString *grantString = [NSString stringWithFormat:@"id=\"%@\"",@"100000000001"];

// grantFullControl 等价于 grantRead + grantWrite
// 赋予被授权者读写权限。
request.grantFullControl = grantString;
```

```
// 赋予被授权者读权限。
request.grantRead = grantString;

[request setFinishBlock:^(id outputObject, NSError *error) {
// 可以从 outputObject 中获取 response 中 etag 或者自定义头部等信息
NSDictionary* info = (NSDictionary *) outputObject;
}];

[[QCloudCOSXMLService defaultCOSXML] PutObjectACL:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let putObjectACl = QCloudPutObjectACLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
putObjectACl.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 "video/xxx/movie.mp4"
putObjectACl.object = "exampleobject";
let grantString = "id=\"1000000000001\"";

// grantFullControl 等价于 grantRead + grantWrite
putObjectACl.grantFullControl = grantString;
// 赋予被授权者读权限。
putObjectACl.grantRead = grantString;


putObjectACl.finishBlock = {(result,error)in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().putObjectACL(putObjectACl);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询对象 ACL

功能说明

查询对象的访问控制列表。

示例代码

Objective-C

```
QCloudGetObjectACLRequest *request = [QCloudGetObjectACLRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
```

```
request.object = @"exampleobject";

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = @"examplebucket-1250000000";

__block QCloudACLPolicy* policy;
[request setFinishBlock:^(QCloudACLPolicy * _Nonnull result,
NSError * _Nonnull error) {

policy = result;
// result.accessControlList; 被授权者与权限的信息
// result.owner; 持有者的信息
}];

[[QCloudCOSXMLService defaultCOSXML] GetObjectACL:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let getObjectACL = QCloudGetObjectACLRequest.init();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
getObjectACL.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 "video/xxx/movie.mp4"
getObjectACL.object = "exampleobject";
getObjectACL.setFinish { (result, error) in
if let result = result {
// 对象授权信息
let acl = result.accessControlList
} else {
print(error!);
}
}
QCloudCOSXMLService.defaultCOSXML().getObjectACL(getObjectACL);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

防盗链

最近更新时间：2022-08-25 09:07:16

简介

本文档提供关于存储桶 Referer 白名单或者黑名单的 API 概览以及 SDK 示例代码。

API	操作名	操作描述
PUT Bucket referer	设置存储桶 Referer	设置存储桶 Referer 白名单或者黑名单
GET Bucket referer	查询存储桶 Referer	查询存储桶 Referer 白名单或者黑名单

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

设置存储桶 Referer

功能说明

设置指定存储桶的 Referer 白名单或者黑名单（PUT Bucket referer）。

> 注意：
> COS iOS SDK 版本需要大于等于 v5.9.6。

请求示例

Objective-C

```
QCloudPutBucketRefererRequest* request = [QCloudPutBucketRefererRequest new];

// 防盗链类型，枚举值：Black-List、White-List
request.refererType = QCloudBucketRefererTypeBlackList;

// 是否开启防盗链，枚举值：Enabled、Disabled
request.status = QCloudBucketRefererStatusEnabled;

// 是否允许空 Referer 访问，枚举值：Allow、Deny，默认值为 Deny
request.configuration = QCloudBucketRefererConfigurationDeny;

// 生效域名列表， 支持多个域名且为前缀匹配， 支持带端口的域名和 IP， 支持通配符*，做二级域名或多级域名的通配
request.domainList = @[@"*.com",@"*.qq.com"];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(id outputObject, NSError *error) {
if (error){
// 添加防盗链失败
}else{
// 添加防盗链失败
}

}];
[[QCloudCOSXMLService defaultCOSXML] PutBucketReferer:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudPutBucketRefererRequest.init();

// 防盗链类型，枚举值：Black-List、White-List
request.refererType = QCloudBucketRefererType.blackList;

// 是否开启防盗链，枚举值：Enabled、Disabled
request.status = QCloudBucketRefererStatus.enabled;

// 是否允许空 Referer 访问，枚举值：Allow、Deny，默认值为 Deny
request.configuration = QCloudBucketRefererConfiguration.allow;

// 生效域名列表， 支持多个域名且为前缀匹配， 支持带端口的域名和 IP， 支持通配符*，做二级域名或多级域名的通配
request.domainList = ["*.com","*.qq.com"];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

request.finishBlock = {(result,error) in
if (error != nil){
// 添加防盗链失败
}else{
// 添加防盗链失败
}
}
QCloudCOSXMLService.defaultCOSXML().putBucketReferer(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询存储桶 Referer

功能说明

查询指定存储桶 Referer 白名单或者黑名单（GET Bucket referer）。

> 注意：
> COS iOS SDK 版本需要大于等于 v5.9.6。

请求示例

Objective-C

```
QCloudGetBucketRefererRequest* request = [QCloudGetBucketRefererRequest new];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

[request setFinishBlock:^(QCloudBucketRefererInfo * outputObject, NSError *error) {
// outputObject 请求到的防盗链，详细字段请查看api文档或者SDK源码
// QCloudBucketRefererInfo 类；
```

```
}];
[[QCloudCOSXMLService defaultCOSXML] GetBucketReferer:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudGetBucketRefererRequest.init();

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

request.finishBlock = {(result,error) in
// outputObject 请求到的防盗链，详细字段请查看api文档或者SDK源码
// QCloudBucketRefererInfo 类；
}
QCloudCOSXMLService.defaultCOSXML().getBucketReferer(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

内容识别
图片标签

最近更新时间：2022-08-25 09:07:22

简介

本文档提供关于识别图片标签功能相关的 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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

下载时识别

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。

示例代码

Objective-C

```
QCloudCIPicRecognitionRequest *request = [QCloudCIPicRecognitionRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.bucket = imageTestBucket.name;
// 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 = @"pic-recognition.png";

[request setFinishBlock:^(QCloudCIPicRecognitionResults *_Nullable result, NSError *_Nullable error) {
//result：返回的标签信息
}];

[[QCloudCOSXMLService defaultCOSXML] CIPicRecognition:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let put = QCloudCIPicRecognitionRequest();

// 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"
put.object = "exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket

put.bucket = "examplebucket-1250000000";
put.setFinish { (outoutObject, error) in
//outoutObject
};
QCloudCOSXMLService.defaultCOSXML().ciPicRecognition(put);
```

说明：
更多完整示例，请前往 GitHub 查看。

图片二维码

最近更新时间：2022-08-25 09:07:25

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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。

下载时识别

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

Objective-C

```
QCloudQRCodeRecognitionRequest *put = [QCloudQRCodeRecognitionRequest new];
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.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"
put.object = @"exampleobject";

QCloudPicOperations *op = [[QCloudPicOperations alloc] init];
// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = NO;
QCloudPicOperationRule * rule = [[QCloudPicOperationRule alloc]init];
rule.fileid = @"test";
//二维码识别的rule
rule.rule = @"QRcode/cover/1";
// 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
rule.fileid = @"test";
op.rule = @[ rule ];
put.picOperations = op;
[put setFinishBlock:^(QCloudCIObject * _Nonnull result, NSError * _Nonnull error) {
//从result.qrcodeInfos中获取二维码信息
NSLog(@"result = %@",result.qrcodeInfos)
}];
[[QCloudCOSXMLService defaultCOSXML] CIQRCodeRecognition:put];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let put = QCloudQRCodeRecognitionRequest();
```

```
// 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"
put.object = "exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket

put.bucket = "examplebucket-1250000000";
let op = QCloudPicOperations.init();

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = false;

let rule = QCloudPicOperationRule.init();

// 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
rule.fileid = "test";

// 二维码识别的rule
rule.rule = "QRcode/cover/1";

op.rule = [rule];
put.picOperations = op;
put.setFinish { (outoutObject, error) in

};
QCloudCOSXMLService.defaultCOSXML().ciqrCodeRecognition(put);
```

说明：

更多完整示例，请前往 GitHub 查看。

语音识别
任务接口

最近更新时间：2022-08-25 09:07:29

简介

本文档提供关于提交语音识别任务、查询指定的语音识别任务、批量拉取语音识别任务相关的 API 概览以及 SDK 示例代码。

API	说明
提交语音识别任务	接口用于提交一个语音识别任务。
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	用于拉取符合条件的语音识别任务。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

提交语音识别任务

功能说明

接口用于提交一个语音识别任务。

> ⚠ 注意：
> COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudPostAudioDiscernTaskRequest * request = [[QCloudPostAudioDiscernTaskRequest alloc]init];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";
request.regionName = @"regionName";

QCloudPostAudioDiscernTaskInfo* taskInfo = [QCloudPostAudioDiscernTaskInfo new];
taskInfo.Tag = @"SpeechRecognition";
// 队列 ID ,通过查询语音识别队列获取
taskInfo.QueueId = @"QueueId";
// 操作规则
QCloudPostAudioDiscernTaskInfoInput * input = QCloudPostAudioDiscernTaskInfoInput.new;
input.Object = @"test1";
// 待操作的语音文件
taskInfo.Input = input;
QCloudPostAudioDiscernTaskInfoOperation * op = [QCloudPostAudioDiscernTaskInfoOperation new];
QCloudPostAudioDiscernTaskInfoOutput * output = QCloudPostAudioDiscernTaskInfoOutput.new;
output.Region = @"regionName";
output.Bucket = @"examplebucket-1250000000";
output.Object = @"test";
// 结果输出地址
op.Output = output;

QCloudPostAudioDiscernTaskInfoSpeechRecognition * speechRecognition = [QCloudPostAudioDiscernTaskInfoSpeechRecognition new];
```

```
speechRecognition.EngineModelType =@"16k_zh";
speechRecognition.ChannelNum = 1;
speechRecognition.ResTextFormat = 0;
speechRecognition.ConvertNumMode = 0;
// 当 Tag 为 SpeechRecognition 时有效，指定该任务的参数
op.SpeechRecognition = speechRecognition;
// 操作规则
taskInfo.Operation = op;
// 语音识别任务
request.taskInfo = taskInfo;

[request setFinishBlock:^(QCloudPostAudioDiscernTaskResult * _Nullable result, NSError * _Nullable error) {
// 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
// QCloudPostAudioDiscernTaskResult 类;
}];
[[QCloudCOSXMLService defaultCOSXML] PostAudioDiscernTask:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudPostAudioDiscernTaskRequest.init();

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";
request.regionName = "regionName";

let taskInfo = QCloudPostAudioDiscernTaskInfo.init();
taskInfo.tag = "SpeechRecognition";

// 队列 ID ,通过查询语音识别队列获取
taskInfo.queueId = "QueueId";
// 操作规则
let input = QCloudPostAudioDiscernTaskInfoInput.init();
input.object = "test1";
// 待操作的语音文件
taskInfo.input = input;
let op = QCloudPostAudioDiscernTaskInfoOperation.init();
let output = QCloudPostAudioDiscernTaskInfoOutput.init();
output.region = "regionName";
output.bucket = "examplebucket-1250000000";
output.object = "test";
// 结果输出地址
op.output = output;

let speechRecognition = QCloudPostAudioDiscernTaskInfoSpeechRecognition.init();
speechRecognition.engineModelType = "16k_zh";
speechRecognition.channelNum = 1;
speechRecognition.resTextFormat = 0;
speechRecognition.convertNumMode = 0;
// 当 Tag 为 SpeechRecognition 时有效，指定该任务的参数
op.speechRecognition = speechRecognition;
// 操作规则
```

```
taskInfo.operation = op;
// 语音识别任务
request.taskInfo = taskInfo;

[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
// QCloudPostAudioDiscernTaskResult 类；
})];
QCloudCOSXMLService.defaultCOSXML().postAudioDiscernTask(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

查询指定的语音识别任务

功能说明

用于查询指定的语音识别任务。

> 注意：
> COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudGetAudioDiscernTaskRequest * request = [[QCloudGetAudioDiscernTaskRequest alloc]init];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// QCloudPostAudioDiscernTaskRequest接口返回的jobid
request.jobId = @"jobid";

request.regionName = @"regionName";

request.finishBlock = ^(QCloudGetAudioDiscernTaskResult * outputObject, NSError *error) {
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudGetAudioDiscernTaskResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetAudioDiscernTask:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudGetAudioDiscernTaskRequest.init();

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

// QCloudPostAudioDiscernTaskRequest接口返回的jobid
request.jobId = "jobid";
```

```
request.regionName = "regionName";

request.setFinish { outputObject, error in
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudGetAudioDiscernTaskResult 类;
};
QCloudCOSXMLService.defaultCOSXML().getAudioDiscernTask(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

批量拉取语音识别任务

功能说明

用于拉取符合条件的语音识别任务。

> 注意：
> COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudBatchGetAudioDiscernTaskRequest * request = [[QCloudBatchGetAudioDiscernTaskRequest alloc]init];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

// 拉取该队列 ID 下的任务。
// 通过查询语音识别队列获取
request.queueId = @"queueId";

request.regionName = @"regionName";

request.states = QCloudTaskStatesSuccess | QCloudTaskStatesCancel;

// 其他更多参数请查阅sdk文档或源码注释

request.finishBlock = ^(QCloudBatchGetAudioDiscernTaskResult * outputObject, NSError *error) {
// outputObject 任务结果，详细字段请查看api文档或者SDK源码
// QCloudBatchGetAudioDiscernTaskResult 类;
};
[[QCloudCOSXMLService defaultCOSXML] BatchGetAudioDiscernTask:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudBatchGetAudioDiscernTaskRequest.init();

// 存储桶名称，格式为 BucketName-APPID
```

```
request.bucket = "examplebucket-1250000000";

// 拉取该队列 ID 下的任务。
// 通过查询语音识别队列获取
request.queueId = "queueId";

request.regionName = "regionName";

request.states = QCloudTaskStatesEnum(rawValue: QCloudTaskStatesEnum.success.rawValue | QCloudTaskStatesEnum.cancel.rawValue)!;

// 其他更多参数请查阅sdk文档或源码注释

request.setFinish { outputObject, error in
// outputObject 任务结果，详细字段请查看api文档或者SDK源码
// QCloudBatchGetAudioDiscernTaskResult 类；
};
QCloudCOSXMLService.defaultCOSXML().batchGetAudioDiscernTask(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

语音 bucket 接口

最近更新时间：2022-08-25 09:07:33

简介

本文档提供关于查询语音识别开通状态相关的 API 概览以及 SDK 示例代码。

API	说明
查询语音识别开通状态	接口用于查询存储桶是否已开通语音识别功能。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询语音识别开通状态

功能说明

接口用于查询存储桶是否已开通语音识别功能。

> ⚠ 注意：
> COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudGetAudioDiscernOpenBucketListRequest * request = [[QCloudGetAudioDiscernOpenBucketListRequest alloc]init];

// 存储桶名称前缀，前缀搜索
request.bucketName = @"examplebucket-1250000000";

request.regionName = @"regionName";
// 地域信息，以“,”分隔字符串，支持 All、ap-shanghai、ap-beijing
request.regions = @"regions";

request.finishBlock = ^(QCloudGetAudioOpenBucketListResult * outputObject, NSError *error) {
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudGetAudioOpenBucketListResult 类;
};
[[QCloudCOSXMLService defaultCOSXML] GetAudioDiscernOpenBucketList:request];
```

> ? 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudGetAudioDiscernOpenBucketListRequest.init();

// 存储桶名称前缀，前缀搜索
request.bucketName = "examplebucket-1250000000";

request.regionName = "regionName";
// 地域信息，以“,”分隔字符串，支持 All、ap-shanghai、ap-beijing
```

版权所有：腾讯云计算（北京）有限责任公司

```
request.regions = "regions";

request.setFinish { outputObject, error in
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudGetAudioOpenBucketListResult 类;
}
QCloudCOSXMLService.defaultCOSXML().getAudioDiscernOpenBucketList(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

队列接口

最近更新时间：2022-08-25 09:07:36

简介

本文档提供关于查询语音识别队列、更新语音识别队列 相关的 API 概览以及 SDK 示例代码。

API	说明
查询语音识别队列	接口用于查询语音识别队列。
更新语音识别队列	接口用于更新语音识别队列。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询语音识别队列

功能说明

接口用于查询语音识别队列。

> 注意：
> COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudGetAudioDiscernTaskQueueRequest * request = [[QCloudGetAudioDiscernTaskQueueRequest alloc]init];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

request.regionName = @"regionName";
// 队列 ID，以“,”符号分割字符串
request.queueIds = @"1,2,3";

// 1. Active 表示队列内的作业会被语音识别服务调度执行
// 2. Paused 表示队列暂停，作业不再会被语音识别服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于识别中的任务将继续执行，不受影响
request.state = 1;

request.finishBlock = ^(QCloudAudioAsrqueueResult * 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
// QCloudAudioAsrqueueResult 类;
};
[[QCloudCOSXMLService defaultCOSXML] GetAudioDiscernTaskQueue:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudGetAudioDiscernTaskQueueRequest.init();
```

```
// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

request.regionName = "regionName";
// 队列 ID，以“,”符号分割字符串
request.queueIds = "1,2,3";

// 1. Active 表示队列内的作业会被语音识别服务调度执行
// 2. Paused 表示队列暂停，作业不再会被语音识别服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于识别中的任务将继续执行，不受影响
request.state = 1;

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
// QCloudAudioAsrqueueResult 类;
};
QCloudCOSXMLService.defaultCOSXML().getAudioDiscernTaskQueue(request);
```

说明：
更多完整示例，请前往 GitHub 查看。

更新语音识别队列

功能说明

接口用于更新语音识别队列。

注意：
COS iOS SDK 版本需要大于等于 v6.1.3。

示例代码

Objective-C

```
QCloudUpdateAudioDiscernTaskQueueRequest * request = [[QCloudUpdateAudioDiscernTaskQueueRequest alloc]init];

// 存储桶名称，格式为 BucketName-APPID
request.bucket = @"examplebucket-1250000000";

request.regionName = @"regionName";
// 模板名称
request.name = @"name";
// 1. Active 表示队列内的作业会被语音识别服务调度执行
// 2. Paused 表示队列暂停，作业不再会被语音识别服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于识别中的任务将继续执行，不受影响
request.state = 1;
// 管道 ID
request.queueID = @"queueID";

// 其他更多参数请查看sdk文档或源码注释

request.finishBlock = ^(QCloudAudioAsrqueueUpdateResult * outputObject, NSError *error) {
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudAudioAsrqueueUpdateResult 类;
};
[[QCloudCOSXMLService defaultCOSXML] UpdateAudioDiscernTaskQueue:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request = QCloudUpdateAudioDiscernTaskQueueRequest.init();

// 存储桶名称，格式为 BucketName-APPID
request.bucket = "examplebucket-1250000000";

request.regionName = "regionName";
// 模板名称
request.name = "name";
// 1. Active 表示队列内的作业会被语音识别服务调度执行
// 2. Paused 表示队列暂停，作业不再会被语音识别服务调度执行，队列内的所有作业状态维持在暂停状态，已经处于识别中的任务将继续执行，不受影响
request.state = 1;
// 管道 ID
request.queueID = "queueID";

// 其他更多参数请查看sdk文档或源码注释

request.setFinish { outputObject, error in
// outputObject 详细字段请查看api文档或者SDK源码
// QCloudAudioAsrqueueUpdateResult 类;
};
QCloudCOSXMLService.defaultCOSXML().updateAudioDiscernTaskQueue(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

图片处理
盲水印

最近更新时间：2022-08-25 09:07:42

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。

关于盲水印 API 文档，请参见 盲水印。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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

Objective-C

```
QCloudPutObjectWatermarkRequest* put = [QCloudPutObjectWatermarkRequest new];

// 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"
put.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.bucket = @"examplebucket-1250000000";
put.body = [@"123456789" dataUsingEncoding:NSUTF8StringEncoding];
QCloudPicOperations * op = [[QCloudPicOperations alloc]init];

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = NO;
QCloudPicOperationRule * rule = [[QCloudPicOperationRule alloc]init];

// 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
rule.fileid = @"test";

// 盲水印图片在cos上的地址：如 http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/protection_blind_watermark_icon.png
rule.imageURL = @"watermarkURL"; // 水印文字只能是 [a-zA-Z0-9]
//操作：有效值 :QCloudPicOperationRuleActionPut:添加盲水印 QCloudPicOperationRuleActionExtrac:提取盲水印
rule.actionType =QCloudPicOperationRuleActionPut;
// 盲水印类型，有效值：QCloudPicOperationRuleHalf 半盲；QCloudPicOperationRuleFull: 全盲；QCloudPicOperationRuleText 文字
rule.type = QCloudPicOperationRuleFull;
op.rule = @[rule];
put.picOperations = op;
[put setFinishBlock:^(QCloudPutObjectWatermarkResult *outputObject, NSError *error) {

}];
[[QCloudCOSXMLService defaultCOSXML] PutWatermarkObject:put];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let put = QCloudPutObjectWatermarkRequest<AnyObject>();

// 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"
put.object = "exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.bucket = "examplebucket-1250000000";
put.body = "123456789".data(using: .utf8)! as NSData;
let op = QCloudPicOperations.init();

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = false;

let rule = QCloudPicOperationRule.init();

// 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

rule.fileid = "test";
//操作：有效值 :QCloudPicOperationRuleActionPut:添加盲水印 QCloudPicOperationRuleActionExtrac:提取盲水印
rule.actionType = .put;

// 盲水印类型，有效值：QCloudPicOperationRuleHalf 半盲；QCloudPicOperationRuleFull: 全盲；QCloudPicOperationRuleText 文字
rule.type = .full;
// 盲水印图片在cos上的地址：如 http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/watermark_icon.png
rule.imageURL = "watermarkURL";
op.rule = [rule];
put.picOperations = op;
put.setFinish { (outoutObject, error) in

};
QCloudCOSXMLService.defaultCOSXML().putWatermarkObject(put);
```

> **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

Objective-C

```
QCloudGetObjectRequest * request = [QCloudGetObjectRequest new];

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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"
request.object = @"exampleobject";

//处理参数，规则参见：https://cloud.tencent.com/document/product/436/46782
request.watermarkRule = @"watermark/3/type/2/image/aHR0cDovL2NpLTEyNTM2NTMzNjcuY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vcHJvdGVjdGlvbl9ibGluZF93YXRlcm1hcmtfaWNvbi5wbmc=";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = [NSURL fileURLWithPath:@"Local File Path"];
```

```
// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

// 监听下载结果
[request setFinishBlock:^(id outputObject, NSError *error) {
// outputObject 包含所有的响应 http 头部
NSDictionary* info = (NSDictionary *) outputObject;
}];

// 监听下载进度
[request setDownProcessBlock:^(int64_t bytesDownload,
int64_t totalBytesDownload,
int64_t totalBytesExpectedToDownload) {

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}];

[[QCloudCOSXMLService defaultCOSXML] defaultCOSXML:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudGetObjectRequest = QCloudGetObjectRequest();

// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
request.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"
request.object = "exampleobject";

// 设置下载的路径 URL，如果设置了，文件将会被下载到指定路径中
request.downloadingURL = NSURL.fileURL(withPath: "Local File Path") as URL?;

// 本地已下载的文件大小，如果是从头开始下载，请不要设置
request.localCacheDownloadOffset = 100;

//处理参数，规则参见：https://cloud.tencent.com/document/product/436/46782
request.watermarkRule = @"watermark/3/type/2/image/aHR0cDovL2NpLTEyNTM2NTMzNjcuY29zLmFwLWd1YW5nemhvdS5teXFjbG91ZC5jb20vcHJvdGVjdGlvbl9ibGluZF93YXRlcm1hcmtfaWNvbi5wbmc=";

// 监听下载进度
request.sendProcessBlock = { (bytesDownload, totalBytesDownload,
totalBytesExpectedToDownload) in

// bytesDownload 新增字节数
// totalBytesDownload 本次下载接收的总字节数
// totalBytesExpectedToDownload 本次下载的目标字节数
}

// 监听下载结果
```

```
request.finishBlock = { (result, error) in
if let result = result {
// result 包含响应的 header 信息
} else {
print(error!);
}
}

QCloudCOSXMLService.defaultCOSXML().getObject(request);
```

说明：

更多完整示例，请前往 GitHub 查看。

基础图片处理

最近更新时间：2022-08-25 09:07:46

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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

Objective-C

```
QCloudPutObjectWatermarkRequest* put = [QCloudPutObjectWatermarkRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
put.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.bucket = @"examplebucket-1250000000";

put.body = [@"123456789" dataUsingEncoding:NSUTF8StringEncoding];
QCloudPicOperations * op = [[QCloudPicOperations alloc]init];

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = NO;
QCloudPicOperationRule * rule = [[QCloudPicOperationRule alloc]init];

// 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
rule.fileid = @"test";
```

```
// 盲水印文字，需要经过 URL 安全的 Base64 编码。当 type 为3时必填，type
rule.text = @"123"; // 水印文字只能是 [a-zA-Z0-9]

// 盲水印类型，有效值：1 半盲；2 全盲；3 文字
rule.type = QCloudPicOperationRuleText;
op.rule = @[rule];
put.picOperations = op;
[put setFinishBlock:^(id outputObject, NSError *error) {

}];
[[QCloudCOSXMLService defaultCOSXML] PutWatermarkObject:put];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let put = QCloudPutObjectWatermarkRequest<AnyObject>();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
put.object = "exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket

put.bucket = "examplebucket-1250000000";
put.body = "123456789".data(using: .utf8)! as NSData;
let op = QCloudPicOperations.init();

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = false;

let rule = QCloudPicOperationRule.init();

// 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

rule.fileid = "test";

// 盲水印文字，需要经过 URL 安全的 Base64 编码。当 type 为3时必填，type 为1或2时无效。
rule.text = "123";

// 盲水印类型，有效值：1 半盲；2 全盲；3 文字
rule.type = .text;

op.rule = [rule];
put.picOperations = op;
put.setFinish { (outoutObject, error) in

};
QCloudCOSXMLService.defaultCOSXML().putWatermarkObject(put);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

云上数据处理

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

Objective-C

```
QCloudCICloudDataOperationsRequest* put = [QCloudCICloudDataOperationsRequest new];

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
put.object = @"exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
put.bucket = @"examplebucket-1250000000";

QCloudPicOperations * op = [[QCloudPicOperations alloc]init];

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = NO;
QCloudPicOperationRule * rule = [[QCloudPicOperationRule alloc]init];

// 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
rule.fileid = @"test";

// 盲水印文字，需要经过 URL 安全的 Base64 编码。当 type 为3时必填，type
rule.text = @"123"; // 水印文字只能是 [a-zA-Z0-9]

// 盲水印类型，有效值：1 半盲；2 全盲；3 文字
rule.type = QCloudPicOperationRuleText;
op.rule = @[rule];
put.picOperations = op;
[put setFinishBlock:^(QCloudPutObjectWatermarkResult *outputObject, NSError *error) {

}];
[[QCloudCOSXMLService defaultCOSXML] CloudDataOperations:put];
```

> **说明：**
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let put = QCloudCICloudDataOperationsRequest<AnyObject>();

// 对象键，是对象在 COS 上的完整路径，如果带目录的话，格式为 "video/xxx/movie.mp4"
put.object = "exampleobject";
// 存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket

put.bucket = "examplebucket-1250000000";
let op = QCloudPicOperations.init();

// 是否返回原图信息。0表示不返回原图信息，1表示返回原图信息，默认为0
op.is_pic_info = false;

let rule = QCloudPicOperationRule.init();
```

```
// 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

rule.fileid = "test";

// 盲水印文字，需要经过 URL 安全的 Base64 编码。当 type 为3时必填，type 为1或2时无效。
rule.text = "123";

// 盲水印类型，有效值：1 半盲；2 全盲；3 文字
rule.type = .text;

op.rule = [rule];
put.picOperations = op;
put.setFinish { (outoutObject, error) in

};
QCloudCOSXMLService.defaultCOSXML().cloudDataOperations(put);
```

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：2022-08-25 09:07:50

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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 iOS SDK 版本需要大于等于 v5.9.6。

请求示例

Objective-C

```
QCloudGetDescribeMediaBucketsRequest * request = [[QCloudGetDescribeMediaBucketsRequest alloc]init];

// 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
request.regions = @[@"ap-shanghai"];
// 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
request.bucketNames = @[@"examplebucket-1250000000"];
// 存储桶名称前缀，前缀搜索
request.bucketName = @"examplebucket-1250000000";
// 第几页
request.pageNumber = 0;
// 每页个数
request.pageSize = 100;

request.finishBlock = ^(QCloudDescribeMediaInfo * outputObject, NSError *error) {
// outputObject 请求到的媒体信息，详细字段请查看api文档或者SDK源码
// QCloudDescribeMediaInfo 类；
};
[[QCloudCOSXMLService defaultCOSXML] CIGetDescribeMediaBuckets:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudGetDescribeMediaBucketsRequest = QCloudGetDescribeMediaBucketsRequest();
// 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
```

```
request.regions = ["ap-shanghai"];
// 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
request.bucketNames = ["bucketNames"];
// 存储桶名称前缀，前缀搜索
request.bucketName = "bucketName";
// 第几页
request.pageNumber = 0;
// 每页个数
request.pageSize = 100;

request.finishBlock = { (result, error) in
// result 请求到的媒体信息，详细字段请查看api文档或者SDK源码
// QCloudMediaInfo 类;
}
QCloudCOSXMLService.defaultCOSXML().ciGetDescribeMediaBuckets(request);
```

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：2022-08-25 09:07:54

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

查询文件信息

功能说明

用于查询媒体文件的信息。

> ⚠ 注意：
> COS iOS SDK 版本需要大于等于 v5.9.6。

请求示例

Objective-C

```
//.cssg-snippet-body-start:[objc-get-media-info]
QCloudGetMediaInfoRequest * request = [[QCloudGetMediaInfoRequest alloc]init];
// 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";
request.finishBlock = ^(QCloudMediaInfo * outputObject, NSError *error) {
// outputObject 请求到的媒体信息，详细字段请查看api文档或者SDK源码
// QCloudMediaInfo 类；
};
[[QCloudCOSXMLService defaultCOSXML] CIGetMediaInfo:request];
```

> ⓘ 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudGetMediaInfoRequest = QCloudGetMediaInfoRequest();
// 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";

request.finishBlock = { (result, error) in
// outputObject 请求到的媒体信息，详细字段请查看api文档或者SDK源码
// QCloudMediaInfo 类；
```

```
}
QCloudCOSXMLService.defaultCOSXML().ciGetMediaInfo(request);
```

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：2022-08-25 09:07:57

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

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

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 iOS SDK 版本需要大于等于 v5.9.6。

请求示例

Objective-C

```
QCloudGetGenerateSnapshotRequest * request = [[QCloudGetGenerateSnapshotRequest alloc]init];

// 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";
// 截图配置信息
request.generateSnapshotConfiguration = [QCloudGenerateSnapshotConfiguration new];
// 截取哪个时间点的内容，单位为秒 必传
request.generateSnapshotConfiguration.time = 10;
// 截图的宽。默认为0
request.generateSnapshotConfiguration.width = 100;
// 截图的宽。默认为0
request.generateSnapshotConfiguration.height = 100;

// 截帧方式:枚举值
// GenerateSnapshotModeExactframe：截取指定时间点的帧
// GenerateSnapshotModeKeyframe：截取指定时间点之前的最近的
// 默认值为 exactframe
request.generateSnapshotConfiguration.mode = QCloudGenerateSnapshotModeExactframe;

// 图片旋转方式:枚举值
// GenerateSnapshotRotateTypeAuto：按视频旋转信息进行自动旋转
// GenerateSnapshotRotateTypeOff：不旋转
// 默认值为 auto
request.generateSnapshotConfiguration.rotate = QCloudGenerateSnapshotRotateTypeAuto;

// 截图的格式:枚举值
// GenerateSnapshotFormatJPG：jpg
```

```
// GenerateSnapshotFormatPNG：png
// 默认 jpg
request.generateSnapshotConfiguration.format = QCloudGenerateSnapshotFormatJPG;

request.finishBlock = ^(QCloudGenerateSnapshotResult * outputObject, NSError *error) {
// outputObject 截图信息，详细字段请查看api文档或者SDK源码
// QCloudGenerateSnapshotResult 类；
};
[[QCloudCOSXMLService defaultCOSXML] GetGenerateSnapshot:request];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let request : QCloudGetGenerateSnapshotRequest = QCloudGetGenerateSnapshotRequest();
// 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 = "test.mp4";
// 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
request.bucket = "CURRENT_BUCKET";
// 截图配置信息
request.generateSnapshotConfiguration = QCloudGenerateSnapshotConfiguration();
// 截取哪个时间点的内容，单位为秒 必传
request.generateSnapshotConfiguration.time = 10;
// 截图的宽。默认为0
request.generateSnapshotConfiguration.width = 100;
// 截图的宽。默认为0
request.generateSnapshotConfiguration.height = 100;

// 截帧方式:枚举值
// GenerateSnapshotModeExactframe：截取指定时间点的帧
// GenerateSnapshotModeKeyframe：截取指定时间点之前的最近的
// 默认值为 exactframe
request.generateSnapshotConfiguration.mode = QCloudGenerateSnapshotMode.exactframe;

// 图片旋转方式:枚举值
// GenerateSnapshotRotateTypeAuto：按视频旋转信息进行自动旋转
// GenerateSnapshotRotateTypeOff：不旋转
// 默认值为 auto
request.generateSnapshotConfiguration.rotate = QCloudGenerateSnapshotRotateType.auto;

// 截图的格式:枚举值
// GenerateSnapshotFormatJPG：jpg
// GenerateSnapshotFormatPNG：png
// 默认 jpg
request.generateSnapshotConfiguration.format = QCloudGenerateSnapshotFormat.JPG;

request.finishBlock = { (result, error) in
// result 截图信息，详细字段请查看api文档或者SDK源码
// QCloudGenerateSnapshotResult 类；
}
QCloudCOSXMLService.defaultCOSXML().getGenerateSnapshot(request);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

内容审核

图片审核

版权所有：腾讯云计算（北京）有限责任公司

最近更新时间：2022-10-14 15:42:15

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 alloc]init];

// 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 | QCloudRecognitionPolitics | QCloudRecognitionAds;
[request setFinishBlock:^(QCloudImageRecognitionResult * _Nullable result, NSError * _Nullable error) {
// 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];
```

版权所有：腾讯云计算（北京）有限责任公司

> 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!

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 alloc]init];

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 | QCloudRecognitionPolitics | QCloudRecognitionAds;
[request setFinishBlock:^(QCloudBatchImageRecognitionResult * _Nullable result, NSError * _Nullable error) {
// 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!
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 alloc]init];

// 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：2022-12-07 09:05:13

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

```
QCloudPostVideoRecognitionRequest * request = [[QCloudPostVideoRecognitionRequest alloc]init];

// 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。
```

```
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/product/460/56345
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和视频声音。默认值为 0。
request.detectContent = YES;

request.finishBlock = ^(QCloudPostVideoRecognitionResult * outputObject, NSError *error) {
// 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。

Swift

```
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.rawValue | QCloudRecognitionEnum.ads.rawValue!

// 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。
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/product/460/56345
```

```
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和视频声音。默认值为0。
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 SDK 源码
// 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

```
QCloudGetVideoRecognitionRequest * request = [[QCloudGetVideoRecognitionRequest alloc]init];

// 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 * _Nullable 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
// QCloudVideoRecognitionResult 类;
}];
[[QCloudCOSXMLService defaultCOSXML] GetVideoRecognition:request];
```

说明：
更多完整示例，请前往 GitHub 查看。

Swift

```
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：2022-10-14 15:42:35

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 alloc]init];

// 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（广告识别）四种，
// 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 *error) {
// 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（广告识别）四种，
// 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!

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 alloc]init];

// 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：2022-10-14 15:42:45

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 alloc]init];

// 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（广告识别）四种，
// 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 | QCloudRecognitionPolitics | QCloudRecognitionTerrorist;

// 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/product/460/56345
request.bizType = BizType;

request.finishBlock = ^(QCloudPostTextRecognitionResult * outputObject, NSError *error) {
```

```
// 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（广告识别）四种，
// 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!

// 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/product/460/56345
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]init];

// 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：2022-10-14 15:42:53

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]init];

// 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（广告识别）四种，
// 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 | QCloudRecognitionPolitics | QCloudRecognitionTerrorist;

// 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/product/460/56345
request.bizType = BizType;
```

```
request.finishBlock = ^(QCloudPostDocRecognitionResult * outputObject, NSError *error) {
// 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（广告识别）四种，
// 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!

// 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/product/460/56345
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]init];

// 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：2022-10-14 15:43:03

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]init];

// 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（广告识别）四种，
// 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 | QCloudRecognitionPolitics | QCloudRecognitionTerrorist;

// 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/product/460/56345
request.bizType = BizType;

request.finishBlock = ^(QCloudPostWebRecognitionResult * outputObject, NSError *error) {
// 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（广告识别）四种，
// 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.rawValue | QCloudRecognitionEnum.ads.rawValue)!

// 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/product/460/56345
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]init];

// 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。

设置自定义头部

最近更新时间：2022-08-25 09:08:29

简介

本文档主要介绍 SDK 如何在请求时携带自定义头部。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

功能说明

COS 在上传对象时可以携带以 x-cos-meta- 开头的自定义头部，包括用户自定义元数据头部后缀和用户自定义元数据信息，这些头部将作为对象元数据保存。

如果您开通了万象服务，可以携带 Pic-Operations 头部，实现后台自动图片处理，详细的 API 说明请参考 数据万象持久化。

示例代码

Objective-C

```
request.customHeaders[@"custom-key"] = @"custom-value";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
request.customHeaders["custom-key"] = "custom-value";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

设置访问域名（CDN/全球加速）

最近更新时间：2022-08-25 09:08:32

简介

本文档提供关于如何使用非默认域名请求 COS 服务。

SDK API 参考

SDK 所有接口的具体参数与方法说明，请参考 SDK API。

CDN 默认加速域名

关于如何开启默认加速域名请参考 开启默认 CDN 加速域名。

以下代码展示了如何使用默认加速域名访问 COS 服务。

示例代码

Objective-C

```
QCloudCOSXMLEndPoint *endpoint = [[QCloudCOSXMLEndPoint alloc] init];
endpoint.suffix = @"file.myqcloud.com";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let endpoint = QCloudCOSXMLEndPoint();
endpoint.suffix = "file.myqcloud.com";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

CDN 自定义加速域名

关于如何开启 CDN 自定义加速域名请参考 开启自定义 CDN 加速域名。

以下代码展示了如何使用自定义加速域名访问 COS 服务。

示例代码

Objective-C

```
QCloudCOSXMLEndPoint *endpoint = [[QCloudCOSXMLEndPoint alloc] initWithLiteralURL:[NSURL URLWithString:@"exampledomain.com"]];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let endpoint = QCloudCOSXMLEndPoint.init(literalURL: NSURL.init(string: "exampledomain.com") as URL?);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

自定义源站域名

关于如何设置自定义源站域名请参考 自定义源站域名。

以下代码展示了如何使用自定义源站域名访问 COS 服务。

示例代码

Objective-C

```
NSString *customDomain = @"exampledomain.com"; // 自定义加速域名
QCloudCOSXMLEndPoint *endpoint = [[QCloudCOSXMLEndPoint alloc] initWithLiteralURL:[NSURL URLWithString:customDomain]];
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let endpoint = QCloudCOSXMLEndPoint.init(literalURL: NSURL.init(string: "exampledomain.com") as URL?);
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

全球加速域名

关于全球加速功能请参考 全球加速功能概述。

以下代码展示了如何使用全球加速域名访问 COS 服务。

示例代码

Objective-C

```
QCloudCOSXMLEndPoint *endpoint = [[QCloudCOSXMLEndPoint alloc]init];
endpoint.suffix = @"cos.accelerate.myqcloud.com";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

Swift

```
let endpoint = QCloudCOSXMLEndPoint();
endpoint.suffix = "cos.accelerate.myqcloud.com";
```

> 说明：
> 更多完整示例，请前往 GitHub 查看。

异常处理

最近更新时间：2022-08-25 09:08:36

当 SDK 请求失败的时候，返回的 error 将不为空，并且包括了错误码、错误描述和其它一些调试必备的信息，以帮助开发者快速解决问题。返回错误码（封装在返回的 error 里）主要包括两类：客户端错误和服务端错误。

客户端错误

- 对于设备本身因为网络原因产生的错误码，都是负数并且是四位数，例如-1001，这类错误码由苹果公司定义，可以参考 Foundation 框架中的 NSURLError.h 头文件内的定义，或者是 苹果官方文档说明。
- 对于腾讯云 SDK 网络层本地客户端自定义错误：主要是指网络异常、证书无效、参数校验失败等，如下表所示：

错误码	错误信息	错误描述
10000	InvalidArgument	参数错误
10001	InvalidCredentials	证书无效
10004	UnsupportOperation	无法支持的操作
20001	InvalidArgument	服务器返回了不合法的数据
20004	PoorNetwork	数据完整性校验失败
30000	UserCancelled	用户取消
30002	AlreadyFinished	任务已完成

服务端错误

对于 COS 返回的错误码，是基于 HTTP 的状态码而来的，也就是404、503这类。对于这类错误码，请参见 错误码 文档寻求解决方案。您也可以使用自助诊断工具，调试报错代码，快速定位问题。

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 COS 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 COS 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。

升级到 XML iOS SDK

最近更新时间：2022-08-25 09:08:40

如果您细心对比过 JSON iOS SDK 和 XML SDK 的文档，您会发现并不是一个简单的增量更新。XML iOS SDK 不仅在架构、可用性和安全性上有了非常大的提升，而且在易用性、健壮性和传输性能上也做了非常大的改进。如果您想要升级到 XML iOS SDK，请参考下面的指引，一步步完成 SDK 的升级工作。

功能对比

下表列出了 JSON SDK 和 XML SDK 的主要功能对比：

功能	XML SDK	JSON SDK
文件上传	支持本地文件、二进制数据、分块上传 默认覆盖上传 智能判断上传模式 简单上传最大支持5GB 分块上传最大支持48.82TB（50,000GB）	只支持本地文件上传 可选择是否覆盖 需要手动选择是简单还是分块上传 简单上传最大支持20MB 分块上传最大支持64GB
文件删除	支持批量删除	只支持单文件删除
存储桶基本操作	创建存储桶 获取存储桶 删除存储桶	不支持
存储桶ACL操作	设置存储桶ACL 获取设置存储桶ACL 删除设置存储桶ACL	不支持
存储桶生命周期	创建存储桶生命周期 获取存储桶生命周期 删除存储桶生命周期	不支持
目录操作	不单独提供接口	创建目录 查询目录 删除目录

升级步骤

请按照下面5个步骤升级 iOS SDK。

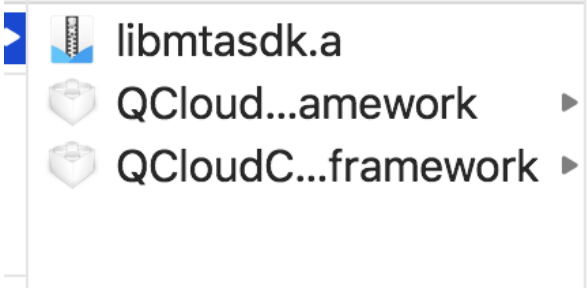
1. 更新 iOS SDK

您可以通过 cocoapods 或下载打包好的动态库的方式来集成 SDK。在这里我们推荐您使用 cocoapods 的方式来进行导入。

- **使用 Cocoapods 导入（推荐）**
  在 Podfile 文件中使用：

```
pod 'QCloudCOSXML'
```

- **使用打包好的动态库导入（手动集成方式）**
  您可以从 realease 中选择需要的版本进行下载
  将 QCloudCOSXML.framework, QCloudCore.framework 和 libmtasdk.a 拖入到工程中，如下图所示：



并添加以下依赖库：

- CoreTelephony
- Foundation
- SystemConfiguration
- libc++.tbd

工程配置

完成上述步骤之后，在 Build Settings 中设置 Other Linker Flags，加入以下参数：

```
-ObjC
-all_load
```

如下图所示：



2. 更改 SDK 鉴权方式

在 JSON SDK 中您需要自己在后台计算好签名，再返回客户端使用。而 XML SDK 使用了新的鉴权算法，我们强烈建议您后台接入我们的临时密钥（STS）方案。该方案您不需要了解签名计算过程，只需要在服务器端接入 CAM，将拿到的临时密钥返回到客户端，并设置到 SDK 中，SDK 会负责管理密钥和计算签名。临时密钥在一段时间后会自动失效，而永久密钥不会泄露。
您还可以按照不同的粒度来控制访问权限。具体的步骤请参考 移动应用直传实践 以及 临时密钥生成及使用指引。

如果您仍然采用后台手动计算签名，再返回客户端使用的方式，请注意我们的签名算法发生了改变。签名不再区分单次和多次签名，而是通过设置签名的有效期来保证安全性。请参考 XML 请求签名 文档更新您签名的实现。

3. 更改 SDK 初始化方式

XML SDK 的初始化接口发生了一些变化：

- QCloudCOSXMLService 代替了 COSClient，但两者作用相同。同时增加了QCloudServiceConfiguration来配置更多的信息。
- 您需要在初始化时实例化一个密钥提供者 QCloudAuthentationV5Creator，用于提供一个有效的密钥，建议使用临时密钥。

JSON SDK 的初始化方式如下：

```
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:@“sh”];
```

XML SDK 的初始化方式如下：

> 说明：
> 示例代码中给出的是通过使用临时密钥的方式获取签名：强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确

```
//AppDelegate.m
//第一步：注册默认的 COS 服务
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
configuration.appID = @"1250000000";
configuration.signatureProvider = self;
QCloudCOSXMLEndPoint* endpoint = [[QCloudCOSXMLEndPoint alloc] init];
```

```
endpoint.regionName = @"COS_REGION";//服务地域名称，可用的地域请参考注释
configuration.endpoint = endpoint;
[QCloudCOSXMLService registerDefaultCOSXMLWithConfiguration:configuration];
[QCloudCOSTransferMangerService registerDefaultCOSTransferMangerWithConfiguration:configuration];
return YES;
}

//第二步：实现 QCloudSignatureProvider 协议
//实现签名的过程，我们推荐在服务器端实现签名的过程，详情请参考接下来的“生成签名”这一章。
```

```
//AppDelegate.m

//这里定义一个成员变量 @property (nonatomic) QCloudCredentailFenceQueue* credentialFenceQueue;

- (void) fenceQueue:(QCloudCredentailFenceQueue * )queue requestCreatorWithContinue:(QCloudCredentailFenceQueueContinue)continueBloc
k
{
QCloudCredential* credential = [QCloudCredential new];
//在这里可以同步过程从服务器获取临时签名需要的 secretID，secretKey，expiretionDate 和 token 参数
credential.secretID = @"COS_SECRETID";
credential.secretKey = @"COS_SECRETKEY";
/*强烈建议返回服务器时间作为签名的开始时间，用来避免由于用户手机本地时间偏差过大导致的签名不正确 */
credential.startDate = [[[NSDateFormatter alloc] init] dateFromString:@"startTime"]; // 单位是秒
credential.experationDate = [[[NSDateFormatter alloc] init] dateFromString:@"expiredTime"];
credential.token = @"COS_TOKEN";
QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc]
initWithCredential:credential];
continueBlock(creator, nil);
}

- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
[self.credentialFenceQueue performAction:^(QCloudAuthentationCreator *creator, NSError *error) {
if (error) {
continueBlock(nil, error);
} else {
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
}];
}
```

4. 更改存储桶名称和可用区域简称

XML SDK 的存储桶名称和可用区域简称与 JSON SDK 的不同，需要您进行相应的更改。

存储桶 Bucket

XML SDK 存储桶名称由两部分组成：用户自定义字符串 和 APPID，两者以中划线“-”相连。例如 examplebucket-1250000000 ，其中 examplebucket 为用户自定义字符串， 1250000000 为 APPID。

> **说明：**

> APPID 是腾讯云账户的账户标识之一，用于关联云资源。在用户成功申请腾讯云账户后，系统自动为用户分配一个 APPID。您可通过在 账号信息 控制台查看 APPID。

在设置 Bucket 时，请参考下面的示例代码：

```
NSString *bucket = "examplebucket-1250000000";
```

存储桶可用区域简称 Region

XML SDK 的存储桶可用区域简称发生了变化，下表列出了不同区域在 JSON SDK 和 XML SDK 中的对应关系：

地域	XML SDK 地域简称	JSON SDK 地域简称
北京一区（华北）	ap-beijing-1	tj
北京	ap-beijing	bj
上海（华东）	ap-shanghai	sh
广州（华南）	ap-guangzhou	gz
成都（西南）	ap-chengdu	cd
重庆	ap-chongqing	无
香港	ap-hongkong	hk
新加坡	ap-singapore	sgp
多伦多	na-toronto	ca
法兰克福	eu-frankfurt	ger
孟买	ap-mumbai	无
首尔	ap-seoul	无
硅谷	na-siliconvalley	无
弗吉尼亚	na-ashburn	无
曼谷	ap-bangkok	无
莫斯科	eu-moscow	无

在初始化时，请将存储桶所在区域简称设置到 QCloudServiceConfiguration 的 regionName 中。

5. 更改 API

升级到 XML SDK之后，一些操作的 API 发生了变化，请您根据实际需求进行相应的更改。我们做了封装让 SDK 更加易用，具体请参考我们的示例和 快速入门 文档。

API 变化有以下三点：

（1）没有单独的目录接口

在 XML SDK 中，不再提供单独的目录接口。对象存储中本身没有文件夹和目录的概念，对象存储不会因为上传对象 project/a.txt 而创建一个 project 文件夹。为了满足用户使用习惯，对象存储在控制台、COS browser 等图形化工具中模拟了「文件夹」或「目录」的展示方式，具体实现是通过创建一个键值为 project/，内容为空的对象，展示方式上模拟了传统文件夹。

例如：上传对象 project/doc/a.txt，分隔符 / 会模拟「文件夹」的展示方式，于是可以看到控制台上出现「文件夹」project 和 doc，其中 doc 是 project 下一级「文件夹」，并包含 a.txt 文件。

因此，如果您的应用场景只是上传文件，可以直接上传即可，不需要先创建文件夹。

如果您的使用场景里面有文件夹的概念，需要提供创建文件夹的功能，您可以上传一个路径以 '/' 结尾的 0KB 文件。这样在您调用 GetBucket 接口时，就可以将这样的文件当做文件夹。

（2）QCloudCOSTransferMangerService

在 XML SDK 中，我们封装了可以智能判断是简单上传（复制）还是分块上传（复制）的操作，命名为 QCloudCOSTransferMangerService，同时对 API 设计和传输性能都做了优化，建议您直接使用。QCloudCOSTransferMangerService 的主要特性有：

- 支持断点上传。
- 支持根据文件大小智能选择简单上传（复制）还是分块上传（复制）。

使用 QCloudCOSTransferMangerService 上传的示例代码：

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];
put.object = @"exampleobject";
put.bucket = @"examplebucket-1250000000";
put.body = [@"testFileContent" dataUsingEncoding:NSUTF8StringEncoding];
//设置一些上传的参数
put.initMultipleUploadFinishBlock = ^(QCloudInitiateMultipartUploadResult * multipleUploadInitResult,
QCloudCOSXMLUploadObjectResumeData resumeData) {
//在初始化分块上传完成以后会回调该 block，在这里可以获取 resumeData，
//并且可以通过 resumeData 生成一个分块上传的请求
QCloudCOSXMLUploadObjectRequest* request = [QCloudCOSXMLUploadObjectRequest
requestWithRequestData:resumeData];
};
[put setSendProcessBlock:^(int64_t bytesSent, int64_t totalBytesSent,
int64_t totalBytesExpectedToSend) {
NSLog(@"upload %lld totalSend %lld aim %lld", bytesSent, totalBytesSent,
totalBytesExpectedToSend);
}];
[put setFinishBlock:^(QCloudUploadObjectResult *result, NSError* error) {
//可以从 result 获取结果
}];

[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
//舍弃一个分块上传并删除已上传的块
[put abort:^(id outputObject, NSError *error) {
//
}];

//•••在完成了初始化，并且上传没有完成前
NSError* error;
//这里是主动调用取消，并且产生 resumetData 的例子
QCloudCOSXMLUploadObjectResumeData resumeData = [put cancelByProductingResumeData:&error];
QCloudCOSXMLUploadObjectRequest* request = nil;
if (resumeData) {
request = [QCloudCOSXMLUploadObjectRequest requestWithRequestData:resumeData];
}
//生成的用于恢复上传的请求可以直接上传
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:request];
```

⚠ 注意：

按照分块上传的运行原理，只有当一个分块上传完了，那么后台服务器才会将该分块记录下来，并且叠加进度。并且以下几种情况无法进行断点续传，而是重新开始一次上传过程：

- 上传的文件小于1M，没有进行分块上传。
- 没有使用 QCloudCOSXMLUploadObjectRequest 类进行上传，而是直接使用简单上传接口。
- 取消生成 resumeData 时候初始化分块上传还没有完成（完成初始化上传的回调还没有调用）。

（3）新增 API

XML SDK 增加了很多新的 API，您可根据需求进行调用。包括：

- 存储桶的操作，如 QCloudPutBucketRequest、QCloudGetBucketRequest、QCloudListBucketRequest 等。
- 存储桶 ACL 的操作，如 QCloudPutBucketACLRequest、QCloudGetBucketACLRequest 等。
- 存储桶生命周期的操作，如 PQCloudutBucketLifecycleRequest、QCloudGetBucketLifecycleRequest 等。

具体请参考我们的 iOS SDK 快速入门。

Java SDK

快速入门

最近更新时间：2022-08-01 17:28:28

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：

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>cos_api</artifactId>
<version>5.6.97</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://cloud.tencent.com/document/product/436/6224
// clientConfig 中包含了设置 region, https(默认 http), 超时, 代理等 set 方法, 使用可参见源码或者常见问题 Java SDK 部分。
Region region = new Region("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, tmpSecretKey, sessionToken);
// 2 设置 bucket 的区域, COS 地域的简称请参阅 https://cloud.tencent.com/document/product/436/6224
// clientConfig 中包含了设置 region, https(默认 http), 超时, 代理等 set 方法, 使用可参阅源码或者常见问题 Java SDK 部分
Region region = new Region("COS_REGION");
ClientConfig clientConfig = new ClientConfig(region);
// 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");
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：2022-08-10 15:57:52

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。

```
<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://cloud.tencent.com/document/product/436/6224
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://cloud.tencent.com/document/product/436/6224
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：

版权所有：腾讯云计算（北京）有限责任公司

```
// 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 SelfDefinedEndpointBuilder();
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：2022-12-28 17:47:27

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：

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"));
```

获取主色调

```
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 获取图片主色调
String rule = "imageAve";
COSObject object = cosClient.getObject(getObj);
COSObjectInputStream objectContent = object.getObjectContent();
```

获取图片基本信息

```
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 图片基本信息
String rule = "imageInfo";
getObj.putCustomQueryParameter(rule, null);
COSObject object = cosClient.getObject(getObj);
COSObjectInputStream objectContent = object.getObjectContent();
```

版权所有：腾讯云计算（北京）有限责任公司

图片持久化处理

最近更新时间：2022-05-11 10:30:50

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

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
size	图片大小	Integer
quality	图片质量	Integer

版权所有：腾讯云计算（北京）有限责任公司

成员名称	描述	类型
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();
}
```

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：2022-05-11 10:30:55

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。

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：2022-05-11 10:30:58

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://cloud.tencent.com/document/product/436/46782
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

成员名称	描述	类型
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/aHR0cDovL2V4YW1wbGVidWNrZXQtMTI1MDAwMDAwMC5jb3MuYXAtZ3Vhbmd6aG91Lm15cWNsb3VkLmNvbS9zaHVpeWluLnBuZw==");
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
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
图片二维码

最近更新时间：2022-12-12 16:21:55

简介

本文档提供关于对象存储图片二维码相关的 API 概览以及 SDK 示例代码。

API	操作描述
二维码生成	指定的文本信息（URL 或文本），生成对应的二维码或条形码
二维码生成	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

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

```
String generateQrcode(GenerateQrcodeRequest request);
```

请求示例

```
//1.创建二维码生成请求对象
GenerateQrcodeRequest request = new GenerateQrcodeRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("examplebucket-1250000000");
request.setQrcodeContent("数据万象");
request.setWidth("400");
request.setMode("0");
String imageBase64 = cosClient.generateQrcode(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
bucketName	Bucket 的命名规则为 BucketName-APPID，详情请参见 存储桶概述	String	是
QrcodeContent	可识别的二维码文本信息	String	是
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	String	是
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	否

返回结果说明

- 成功：成功则返回 String，信息为 base64编码的图片。
- 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。

二维码识别

对象存储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。

版权所有：腾讯云计算（北京）有限责任公司

示例代码1：上传时识别二维码

```
// bucket 名需包含 appid
// api 请参见 https://cloud.tencent.com/document/product/436/54070
String bucketName = "examplebucket-1250000000";
String key = "qrcode.png";
File localFile = new File("E://qrcode.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-1.png");
rule.setRule("QRcode/cover/1");
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();
}
```

示例代码2：下载时识别二维码

```
GetObjectRequest getObj = new GetObjectRequest(bucketName, key);
// 这里是图片二维码识别参数，具体请参考数据万象 API，这里仅是示例
String cover = "0";
getObj.putCustomQueryParameter("ci-process", "QRcode");
getObj.putCustomQueryParameter("cover", cover);
```

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
size	图片大小	Integer
quality	图片质量	Integer
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	Integer
QRcodeInfoList	二维码识别结果，可能有多个	List

QRcodeInfo 用于记录二维码识别结果，其主要成员说明如下：

成员名称	描述	类型
codeUrl	二维码的内容可能识别不出	String
codeLocation	图中识别到的二维码位置坐标	CodeLocation

CodeLocation 用于记录图中识别到的二维码位置坐标 ，其主要成员说明如下：

成员名称	描述	类型

版权所有：腾讯云计算（北京）有限责任公司

成员名称	描述	类型
points	二维码坐标点的集合	List

图片标签

最近更新时间：2022-06-08 11:49:54

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。

汽车识别

最近更新时间：2022-06-08 11:50:02

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。

以图搜图

最近更新时间：2023-01-18 12:28:20

简介

本文档提供关于以图搜图的 API 概览和 SDK 示例代码。

API	操作描述
开通以图搜图	该接口用于开通 Bucket 搜图功能
添加图库图片	该接口用于添加图库图片
图片搜索接口	该接口用于检索图库中的图片
删除图库图片	该接口用于删除图库图片

开通以图搜图

功能说明

该接口用于开通 Bucket 搜图功能。

方法原型

```
boolean openImageSearch(OpenImageSearchRequest imageSearchRequest);
```

请求示例

```
//1.创建任务请求对象
OpenImageSearchRequest request = new OpenImageSearchRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("demobucket-123456789");
request.setMaxCapacity("100");
request.setMaxQps("10");
//3.调用接口,获取任务响应对象
boolean response = client.openImageSearch(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
maxCapacity	Request	图库容量限制	String	是
MaxQps	Request	图库访问限制，默认最高为10qps	String	否

返回结果说明

- 成功：成功则返回 true。
- 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。

添加图库图片

功能说明

将图片资源添加至图库。

方法原型

```
boolean addGalleryImages(ImageSearchRequest imageSearchRequest);
```

请求示例

```
//1.创建任务请求对象
ImageSearchRequest request = new ImageSearchRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("demobucket-123456789");
request.setObjectKey("1.png");
request.setEntityId("mark1");
//3.调用接口,获取任务响应对象
boolean response = client.addGalleryImages(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
entityId	Request	物品 ID，最多支持64个字符。若 EntityId 已存在，则对其追加图片	String	是
customContent	Request	用户自定义的内容，最多支持4096个字符，查询时原样带回	String	否
tags	Request	图片自定义标签，最多不超过10个，json 字符串，格式为 key:value 对	String	否

返回结果说明

- 成功：成功则返回 true。
- 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。

图片搜索接口

功能说明

检索图库中的图片。

方法原型

```
ImageSearchResponse searchGalleryImages(ImageSearchRequest imageSearchRequest);
```

请求示例

```
//1.创建任务请求对象
ImageSearchRequest request = new ImageSearchRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("demobucket-123456789");
request.setObjectKey("1.png");
//3.调用接口,获取任务响应对象
ImageSearchResponse response = client.searchGalleryImages(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
matchThreshold	Request	出参 Score 中，只有超过 MatchThreshold 值的结果才会返回。默认为0	String	否
Offset	Request	起始序号，默认值为0	String	否
limit	Request	返回数量，默认值为10，最大值为100	String	否
filter	Request	针对入库时提交的 Tags 信息进行条件过滤。支持>、>=、<、<=、=、!=，多个条件之间支持 AND 和 OR 进行连接	String	否

返回结果说明

- 成功：成功则返回 ImageSearchResponse 响应对象。
- 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。

删除图库图片

功能说明

将图片资源移出图库。

方法原型

```
boolean deleteGalleryImages(ImageSearchRequest imageSearchRequest);
```

请求示例

```
//1.创建任务请求对象
ImageSearchRequest request = new ImageSearchRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("demobucket-123456789");
request.setObjectKey("1.png");
request.setEntityId("mark");
//3.调用接口,获取任务响应对象
boolean response = client.deleteGalleryImages(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
entityId	Request	物品 ID	String	是

返回结果说明

- 成功：成功则返回 true。
- 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：2022-08-01 10:23:05

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 智能封面	String	是
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 • 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。
AnimateOnly KeepKeyFrame	Request.Video	动图只保留关键帧	String	否	无	• true、false • 动图保留关键帧参数
AnimateTime IntervalOfFrame	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
AnimateFrames PerSecond	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：AnimateFrames PerSecond > 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	截图模式	String	是	Interval	• 值范围：{Interval, Average} • 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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
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	基准位置	String	是	无	TopRight、TopLeft、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

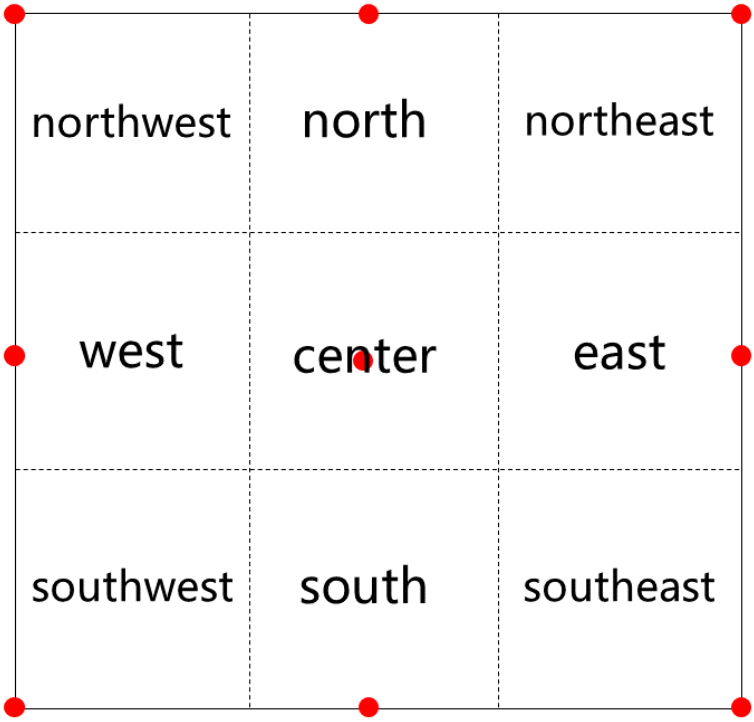
Container 类型 Image 的具体数据描述如下：

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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
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

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 • 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

版权所有：腾讯云计算（北京）有限责任公司

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	分辨率调整方式	String	否	0	• 取值0、1；0 表示使用原视频分辨率；1表示返回转码失败 • 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
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Maxrate	Request.Video	视频码率峰值	String	否	无	• 值范围：[10，50000] • 单位：Kbps
HlsTsTime	Request.Video	hls分片时间	String	否	5	• (0 视频时长] • 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

节点名称（关键字	描述	类型	必选
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);
```

任务操作

最近更新时间：2020-12-28 11:39:52

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
CreateMediaJobs	创建任务	创建一个媒体处理任务
CancelMediaJob	删除任务	删除一个媒体处理任务，正在处理中的任务不可删除
DescribeMediaJob	查询任务	查询一个任务
DescribeMediaJobs	查询任务列表	查询队列中的任务列表

基本操作

创建任务

功能说明

创建一个媒体处理任务。

方法原型

```
public MediaJobResponse createMediaJobs(MediaJobsRequest req);
```

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	Request	创建任务的 Tag：Animation 动图任务，Snapshot 截图任务，Transcode 转码任务，SmartCover 智能封面任务	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

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	必选
Object	Request.Input	媒体文件的名字	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	必选
Animation	Request.Operation	指定该任务的参数，同创建动图模板 CreateMediaTemplate 接口中的 Request.Animation	Container	否
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
Transcode	Request.Operation	指定该任务的参数，同创建转码模板 CreateMediaTemplate 接口中的 Request.Transcode	Container	否

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选
Watermark	Request.Operation	指定该任务的参数，同创建水印模板 CreateMediaTemplate 接口中的 Request.Watermark	Container	否
SmartCover	Request.Operation	当 Tag 为 SmartCover 时有效，目前为空	Container	否
TemplateId	Request.Operation	指定的模版 ID	String	否
Output	Request.Operation	结果输出地址	Container	是

> ⚠ 注意：
> 优先使用 TemplateId，无 TemplateId 时使用对应任务类型的参数。

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	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	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	String	是

返回结果说明

- 成功：返回 job 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
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");
request.getOperation().getOutput().setObject("2.mp4");
request.setQueueId("p9900025e4ec44b5e8225e70a52170834");
//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。

方法原型

```
public Boolean cancelMediaJob(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
jobId	要取消的任务 id	String	是

返回结果说明

- 成功：返回一个布尔类型，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

```
MediaJobsRequest request = new MediaJobsRequest();
request.setBucketName("examplebucket-1250000000");
request.setJobId("jae776cb4ec3011eab2cdd3817d4*****");
Boolean response = client.cancelMediaJob(request);
```

查询任务

功能说明

根据任务 id 查询任务详情。

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
jobId	要取消的任务 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 MediaJobObject 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
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

功能说明

查询队列中的任务列表。

方法原型

```
public MediaListJobResponse describeMediaJobs(MediaJobsRequest cIMediaJobsRequest);
```

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
queueId	拉取该队列 ID 下的任务	String	是
tag	任务的Tag：Animation	String	是
orderByTime	Desc 或者 Asc。默认为 Desc	String	否
nextToken	请求的上下文，用于翻页。上次返回的值	String	否
size	拉取的最大任务数。默认为10。最大为100	Integer	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
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	否
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	否

返回结果说明

- 成功： 返回一个任务集合响应实体。
- 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

```
MediaJobsRequest request = new MediaJobsRequest();
request.setBucketName("examplebucket-1250000000");
request.setQueueId("p9900025e4ec44b5e8225e70a521*****");
request.setTag("Transcode");
MediaListJobResponse response = client.describeMediaJobs(request);
List<MediaJobObject> jobsDetail = response.getJobsDetail();
```

版权所有：腾讯云计算（北京）有限责任公司

工作流操作

最近更新时间：2022-06-08 11:50:17

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(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
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");
```

版权所有：腾讯云计算（北京）有限责任公司

```
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(MediaWorkflowListRequest 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
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(request);
List<MediaWorkflowExecutionObject> workflowExecutionList = response.getWorkflowExecutionList();
```

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 request);
```

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。

版权所有：腾讯云计算（北京）有限责任公司

桶操作

最近更新时间：2020-10-23 11:39:41

简介

本文档提供关于数据万象媒体处理桶的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DescribeMediaBuckets	查询媒体 Bucket	查询当前账号下已经开通媒体处理功能的桶列表

基本操作

查询开通媒体处理功能的桶列表

功能说明

查询当前账号下已经开通媒体处理功能的桶列表。

方法原型

```
public MediaBucketResponse describeMediaBuckets(MediaBucketRequest mediaBucketRequest);
```

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
bucketNames	bucket 名字，以,分隔，支持多 bucket，精确搜索	String	否
regions	地区信息，以,分隔字符串，支持 All，ap-shanghai，ap-beijing	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 bucket 对象信息。
- 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);
```

队列操作

最近更新时间：2020-10-23 11:40:12

简介

本文档提供关于数据万象媒体处理队列的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DescribeMediaQueues	查询队列	查询当前账号下对应的队列信息
UpdateMediaQueue	更新队列	接口用于更新队列，修改回调信息

基本操作

查询队列

功能说明

查询当前账号下对应的队列信息。

方法原型

```
public MediaListQueueResponse describeMediaQueues(MediaQueueRequest mediaQueueRequest);
```

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
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转码服务调度转码执行 • Paused 表示队列暂停，作业不再会被媒体转码调度转码执行，队列内的所有作业状态维持在暂停状态，已经处于转码中的任务将继续转码，不受影响	string	否
pageNumber	第几页	string	否
pageSize	每页个数	string	否

返回结果说明

- 成功：返回队列对象集合信息。
- 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

```
MediaQueueRequest request = new MediaQueueRequest();
request.setBucketName("examplebucket-1250000000");
MediaListQueueResponse response = client.describeMediaQueues(request);
```

更新队列

功能说明

接口用于更新队列，修改回调信息。

方法原型

```
public MediaQueueResponse updateMediaQueue(MediaQueueRequest mediaQueueRequest);
```

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
Name	模板名称,长度限制100字符	string	是
state	• Active 表示队列内的作业会被媒体转码服务调度转码执行 • Paused 表示队列暂停，作业不再会被媒体转码调度转码执行， 队列内的所有作业状态维持在暂停状态，已经处于转码中的任务将继续转码，不受影响	string	是
QueueID	管道 ID	string	是
NotifyConfig	通知渠道,第三方回调 url	Container	是

Container 类型 NotifyConfig 的具体数据描述如下：

参数名称	描述	类型	必选
Url	回调 url 地址	String	否
Type	回调类型，普通回调：Url	String	否
Event	回调事件，视频转码完成：TransCodingFinish	String	否
State	回调开关，Off，On	String	否

返回结果说明

- 成功： 返回队列响应实体，包含有关 queue 的描述。
- 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

```
MediaQueueRequest request = new MediaQueueRequest();
request.setBucketName("examplebucket-1250000000");
request.setQueueId("p9900025e4ec44b5e8225e70a521*****");
request.getNotifyConfig().setUrl("cloud.tencent.com");
request.setState("Active");
request.setName("testQueue");
MediaQueueResponse response = client.updateMediaQueue(request);
```

视频截帧

最近更新时间：2022-05-06 14:14:25

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称）	父节点	描述	类型	是否必选
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

视频元信息获取

最近更新时间：2020-10-23 11:40:59

简介

本文档提供关于数据万象处理媒体信息的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
GenerateMediainfo	获取媒体文件的信息	查询 Bucket 中文件的媒体信息详情

基本操作

获取媒体文件的信息

功能说明

查询 Bucket 中文件的媒体信息详情。

方法原型

```
public MediaInfoResponse generateMediainfo(MediaInfoRequest 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
object	input.objet 查询的对象在 Bucket 中的位置信息	String	是

返回结果说明

- 成功：返回媒体对象信息。
- 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.创建媒体信息请求对象
MediaInfoRequest request = new MediaInfoRequest();
//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.mp3");
//3.调用接口,获取媒体信息响应对象
MediaInfoResponse response = client.generateMediainfo(request);
```

文档预览
任务操作

最近更新时间：2022-12-12 16:23:30

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，此处以动图任务举例。

API	操作名	操作描述
CreateDocProcessJobs	创建任务	创建一个文档预览任务
DescribeDocProcessJob	查询任务	查询一个文档预览任务
DescribeDocProcessJobs	查询任务列表	查询队列中的文档预览任务列表

基本操作

创建任务

功能说明

创建一个文档预览任务。

方法原型

```
public DocJobResponse createDocProcessJobs(DocJobRequest 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
Tag	Request	文档预览固定传 DocProcess	String	是
Input	Request	待操作的文件路径信息	Container	是
Operation	Request	操作规则	Container	是
QueueId	Request	任务所在的队列 ID，开通预览服务后自动生成，请使用 查询队列 获取或前往 万象控制台 在存储桶中查询。	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	文件路径	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
DocProcess	Request.Operation	当 Tag 为 DocProcess 时有效，指定该任务的参数	Container	是
Output	Request.Operation	结果输出地址	Container	是

Container 类型 DocProcess 的具体数据描述如下：

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
SrcType	Request.Operation.DocProcess	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	Request.Operation.DocProcess	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	String	否
SheetId	Request.Operation.DocProcess	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	Int	否
StartPage	Request.Operation.DocProcess	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	Request.Operation.DocProcess	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	否
ImageParams	Request.Operation.DocProcess	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
DocPassword	Request.Operation.DocProcess	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	Request.Operation.DocProcess	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
PaperDirection	Request.Operation.DocProcess	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
Quality	Request.Operation.DocProces	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	Request.Operation.DocProces	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

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

返回结果说明

- 成功：返回 DocJobResponse 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
DocJobRequest request = new DocJobRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("examplebucket-1250000000");
DocJobObject docJobObject = request.getDocJobObject();
docJobObject.setTag("DocProcess");
docJobObject.getInput().setObject("demo.docx");
docJobObject.setQueueId("pc02270c617ae4b6d9b0a52cb1c*****");
DocProcessObject docProcessObject = docJobObject.getOperation().getDocProcessObject();
docProcessObject.setQuality("100");
docProcessObject.setZoom("100");
docProcessObject.setStartPage("1");
docProcessObject.setEndPage("3");
docProcessObject.setTgtType("png");
docProcessObject.setDocPassword("123");
MediaOutputObject output = docJobObject.getOperation().getOutput();
output.setRegion("ap-chongqing");
output.setBucket("examplebucket-1250000000");
output.setObject("mark/pic-${Page}.jpg");
//3.调用接口,获取任务响应对象
DocJobResponse docProcessJobs = client.createDocProcessJobs(request);
```

查询任务

功能说明

根据任务 ID 查询任务详情。

方法原型

```
public DocJobResponse describeDocProcessJob(DocJobRequest 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	要取消的任务 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 DocJobResponse 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
DocJobRequest request = new DocJobRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("examplebucket-1250000000");
request.setJobId("d75b6ea083df711eb8d09476dfb8*****");
//3.调用接口,获取任务响应对象
DocJobResponse docJobResponse = client.describeDocProcessJob(request);
```

查询任务列表

功能说明

查询队列中的任务列表。

方法原型

```
public DocJobListResponse describeDocProcessJobs(DocJobListRequest request);;
```

参数说明

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
queueId	拉取该队列 ID 下的任务	String	是
tag	任务的Tag：Animation	String	是
orderByTime	Desc 或者 Asc。默认为 Desc	String	否
nextToken	请求的上下文，用于翻页。上次返回的值	String	否
size	拉取的最大任务数。默认为10。最大为100	Integer	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
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	否
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	否

返回结果说明

- 成功：返回 DocJobListResponse 响应实体。
- 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
DocJobListRequest request = new DocJobListRequest();
//2.添加请求参数 参数详情请见 api 接口文档
request.setBucketName("examplebucket-1250000000");
request.setQueueId("pc02270c617ae4b6d9b0a52cb1c*****");
request.setTag("DocProcess");
request.setStartCreationTime("2020-12-10T16:20:07+0800");
//3.调用接口,获取任务响应对象
DocJobListResponse docJobResponse = client.describeDocProcessJobs(request);
```

桶操作

最近更新时间：2020-12-17 14:47:47

简介

本文档提供关于数据万象文档预览桶的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DescribeDocProcessBuckets	查询文档预览 Bucket	查询当前账号下已经开通文档处理功能的桶列表

基本操作

查询开通文档处理功能的桶列表

功能说明

查询当前账号下已经开通文档预览功能的桶列表。

方法原型

```
public DocBucketResponse describeDocProcessBuckets(DocBucketRequest 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
bucketNames	bucket 名字，以,分隔，支持多 bucket，精确搜索	String	否
regions	地区信息，以,分隔字符串，支持 All，ap-shanghai，ap-beijing	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 bucket 对象信息。
- 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.创建模板请求对象
DocBucketRequest request = new DocBucketRequest();
//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,获取任务响应对象
DocBucketResponse docBucketResponse = client.describeDocProcessBuckets(request);
```

队列操作

最近更新时间：2020-12-17 14:47:53

简介

本文档提供关于数据万象文档预览队列的相关 API 概览以及 SDK 示例代码。

API	操作名	操作描述
DescribeDocProcessQueues	查询队列	查询当前账号下对应的队列信息
UpdateDocProcessQueue	更新队列	接口用于更新队列，修改回调信息

基本操作

查询队列

功能说明

查询当前账号下对应的队列信息。

方法原型

```
public DocListQueueResponse describeDocProcessQueues(DocQueueRequest 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
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文档转码服务调度转码执行 • Paused 表示队列暂停，作业不再会被文档转码调度转码执行，队列内的所有作业状态维持在暂停状态，已经处于转码中的任务将继续转码，不受影响	string	否
pageNumber	第几页	string	否
pageSize	每页个数	string	否

返回结果说明

- 成功：返回队列对象集合信息。
- 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
DocQueueRequest request = new DocQueueRequest();
//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,获取任务响应对象
DocListQueueResponse response = client.describeDocProcessQueues(request);
```

更新队列

功能说明

接口用于更新队列，修改回调信息。

方法原型

```
public boolean updateDocProcessQueue(DocQueueRequest 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
Name	模板名称,长度限制100字符	string	是
state	• Active 表示队列内的作业会被文档转码服务调度转码执行 • Paused 表示队列暂停，作业不再会被文档转码调度转码执行， 队列内的所有作业状态维持在暂停状态，已经处于转码中的任务将继续转码，不受影响	string	是
QueueID	管道 ID	string	是
NotifyConfig	通知渠道,第三方回调 url	Container	是

Container 类型 NotifyConfig 的具体数据描述如下：

参数名称	描述	类型	必选
Url	回调 url 地址	String	否
Type	回调类型，普通回调：Url	String	否
Event	回调事件，视频转码完成：TransCodingFinish	String	否
State	回调开关，Off，On	String	否

返回结果说明

- 成功： 返回布尔值 true 则修改成功 修改失败则抛出异常。
- 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
DocQueueRequest request = new DocQueueRequest();
//2.添加请求参数 参数详情请见api接口文档
request.setBucketName("examplebucket-1250000000");
request.setQueueId("pc02270c617ae4b6d9b0a52cb1cf****");
request.getNotifyConfig().setUrl("http://cloud.tencent.com");
request.getNotifyConfig().setState("On");
request.getNotifyConfig().setEvent("TransCodingFinish");
request.getNotifyConfig().setType("Url");
request.setState("Active");
request.setName("mark");
//3.调用接口,获取任务响应对象
boolean result = client.updateDocProcessQueue(request);
```

版权所有：腾讯云计算（北京）有限责任公司

生成同步预览链接

最近更新时间：2022-08-24 15:01:06

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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	描述	类型	是否必选
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：2022-10-14 15:43:22

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

节点名称（关键字）	描述	类型	是否必选
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ObjectKeydetectUrl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
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。

参数名称	类型	描述
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。

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
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

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
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，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	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

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
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：2021-12-21 14:30:45

简介

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：

版权所有：腾讯云计算（北京）有限责任公司

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，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

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

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：2022-11-03 11:01:21

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

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

- 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：2022-10-14 16:00:37

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.com/%E9%97%AE%E9%A2%98%E6%B1%87%E6%80%BB.pptx");
//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。
- 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：2022-10-14 16:01:23

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。

文件处理
文件压缩

最近更新时间：2023-02-01 10:34:37

简介

本文档提供关于数据万象文件处理压缩任务的相关 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	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。

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，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	是否必选

版权所有：腾讯云计算（北京）有限责任公司

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。	String	是
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

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.FileCompress);
FileCompressConfig fileCompressConfig=request.getOperation().getFileCompressConfig();
fileCompressConfig.setFormat("zip");
fileCompressConfig.setFlatten("0");
List<String> keyList=fileCompressConfig.getKey();
keyList.add("mark/pic-1.jpg");
keyList.add("mark/pic-1.pdf");
```

```
//QueueId 需要从控制台获取
request.setQueueId("p1ff062b35a494cf0ac4b572df22****");
MediaOutputObject output=request.getOperation().getOutput();
output.setBucket("demo-1234567890");
output.setRegion("ap-shanghai");
output.setObject("output/demo.zip");
//3.调用接口,获取任务响应对象
FileProcessJobResponse response=client.createFileProcessJob(request);
```

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

```
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);
```

文件哈希值计算

最近更新时间：2022-12-28 17:46:35

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，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

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

版权所有：腾讯云计算（北京）有限责任公司

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

- 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().getFileHashCodeConfig();
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);
```

文件解压

最近更新时间：2022-12-28 17:46:40

简介

本文档提供关于数据万象文件处理解压任务的相关 API 概览以及 SDK 示例代码。

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

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，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

版权所有：腾讯云计算（北京）有限责任公司

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： - 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

示例：
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

- 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.FileUncompress);
request.getInput().setObject("output/demo.zip");
FileUnCompressConfig fileUnCompressConfig = request.getOperation().getFileUnCompressConfig();
fileUnCompressConfig.setPrefix("output/");
fileUnCompressConfig.setPrefixReplaced("1");
request.setQueueId("p1ff062b35a494cf0ac4b572df22a5650");
MediaOutputObject output = request.getOperation().getOutput();
output.setBucket("demo-1234567890");
output.setRegion("ap-shanghai");
//3.调用接口,获取任务响应对象
FileProcessJobResponse response = client.createFileProcessJob(request);
```

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

```
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);
```

回调信息处理

最近更新时间：2022-08-10 15:57:42

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.getBucketName(), "/doc_jobs/" + request.getJobId(), request, H
ttpMethodName.GET);
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 Unmarshallers.DescribeDocJobUnmarshaller();
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 imageAuditingRequest) {
```

版权所有：腾讯云计算（北京）有限责任公司

```
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(imageAuditingRequest.getBucketName(), "/image/auditing/" + i
mageAuditingRequest.getJobId(), imageAuditingRequest, HttpMethodName.GET);
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 Exception {
JSONObject response = new JSONObject(jsonResponse);
JSONObject json = new JSONObject();
json.put("Response",response);
String xml = XML.toString(json);
Unmarshallers.ImageAuditingDescribeJobUnmarshaller imageJobUnmarshaller = new Unmarshallers.ImageAuditingDescribeJobUnmarshaller
();
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：2022-08-10 15:57:46

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：

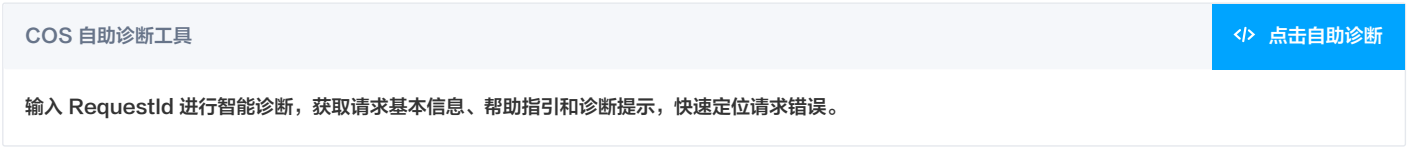
request 成员	描述	类型
requestId	请求 ID，用于表示一个请求，对于排查问题十分重要	String
traceId	辅助排查问题的 ID	String
statusCode	response 的 status 状态码，4xx 是指请求因客户端而失败，5xx 是服务端异常导致的失败。请参见 COS 错误码	String
errorType	枚举类，表示异常的种类，分为 Client，Service，Unknown	ErrorType
errorCode	请求失败时 body 返回的 Error Code 请参见 COS 错误码	String
errorMessage	请求失败时 body 返回的 Error Message 请参见 COS 错误码	String

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. 稍候片刻，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。

JavaScript SDK
快速入门

最近更新时间：2023-01-06 10:48:37

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。如果您仍在使用 JSON 版本 SDK，请 升级到 XML JavaScript SDK。

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：

单击此处 下载最新 cos-js-sdk-v5.min.js。

```
<script src="https://cdn.jsdelivr.net/npm/cos-js-sdk-v5/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');
```

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。 https://console.cloud.tencent.com/cos5/bucket
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.cloud.tencent.com/cos5/bucket/
// 关于地域的详情见 https://cloud.tencent.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/master/server/
// 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-sts-sdk
// STS 详细文档指引看：https://cloud.tencent.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, null, 2))
};
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/master/server/
// 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-sts-sdk
// STS 详细文档指引看：https://cloud.tencent.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, null, 2))
};
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/scope.md
// 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(data, null, 2))
};
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
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重复使用
```

```
});
};
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 invalid');
callback({
Authorization: data.authorization,
// SecurityToken: data.sessionToken, // 如果使用临时密钥，需要把 sessionToken 传给 SecurityToken
});
};
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.cloud.tencent.com/cam/capi 进行查看和管理
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 pathname 计算，默认 false	Boolean	否
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），必须字段 */
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使用简单上传。可自行设置，非必须 */
});
}
},
}
</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。

版权所有：腾讯云计算（北京）有限责任公司

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.jpg）支持中文，必须字段 */
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使用简单上传。可自行设置，非必须 */
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 ? '&' : '?') + 'response-content-disposition=attachment';
/* 可拼接 filename 来实现下载时重命名 */
/* downloadUrl += ';filename=myname'; */
```

版权所有：腾讯云计算（北京）有限责任公司

```
// （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，可以使用隐藏的 iframe 下载，或使用 a 标签 download 属性协助下载
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);
});
```

版权所有：腾讯云计算（北京）有限责任公司

文档处理
查询文档预览开通状态

最近更新时间：2022-12-27 15:15:37

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

用于查询存储桶是否已开通文档预览功能。

示例代码

```
function describeDocProcessBuckets() {
var config = {
// 需要替换成您自己的存储桶信息
Region: 'COS_REGION', /* 存储桶所在地域，必须字段 */
};
var host = 'ci.' + config.Region + '.myqcloud.com/docbucket';
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: 'docbucket',
Url: url,
Query: {
// regions: '', /* 非必须，地域信息，以“,”分隔字符串，支持 All、ap-shanghai、ap-beijing */
// bucketNames: '', /* 非必须，存储桶名称，以“,”分隔，支持多个存储桶，精确搜索 */
// bucketName: '', /* 非必须，存储桶名称前缀，前缀搜索 */
// pageNumber: '1', /* 非必须，第几页 */
// pageSize: '20', /* 非必须，每页个数 */
},
},
function(err, data){
console.log(err || data);
});
}
describeDocProcessBuckets();
```

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

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 查询文档预览开通状态。

参数名称	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	-	Object
- - RequestId	请求的唯一 ID	string
- - TotalCount	文档预览 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
- - DocBucketList	文档预览 Bucket 列表	Array
- - - 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

文档转码
同步请求接口

最近更新时间：2022-12-27 15:15:42

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

文档转码同步请求，获取文档转码的请求 URL。

示例代码

```
function getDocPreviewUrl() {
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.getObjectUrl({
Bucket: config.Bucket, // Bucket 格式：test-1250000000
Region: config.Region,
Key: '文档.docx',
Query: {
'ci-process': 'doc-preview', /* 必须，数据万象处理能力，文档预览固定为 doc-preview */
// srcType: '', /* 非必须，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 */
// page: '', /* 非必须，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 */
// dstType: '', /* 非必须，转换输出目标文件类型 */
},
}, function(err, data) {
if (err) {
console.log(err);
} else {
// 使用浏览器打开url即可预览
var url = data.Url;
console.log(url);
}
});
}
getDocPreviewUrl();
```

参数说明

参数名称	描述	类型
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
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

返回结果说明

```
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，详情请参见 错误码 文档	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
- RequestId	请求的唯一 ID	String
- Url	文档预览的 Url	String

异步处理任务接口

最近更新时间：2022-12-27 15:15:47

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	说明
提交文档预览任务	用于提交一个文档预览任务
查询指定的文档预览任务	用于查询指定的文档预览任务
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

用于提交一个文档预览任务。

示例代码

```
function createDocProcessJobs() {
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/doc_jobs';
var url = 'https://' + host;
var body = COS.util.json2xml({
Request: {
Tag: 'DocProcess',
Input: {
Object: '文档.docx', // 存在 cos 里的路径
},
Operation: {
DocProcess: {
TgtType: 'jpg',
},
Output: {
Bucket: config.Bucket,
Region: config.Region,
Object: '文档转码_${Number}.jpg', // 转码后存到 cos 的路径
},
},
QueueId: 'pa2e2c3d3fae042de909cafc16f1d801b', // 替换为自己的QueueId
}
});
cos.request({
Method: 'POST',
Key: 'doc_jobs',
Url: url,
Body: body,
ContentType: 'application/xml',
},
function(err, data){
```

```
console.log(err || data);
});
}
createDocProcessJobs();
```

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	否
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

返回结果说明

详情请参见 提交文档转码任务。

参数名称	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	-	Object
- - - 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

用于查询指定的文档预览任务。

示例代码

```
function describeDocProcessJob() {
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 = 'd87fbabd07b8611ed974b3f4b4064872e'; // 替换为自己的jobId
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/doc_jobs/' + jobId;
var url = 'https://' + host;
cos.request({
Method: 'GET',
```

```
Key: 'doc_jobs/' + jobId,
Url: url,
},
function(err, data){
console.log(err || data);
});
}
describeDocProcessJob();
```

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

详情请参见 查询指定的文档转码任务。

参数名称	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	-	Object
- - 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

用于拉取符合条件的文档预览任务。

示例代码

```
function describeDocProcessJobs() {
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/doc_jobs';
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: 'doc_jobs',
Url: url,
Query: {
queueId: 'pa2e2c3d3fae042de909cafc16f1d801b', // 替换为自己的QueueId
tag: 'DocProcess',
},
```

```
},
function(err, data){
console.log(err || data);
});
}
describeDocProcessJobs();
```

参数说明

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

详情请参见 拉取符合条件的文档转码任务。

参数名称	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	-	Object
- - 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：2022-12-27 15:15:52

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	说明
查询文档预览队列	查询文档预览队列信息
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

用于查询文档预览队列。

示例代码

```
function describeDocProcessQueues() {
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/docqueue';
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: 'docqueue',
Url: url,
Query: {
// queueIds: '', /* 非必须，队列 ID，以“,”符号分割字符串 */
// state: '', /* 非必须，可选Active或Paused */
// pageNumber: 1, /* 非必须，第几页 */
// pageSize: 2, /* 非必须，每页个数 */
},
},
function(err, data){
console.log(err || data);
});
}
describeDocProcessQueues();
```

参数说明

参数名称	描述	类型
queueIds	队列 ID，以“,”符号分割字符串	String
state	• Active 表示队列内的作业会被文档预览服务调度执行 • 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

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 查询文档转码队列。

参数名称	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	-	Object
- - 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

用于更新文档预览队列。

请求示例

```
function updateDocProcessQueue() {
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 */
};
```

```
// queueId 可通过查询文档预览队列获得
var queueId = 'pa2e2c3d3fae042de909cafc16f1d8xxx';
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com/docqueue/' + queueId;
var url = 'https://' + host;
var body = COS.util.json2xml({
Request: {
Name: 'queue-doc-process-1', // Name可通过查询文档预览队列获得
QueueID: queueId,
State: 'Active',
NotifyConfig: {
State: 'Off',
}
}
});
cos.request({
Method: 'PUT',
Key: 'docqueue/' + queueId,
Url: url,
Body: body,
ContentType: 'application/xml',
},
function(err, data){
console.log(err || data);
});
}
updateDocProcessQueue();
```

参数说明

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
- Url	回调配置	String
- Type	回调类型，普通回调：Url	String
- Event	回调事件，文档预览任务完成	String
- State	回调开关，Off，On	Container

返回结果说明

详情请参见 更新文档转码队列。

参数名称	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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
- headers	请求返回的头部信息	Object
- Response	-	Object
- - RequestId	请求的唯一 ID	String
- - Queue	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：2022-12-27 15:15:57

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。

请求示例

```
function getDocHtmlUrl() {
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.getObjectUrl({
Bucket: config.Bucket,
Region: config.Region,
Key: '文档.docx',
Query: {
'ci-process': 'doc-preview', /* 必须，数据万象处理能力，文档预览固定为 doc-preview */
dstType: 'html',
},
}, function(err, data) {
if (err) {
console.log(err);
} else {
// 使用浏览器打开 url 即可预览
var url = data.Url;
console.log(url);
}
});
}
getDocPreviewUrl();
```

参数说明

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 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

版权所有：腾讯云计算（北京）有限责任公司

名称	参数说明	类型	是否必选
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

返回结果说明

```
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，详情请参见 错误码 文档	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
- RequestId	请求的唯一 ID	String
- Url	文档预览的 Url	String

内容识别
图片标签

最近更新时间：2022-12-27 15:16:01

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

```
function getImageLabel() {
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 key = '素材.jpeg';
var host = config.Bucket + '.cos.' + config.Region + '.myqcloud.com/' + key;
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: key,
Url: url,
Query: {
'ci-process': 'detect-label'
},
},
function(err, data){
console.log(err || data);
});
}
getImageLabel();
```

参数说明

参数名称	类型	描述	是否必填
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

返回结果说明

参数名称	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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
- 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
- RecognitionResult	-	Object
- - Labels	Array	标签信息

图片二维码

最近更新时间：2022-12-27 15:16:06

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

```
<!-- 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 identifyQrcode_put(file) {
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 */
};
// 通过input组件选择文件得到 file 代码省略
cos.putObject({
Bucket: config.Bucket,
Region: config.Region,
Key: '上传二维码.png',
Body: file, // file 是 input[type="file"]标签选择本地文件后得到的 file 对象
Headers: {
'Pic-Operations':
'{"is_pic_info": 1, "rules": [{"fileid":"test.jpg","rule":" QRcode/cover/0"}]}',
},
onProgress: function (progressData) {
console.log('onProgress', JSON.stringify(progressData));
},
}, function (err, data) {
console.log('CIExample1:', 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;
}
identifyQrcode_put(file);
};
```

参数说明

参数名称	类型	描述	是否必填
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

返回结果说明

详情请参见 上传时识别。

请求示例2：下载时识别

```
function identifyQrcode_get() {
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 key = '二维码图片.png';
var host = config.Bucket + '.cos.' + config.Region + '.myqcloud.com/' + key;
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: key,
Url: url,
Query: {
'ci-process': 'QRcode'
},
},
function(err, data){
console.log(err || data);
});
}
identifyQrcode_get();
```

参数说明

参数名称	类型	描述	是否必填
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

返回结果说明

详情请参见 下载时识别。

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

```
function generateQrcode() {
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 + '.cos.' + config.Region + '.myqcloud.com';
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: '',
Url: url,
Query: {
'ci-process': 'qrcode-generate', /* 必须，对象存储处理能力，二维码生成参数为 qrcode-generate */
'qrcode-content': '二维码文案', /* 必须，可识别的二维码文本信息 */
// mode: 0, /* 非必须，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 */
width: 200, /* 必须，指定生成的二维码或条形码的宽度，高度会进行等比压缩 */
},
},
function(err, data){
if (!err) {
// 获得二维码 base64
var imgBase64 = data.Response.ResultImage;
// 比如可拼接前缀直接展示在 img 里
// document.querySelector('#img').src = '' + imgBase64;
}
});
}
generateQrcode();
```

参数说明

参数名称	类型	描述	是否必填
qrcode-content	String	可识别的二维码文本信息	是
mode	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
width	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

返回结果说明

参数名称	类型	描述	父节点
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	-

版权所有：腾讯云计算（北京）有限责任公司

参数名称	类型	描述	父节点
- RequestId	请求的唯一 ID	String	-
- Response	-	Object	-
- - ResultImage	二维码图片 base64 数据	String	-

图片文字识别（OCR）

最近更新时间：2022-12-27 15:16:11

简介

本文档提供关于通用文字识别的 API 概览和 SDK 示例代码。

API	说明
通用文字识别	通用文字识别功能将图片上的文字内容，智能识别为可编辑的文本。

通用文字识别

功能说明

通用文字识别功能（Optical Character Recognition，OCR）基于行业前沿的深度学习技术，将图片上的文字内容，智能识别为可编辑的文本，可应用于随手拍扫描、纸质文档电子化、电商广告审核等多种场景，大幅提升信息处理效率。

请求示例

```
function orc() {
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 key = '素材.jpeg';
var host = config.Bucket + '.cos.' + config.Region + '.myqcloud.com/' + key;
var url = 'https://' + host;
cos.request({
Method: 'GET',
Key: key,
Url: url,
Query: {
'ci-process': 'OCR', /* 必须，数据万象处理能力，图片文字识别固定为 OCR。 */
// type: '', /* 非必须，OCR 的识别类型 */
// 'language-type': '', /* 非必须，type 值为 general 时有效，表示识别语言类型 */
// ispdf: false, /* 非必须，type 值为 general、fast 时有效，表示是否开启 PDF 识别 */
// 'pdf-pagenumber': '', /* 非必须，type 值为 general、fast 时有效，表示需要识别的 PDF 页面的对应页码 */
// isword: false, /* 非必须，type 值为 general、accurate 时有效，表示识别后是否需要返回单字信息 */
// 'enable-word-polygon': false, /* 非必须，type 值为 handwriting 时有效，表示是否开启单字的四点定位坐标输出 */
},
},
function(err, data){
console.log(err || data);
});
}
orc();
```

参数说明

参数名称	描述	类型	是否必选
Key	对象文件名，例如：folder/document.jpg。	String	是
ci-process	数据万象处理能力，图片文字识别固定为 OCR。	String	是

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
type	OCR 的识别类型，有效值为 general，accurate，efficient，fast，handwriting。general 表示通用印刷体识别；accurate 表示印刷体高精度版；efficient 表示印刷体精简版；fast 表示印刷体高速版；handwriting 表示手写体识别。默认值为 general。	String	否
language-type	type 值为 general 时有效，表示识别语言类型。支持自动识别语言类型，同时支持自选语言种类，默认中英文混合(zh)，各种语言均支持与英文混合的文字识别。可选值请参见 可识别的语言类型。	String	否
ispdf	type 值为 general、fast 时有效，表示是否开启 PDF 识别，有效值为 true 和 false，默认值为 false，开启后可同时支持图片和 PDF 的识别。	Boolean	否
pdf-pagenumber	type 值为 general、fast 时有效，表示需要识别的 PDF 页面的对应页码，仅支持 PDF 单页识别，当上传文件为 PDF 且 ispdf 参数值为 true 时有效，默认值为1。	Integer	否
isword	type 值为 general、accurate 时有效，表示识别后是否需要返回单字信息，有效值为 true 和 false，默认为 false。	Boolean	否
enable-word-polygon	type 值为 handwriting 时有效，表示是否开启单字的四点定位坐标输出，有效值为 true 和 false，默认值为 false。	Boolean	否

返回结果说明

详情请参见 通用文字识别。

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：2022-12-27 15:18:16

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

水印的文字内容需要经过 URL 安全的 Base64 编码，可引入 js-base64 来解决。

示例代码如下：

```
1. 项目内安装依赖
npm install --save js-base64

2. 代码里引入 encode 方法
import { encode } from 'js-base64';

// 文字水印内容
var text = '腾讯云万象优图';
// 有效的文字水印
var safeBase64 = encode(text).replaceAll('+', '-').replaceAll('/', '_').replaceAll('=', '');
// 可使用如下Pic-Operations
'Pic-Operations': `{"is_pic_info": 1, "rules": [{"fileid": "waterMask.jpg", "rule": "watermark/2/text/${safeBase64}"}]}`
```

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

```
<!-- 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": "imageMogr2/thumbnail/200x/"}]}',
},
},
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": "imageMogr2/thumbnail/200x/"}]}',
},
},
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) {
// 使用浏览器打开url预览或触发浏览器下载
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) {
// 使用浏览器打开url预览或触发浏览器下载
console.log(data.URL);
}
},
);
```

盲水印

最近更新时间：2022-04-22 12:15:18

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": "watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn"}]}',
},
},
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/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn`,
},
function (err, data) {
console.log(err || data);
},
);
}
```

媒体处理
媒体 bucket 接口

最近更新时间：2022-04-22 12:15:24

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'ap-shanghai,ap-beijing'，非必须 */
// bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-1250000000'，支持多个值用逗号分隔如'test1-1250000000,test2-1250000000'，非必须 */
// 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隔如'test1,test2'，非必须 */
}
}, 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

参数名称	参数描述	类型	是否必填
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

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：2022-04-22 12:15:27

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
- - - - - 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

版权所有：腾讯云计算（北京）有限责任公司

参数名	参数描述	类型
- - - - 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
- - - - - 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：2022-04-22 12:15:31

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

参数名称	参数描述	类型	是否必填
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：2022-12-27 15:18:28

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。Content，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。Content，Object 和 Url 只能选择其中一种。	String	否
Content	Request.Input	图片文件的内容，需要先经过 base64 编码。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	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：2022-10-14 16:02:38

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-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	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

版权所有：腾讯云计算（北京）有限责任公司

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

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：2022-10-14 16:02:45

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

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：2022-10-14 16:02:52

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（Input传入Object）
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。

文档审核

最近更新时间：2022-05-31 10:50:21

简介

本文档提供关于文档审核的 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。

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 postDocumentAuditing() {
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/document/auditing';
var body = COS.util.json2xml({
Request: {
Input: {
Object: 'test.xlsx', /* 需要审核的文档文件，存储桶里的路径 */
},
Conf: {
BizType: '',
DetectType: 'Porn',
}
}
});
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'POST',
Url: url,
Key: '/document/auditing', /** 固定值，必须 */
ContentType: 'application/xml', /** 固定值，必须 */
Body: body
},
function(err, data){
console.log(err || data);
});
}
```

版权所有：腾讯云计算（北京）有限责任公司

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
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
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

返回结果说明

详情请参见 提交文档审核任务。

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。

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 getDocumentAuditingResult() {
var jobId = 'sd7815c21caff611eca12f525400d88xxx'; // jobId可以通过提交文档审核任务返回
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/document/auditing/' + jobId;
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'GET',
Key: '/document/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 查询文档审核任务结果。

网页审核

最近更新时间：2022-10-14 16:03:09

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，可以调用查询文本审核结果api查询
},
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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示的 html 内容。取值为 true 或者 false，默认为 false。	Boolean	否

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。

Node.js SDK
快速入门

最近更新时间：2023-01-06 10:49:10

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。

使用永久密钥初始化

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.cloud.tencent.com/cam/capi 进行查看和管理
var COS = require('cos-nodejs-sdk-v5');
var cos = new COS({
SecretId: 'SECRETID',
SecretKey: 'SECRETKEY'
});
```

使用临时密钥初始化

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 tmpSecretId
TmpSecretKey: credentials.tmpSecretKey, // 临时密钥的 tmpSecretKey
SecurityToken: credentials.sessionToken, // 临时密钥的 sessionToken
ExpiredTime: data.expiredTime, // 临时密钥失效时间戳，是申请临时密钥时，时间戳加 durationSeconds
});
});
}
});
```

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，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，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，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

参数名	参数描述	类型	是否必填
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/已取消状态的任务会被清理，默认1000	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 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) { ... }
```

版权所有：腾讯云计算（北京）有限责任公司

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
- 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);
});
```

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：2022-04-22 12:15:45

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

```
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.jpg", "rule": "imageMogr2/thumbnail/200x/"}]}'
},
}, 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。

版权所有：腾讯云计算（北京）有限责任公司

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.jpg", "rule": "imageMogr2/thumbnail/200x/"}]}'
},
}, 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：2022-04-22 12:15:48

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.jpg", "rule": "watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEy
NTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn"}]}'
},
}, 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/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn',
}, function (err, data) {
console.log(err || data);
fs.writeFileSync('filepath', data.Body); // 将图片内容保存本地
});
```

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：2022-06-29 14:40:17

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'ap-shanghai,ap-beijing'，非必须 */
// bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-1250000000'，支持多个值用逗号分隔如'test1-1250000000,test2-1250000000'，非必须 */
// 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隔如'test1,test2'，非必须 */
}
}, 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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型	是否必填
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

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：2022-06-29 14:40:22

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
- - - - - 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

版权所有：腾讯云计算（北京）有限责任公司

参数名	参数描述	类型
- - - - 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
- - - - - 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：2022-06-29 14:40:27

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

参数名称	参数描述	类型	是否必填
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：2022-12-27 15:18:43

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 Node.js SDK 版本需要大于等于 v2.11.2。

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。Content，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。Content，Object 和 Url 只能选择其中一种。	String	否
Content	Request.Input	图片文件的内容，需要先经过 base64 编码。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	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：2022-10-14 16:03:31

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 Node.js SDK 版本需要大于等于 v2.11.2。

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-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	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

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

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：2022-10-14 16:03:40

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 Node.js SDK 版本需要大于等于 v2.11.2。

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

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：2022-10-14 16:03:49

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 Node.js SDK 版本需要大于等于 v2.11.2。

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（Input传入Object）
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。

文档审核

最近更新时间：2022-10-14 16:03:57

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。

> 注意：
> COS Node.js SDK 版本需要大于等于 v2.11.2。

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。

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 postDocumentAuditing() {
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/document/auditing';
var body = COS.util.json2xml({
Request: {
Input: {
Object: 'test.xlsx', /* 需要审核的文档文件，存储桶里的路径 */
},
Conf: {
BizType: '',
DetectType: 'Porn',
}
}
});
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'POST',
Url: url,
```

```
Key: '/document/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
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
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
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

返回结果说明

详情请参见 提交文档审核任务。

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。

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 getDocumentAuditingResult() {
var jobId = 'sd7815c21caff611eca12f525400d88xxx'; // jobId可以通过提交文档审核任务返回
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
```

```
var url = 'https://' + host + '/document/auditing/' + jobId;
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'GET',
Key: '/document/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 查询文档审核任务结果。

网页审核

最近更新时间：2022-06-29 14:41:01

简介

本文档提供关于网页审核的 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，可以调用查询文本审核结果api查询
},
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);
});
}
```

版权所有：腾讯云计算（北京）有限责任公司

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示的 html 内容。取值为 true 或者 false，默认为 false。	Boolean	否

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。

PHP SDK
快速入门

最近更新时间：2023-01-04 20:39:23

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。如果您仍在使用 JSON 版本 SDK，请 升级到 XML PHP SDK。

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。

版权所有：腾讯云计算（北京）有限责任公司

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。

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

若您使用 临时密钥 初始化，请用下面方式创建实例：

```
$tmpSecretId = "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://console.cloud.tencent.com/cam/capi
$tmpSecretKey = "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，https://console.cloud.tencent.com/cam/capi
$tmpToken = "COS_TOKEN"; //使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
$region = "COS_REGION"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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)));
```

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：2022-12-14 17:52:21

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

```
$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<Height>>
$imageRule->thumbnailEqualRatioEnlargeByWH(300, 300); // /thumbnail/<Width>x<Height><
$imageRule->pad(1); // /pad/1
$imageRule->color('#3D3D3D'); // /color/IzNEM0QzRA
$imageRule->ignoreError(); // /ignore-error/1

$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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<dy>
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 /interlace/<Mode>
$imageRule->ignoreError(); // /ignore-error/1

$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$picOperations->setIsPicInfo(1); // is_pic_info
$picOperations->addRule($imageRule, "output.png"); // rules
```

版权所有：腾讯云计算（北京）有限责任公司

```
// ------------------- 1. 下载时处理 ------------------- //
// $downloadUrl = $cosClient->getObjectUrl('examplebucket-1250000000', 'xxx.jpg'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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；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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
```

```
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
```

```
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\ImageWatermarkTemplate();
$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（如水印墙）
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-1250000000.cos.ap-shanghai.myqcloud.com/shuiyin_2.png，则该处编码后的字符串为 aHR0cDovL2V4YW1wbGVidWNrZXQtMTI1MDAwMDAwMC5jb3MuYXAtc2hhbmdoYWkubXlxY2xvdWQuY29tL3NodWI5aW5fMi5wbmc
/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
/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

版权所有：腾讯云计算（北京）有限责任公司

参数	含义
/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, 1000]，单位为千分比当 scatype 设置为3时，该有效值为[1, 250]，单位为千分比。示例：http://examples-1251000004.cos.ap-shanghai.myqcloud.com/sample.jpeg?watermark/1/image/xxxxxxx/scatype/3/spcent/250
/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）。**
- **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
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 格式（例如 #FF0000）
$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（参见九宫格方位图），默认值 SouthEast
$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，表示开启平铺水印功能
```

```
$textWatermarkRule->setDegree(10); // 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
$textWatermarkRule->setShadow(10); // 文字阴影效果，有效值为[0,100]，默认为0，表示无阴影

$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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

参数	含义
/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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->ImageInfo(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->ImageExif(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->ImageAve(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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/<quality>
$imageViewRule->ignoreError(); // /ignore-error/1

$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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； 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

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
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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-limit/<value>!
$imageRule->ignoreError(); // /ignore-error/1

$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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：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


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

版权所有：腾讯云计算（北京）有限责任公司

```
$picOperations = new Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();
$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'); // 获取下载链接
$downloadUrl = 'https://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/xxx.jpg'; // 私有图片处理方式同上，仅增加签名部分，并与图片处理参数以“&”连接
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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：2022-04-22 12:16:19

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->PutBucketImageStyle(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'StyleName' => 'style_name',//样式名称
'StyleBody' => 'imageMogr2/thumbnail/!50px', //样式信息
));
// 请求成功
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	类型	描述	是否必填
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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
```

```
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->DeleteBucketImageStyle(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'StyleName' => 'style_name', //样式名称
));
// 请求成功
print_r($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：2022-04-22 12:16:23

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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();//创建基础图片处理参数模板实例
$imageMogrTemplate->thumbnailByScale(50);//指定图片的宽高为原图的 50%
$picOperationsTemplate = new \Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();//创建图片持久化处理参数模版实例
$picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
$picOperationsTemplate->addRule($imageMogrTemplate, "resultobject");//设置图片处理规则
$result = $cosClient->putObject(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
'Body' => fopen('path/to/localFile', 'rb'),
'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参数
));
// 请求成功
print_r($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

参数名称	类型	描述	是否必填
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/exampleobject
[Data] => Array
(
[OriginalInfo] => Array
(
[Key] => exampleobject
[Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exampleobject
[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.myqcloud.com/resultobject
[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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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();//创建基础图片处理参数模版实例
$imageMogrTemplate->thumbnailByScale(50);//指定图片的宽高为原图的 50%
$picOperationsTemplate = new \Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();//创建图片持久化处理参数模版实例
$picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
$picOperationsTemplate->addRule($imageMogrTemplate, "resultobject");//设置图片处理规则
$result = $cosClient->ImageProcess(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参数
));
// 请求成功
print_r($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
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/exampleobject
[OriginalInfo] => Array
(
[Key] => exampleobject
[Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exampleobject
[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.myqcloud.com/resultobject
[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
ProcessResults	Array	图片处理结果信息	无

盲水印

最近更新时间：2022-04-22 12:16:27

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\BlindWatermarkTemplate();//创建盲水印参数模版实例
$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 的取值范围为{1,2,3}，默认值为1，level 值越大则图片受影响程度越大、盲水印效果越好。
$picOperationsTemplate = new \Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();//创建图片持久化处理参数模版实例
$picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
$picOperationsTemplate->addRule($blindWatermarkTemplate, "resultobject");//设置图片持久化处理参数
$result = $cosClient->putObject(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
'Body' => fopen('path/to/localFile', 'rb'),
'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参数
));
```

```
// 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
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/exampleobject
[Data] => Array
(
[OriginalInfo] => Array
(
[Key] => exampleobject
[Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exampleobject
[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.myqcloud.com/resultobject
[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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\BlindWatermarkTemplate();//创建盲水印参数模版实例
$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 的取值范围为{1,2,3}，默认值为1，
$result = $cosClient->getObject(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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/exampleobject
)

)
```

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	类型	描述	父节点
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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\BlindWatermarkTemplate();//创建盲水印参数模版实例
$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 的取值范围为{1,2,3}，默认值为1，level 值越大则图片受影响程度越大、盲水印效果越好。
$picOperationsTemplate = new \Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();//创建图片持久化处理参数模版实例
$picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
$picOperationsTemplate->addRule($blindWatermarkTemplate, "resultobject");//设置图片持久化处理参数
$result = $cosClient->putObject(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'Key' => 'exampleobject',
'Body' => fopen('path/to/localFile', 'rb'),
'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参数
));
// 请求成功
print_r($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
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/exampleobject
[Data] => Array
(
[OriginalInfo] => Array
(
[Key] => exampleobject
[Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exampleobject
[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.myqcloud.com/resultobject
[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：2022-04-22 12:16:31

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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();//创建基础图片处理参数模版实例
$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，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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

参数名称	类型	描述	是否必填
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/exampleobject
)

)
```

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：2022-04-22 12:16:35

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->PutBucketGuetzli(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
));
// 请求成功
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

```
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
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->GetBucketGuetzli(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->DeleteBucketGuetzli(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
));
// 请求成功
print_r($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

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：2022-04-29 10:36:42

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->describeMediaBuckets(array(
'Regions' => '', // 可选 地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串
'BucketNames' => '', // 可选 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
'BucketName' => '', // 可选 存储桶名称前缀，前缀搜索
'PageNumber' => 1, // 可选 第几页
'PageSize' => 20, // 可选 每页个数
));
// 请求成功
print_r($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：2022-04-22 12:16:40

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->GetMediaInfo(
array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject', //媒体文件对象路径,如folder/movie.mp4
'ci-process' => 'videoinfo' //操作类型，固定使用 videoinfo
)
);
// 请求成功
echo($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 part 10
[CodecTimeBase] => 1/12800
[CodecTag] => 0x31637661
[Profile] => Main
[Height] => 720
[Width] => 1280
```

版权所有：腾讯云计算（北京）有限责任公司

```
[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/exampleobject
)
```

媒体截图接口

最近更新时间：2022-04-22 12:16:45

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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/exampleobject
)
```

私有 M3U8 接口

最近更新时间：2022-04-29 10:37:12

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->GetPrivateM3U8(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'xxx.m3u8', // 桶文件
'ci-process' => 'pm3u8', // 操作类型，固定使用 pm3u8
'expires' => '3600', // 私有 ts 资源 url 下载凭证的相对有效期，单位为秒，范围为[3600, 43200]
));
// 请求成功
print_r($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
(
)

)

[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
)
```

文档处理
查询文档预览开通状态

最近更新时间：2022-09-07 15:08:25

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->describeDocProcessBuckets(array(
'Regions' => '', // 可选 地域信息，例如 ap-shanghai、ap-beijing，若查询多个地域以“,”分隔字符串
'BucketNames' => '', // 可选 存储桶名称，以“,”分隔，支持多个存储桶，精确搜索
'BucketName' => '', // 可选 存储桶名称前缀，前缀搜索
'PageNumber' => 1, // 可选 第几页
'PageSize' => 20, // 可选 每页个数
));
// 请求成功
print_r($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：2022-09-21 15:48:08

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

版权所有：腾讯云计算（北京）有限责任公司

```
// '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

名称	描述	类型	是否必选
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-sign-algorithmxxxxxxxxxxxxx&ci-process=doc-preview&page=1&dstType=png

异步处理任务接口

最近更新时间：2022-09-07 15:08:40

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	类型	描述	是否必填
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	否
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

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->describeDocProcessJob(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'examplejobid', // JobId
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->getDescribeDocProcessJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'QueueId' => 'pd8e422a2ea134165a92f2012ea43****', //拉取该队列 ID 下的任务
'Tag' => 'DocProcess', //任务的 Tag：DocProcess 固定值
// 'NextToken' => '143486', //请求的上下文，用于翻页。上次返回的值
// 'OrderByTime' => 'Desc', //Desc 或者 Asc。默认为 Desc
// 'Size' => 2, //拉取的最大任务数。默认为10。最大为100
// 'States' => 'All', //拉取该状态的任务，以,分割，支持多状态：All、Submitted、Running、Success、Failed、Pause、Cancel。默认为 All。
// 'StartCreationTime' => '2021-10-10T16:20:07+0800', //拉取创建时间大于该时间的任务
// 'EndCreationTime' => '2021-10-10T16:20:07+0800', //拉取创建时间小于该时间的任务
));
// 请求成功
print_r($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：2022-09-07 15:08:45

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

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->describeDocProcessQueues(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
));
// 请求成功
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->updateDocProcessQueue(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // QueueID
'Name' => '',
'QueueID' => '',
'State' => '',
'NotifyConfig' => array(
'Url' => '',
'Type' => '',
'Event' => '',
'State' => '',
),
));
// 请求成功
print_r($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 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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型
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：2022-09-21 15:47:51

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 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-sign-algorithmxxxxxxxxxxxxx&ci-process=doc-preview&dstType=html

内容识别
图片标签

最近更新时间：2022-04-22 12:16:54

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->DetectLabel(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
));
// 请求成功
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

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/exampleobject
[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：2022-04-22 12:16:58

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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\ImageQrcodeTemplate();//创建二维码识别参数模版实例
$imageQrcodeTemplate->setMode(0);//二维码覆盖功能。0表示不开启二维码覆盖，1表示开启二维码覆盖功能，开启后，将对识别出的二维码覆盖上马赛克，默认值0
$picOperationsTemplate = new \Qcloud\Cos\ImageParamTemplate\PicOperationsTransformation();//创建图片持久化处理参数模版实例
$picOperationsTemplate->setIsPicInfo(1);//设置是否返回原图信息，0不返回原图信息，1返回原图信息，默认为0
$picOperationsTemplate->addRule($imageQrcodeTemplate, "resultobject");//设置图片持久化处理参数
$result = $cosClient->putObject(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
'Body' => fopen('path/to/localFile', 'rb'),
'PicOperations' => $picOperationsTemplate->queryString(),//生成图片持久化处理参数
));
// 请求成功
print_r($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
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/exampleobject
[Data] => Array
(
[OriginalInfo] => Array
(
[Key] => exampleobject
[Location] => examplebucket-1250000000.cos.ap-beijing.myqcloud.com/exampleobject
[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.myqcloud.com/resultobject
[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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->Qrcode(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
'Cover' => 1,//二维码覆盖功能
));
// 请求成功
print_r($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/exampleobject
[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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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

图片文字识别（OCR）

最近更新时间：2022-12-14 17:53:03

简介

本文档提供关于通用文字识别的 API 概览和 SDK 示例代码。

API	操作描述
通用文字识别	通用文字识别功能将图片上的文字内容，智能识别为可编辑的文本。

通用文字识别

功能说明

通用文字识别功能（Optical Character Recognition，OCR）基于行业前沿的深度学习技术，将图片上的文字内容，智能识别为可编辑的文本，可应用于随手拍扫描、纸质文档电子化、电商广告审核等多种场景，大幅提升信息处理效率。

方法原型

```
public Guzzle\Service\Resource\Model opticalOcrRecognition(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/64324 通用文字识别
$result = $cosClient->opticalOcrRecognition(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'test01.pdf',
'CiProcess' => 'OCR',
'Type' => 'general',
'LanguageType' => 'zh',
'IsPDF' => 'true',
'PdfPageNumber' => 2,
'IsWord' => 'true',
'EnableWordPolygon' => 'false',
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
```

```
echo($e);
}
```

参数说明

参数名称	描述	类型	是否必选
Key	对象文件名，例如：folder/document.jpg。	String	是
CiProcess	数据万象处理能力，图片文字识别固定为 OCR。	String	是
Type	OCR 的识别类型，有效值为 general，accurate，efficient，fast，handwriting。general 表示通用印刷体识别；accurate 表示印刷体高精度版；efficient 表示印刷体精简版；fast 表示印刷体高速版；handwriting 表示手写体识别。默认值为 general。	String	否
LanguageType	type 值为 general 时有效，表示识别语言类型。支持自动识别语言类型，同时支持自选语言种类，默认中英文混合(zh)，各种语言均支持与英文混合的文字识别。可选值请参见 可识别的语言类型。	String	否
IsPDF	type 值为 general、fast 时有效，表示是否开启 PDF 识别，有效值为 true 和 false，默认值为 false，开启后可同时支持图片和 PDF 的识别。	Boolean	否
PdfPageNumber	type 值为 general、fast 时有效，表示需要识别的 PDF 页面的对应页码，仅支持 PDF 单页识别，当上传文件为 PDF 且 ispdf 参数值为 true 时有效，默认值为1。	Integer	否
IsWord	type 值为 general、accurate 时有效，表示识别后是否需要返回单字信息，有效值为 true 和 false，默认为 false。	Boolean	否
EnableWordPolygon	type 值为 handwriting 时有效，表示是否开启单字的四点定位坐标输出，有效值为 true 和 false，默认值为 false。	Boolean	否

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
(
)

)

[RequestId] => NjJhOThmYjZfAOIUDOIUSDOITkwYTU1Zg==
[ContentType] => application/xml
[ContentLength] => 3144
[Key] => test01.pdf
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/test01.pdf
[Response] => Array
(
[Angel] => 359.99
[Language] => zh
[PdfPageSize] => 5
[RequestId] => NjJhOThmYjZfAOIUDOIUSDOITkwYTU1Zg=
[TextDetections] => Array
```

```
(
[0] => Array
(
[Confidence] => 99
[DetectedText] => 第二页
[ItemPolygon] => Array
(
[Height] => 118
[Width] => 351
[X] => 782
[Y] => 403
)

[Polygon] => Array
(
[0] => Array
(
[X] => 782
[Y] => 403
)

[1] => Array
(
[X] => 1133
[Y] => 403
)

[2] => Array
(
[X] => 1133
[Y] => 521
)

[3] => Array
(
[X] => 782
[Y] => 521
)

)

[Words] => Array
(
[0] => Array
(
[Character] => 第
[Confidence] => 100
[WordCoordPoint] => Array
(
[WordCoordinate] => Array
(
[0] => Array
(
[X] => 796
[Y] => 403
)
```

```
[1] => Array
(
[X] => 885
[Y] => 403
)

[2] => Array
(
[X] => 885
[Y] => 521
)

[3] => Array
(
[X] => 796
[Y] => 521
)

)

)

)

[1] => Array
(
[Character] => 二
[Confidence] => 99
[WordCoordPoint] => Array
(
[WordCoordinate] => Array
(
[0] => Array
(
[X] => 885
[Y] => 403
)

[1] => Array
(
[X] => 1034
[Y] => 403
)

[2] => Array
(
[X] => 1034
[Y] => 521
)

[3] => Array
(
[X] => 885
[Y] => 521
)
```

```
)

)

)

[2] => Array
(
[Character] => 页
[Confidence] => 99
[WordCoordPoint] => Array
(
[WordCoordinate] => Array
(
[0] => Array
(
[X] => 1033
[Y] => 403
)

[1] => Array
(
[X] => 1133
[Y] => 403
)

[2] => Array
(
[X] => 1133
[Y] => 521
)

[3] => Array
(
[X] => 1033
[Y] => 521
)

)

)

)

)

)

[1] => Array
(
[Confidence] => 99
[DetectedText] => 内容
[ItemPolygon] => Array
(
[Height] => 50
```

```
[Width] => 93
[X] => 914
[Y] => 569
)

[Polygon] => Array
(
[0] => Array
(
[X] => 914
[Y] => 569
)

[1] => Array
(
[X] => 1007
[Y] => 572
)

[2] => Array
(
[X] => 1005
[Y] => 622
)

[3] => Array
(
[X] => 912
[Y] => 619
)

)

[Words] => Array
(
[0] => Array
(
[Character] => 内
[Confidence] => 99
[WordCoordPoint] => Array
(
[WordCoordinate] => Array
(
[0] => Array
(
[X] => 919
[Y] => 569
)

[1] => Array
(
[X] => 958
[Y] => 570
)

[2] => Array
```

```
(
[X] => 956
[Y] => 620
)

[3] => Array
(
[X] => 917
[Y] => 619
)

)

)

)

[1] => Array
(
[Character] => 容
[Confidence] => 99
[WordCoordPoint] => Array
(
[WordCoordinate] => Array
(
[0] => Array
(
[X] => 958
[Y] => 570
)

[1] => Array
(
[X] => 1007
[Y] => 572
)

[2] => Array
(
[X] => 1005
[Y] => 622
)

[3] => Array
(
[X] => 956
[Y] => 620
)

)

)

)

)
```

```
)
)
)
)
```

图片质量评分

最近更新时间：2022-12-14 17:52:46

简介

本文档提供关于图片质量评分的 API 概览和 SDK 示例代码。

API	操作描述
图片质量评分	对图片文件进行检测，从多方面评估，最终给出综合可观的清晰度评分和主观的美观度评分。

图片质量评分

功能说明

图片质量评分功能为同步请求方式，您可以通过本接口对图片文件进行检测，从多方面评估，最终给出综合可观的清晰度评分和主观的美观度评分。该接口属于 GET 请求。

方法原型

```
public Guzzle\Service\Resource\Model imageAssessQualityProcess(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 {
// 图片质量评估 - https://cloud.tencent.com/document/product/460/63228
$result = $cosClient->imageAssessQualityProcess(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'exampleobject',
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

参数说明

参数名称	描述	类型	是否必选

参数名称	描述	类型	是否必选
Key	对象文件名，例如：folder/document.jpg。	String	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjM5MmU2ZWJfOTQyAHSODHAOIDHAIOSDz==
[ContentType] => application/xml
[ContentLength] => 317
[LongImage] => false
[BlackAndWhite] => false
[SmallImage] => false
[BigImage] => false
[PureImage] => false
[ClarityScore] => 86
[AestheticScore] => 92
[Key] => test.png
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/test.png
)
```

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：2022-12-14 17:48:33

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：
> 需要 COS PHP SDK v2.4.4 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
单次图片审核	图片审核功能支持同步、异步请求方式，您可以通过本接口对图片文件进行内容审核。该接口属于 GET 请求。
批量图片审核	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。

单次图片审核

功能说明

图片审核功能支持同步、异步请求方式，您可以通过本接口对图片文件进行内容审核。该接口属于 GET 请求。

方法原型

```
public Guzzle\Service\Resource\Model detectImage(array $args = array());
```

请求示例

示例一：审核某个存储桶路径下的图片

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为 https
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
//存储桶图片审核
```

```
$result = $cosClient->detectImage(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'test.png', // 桶文件
'ci-process' => 'sensitive-content-recognition',
// 'BizType' => '', // 可选 定制化策略，不传走默认策略
// 'DetectType' => 'porn,ads', // 可选 不填走默认策略 或 定制化策略，四种参数：porn,ads等，可使用多种规则，注意规则间不要加空格
// 'Interval' => 5, // 可选 审核 GIF 时使用 截帧的间隔
// 'MaxFrames' => 5, // 可选 针对 GIF 动图审核的最大截帧数量，需大于0。
// 'LargeImageDetect' => '',
// 'DataId' => '',
// 'Async' => '',
// 'Callback' => '',
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

示例二：审核图片 URL

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为 https
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
//图片链接审核
$imgUrl = 'https://test.jpg';
$result = $cosClient->detectImage(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '/', // 链接图片资源路径写 / 即可
'ci-process' => 'sensitive-content-recognition',
'DetectUrl' => $imgUrl,
// 'BizType' => '', // 可选 定制化策略，不传走默认策略
// 'DetectType' => 'porn,ads', // 可选 不填走默认策略 或 定制化策略，四种参数：porn,ads 等，可使用多种规则，注意规则间不要加空格
// 'Interval' => 5, // 可选 审核 GIF 时使用 截帧的间隔
// 'MaxFrames' => 5, // 可选 针对 GIF 动图审核的最大截帧数量，需大于0。
// 'LargeImageDetect' => '',
// 'DataId' => '',
// 'Async' => '',
// 'Callback' => '',
));
```

```
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

单次审核参数说明

Request 中的主要参数描述如下：

参数名称	描述	类型	是否必选
Key	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，Key 填写 test/img.jpg。	String	否
ci-process	标识数据处理功能的字段，内容审核的值为：sensitive-content-recognition。	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 biz-type。biz-type 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。 biz-type 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 biz-type 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。	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
Async	请求方式参数，有效值：0（同步返回结果），1（异步进行审核），默认值为0。	Int	否
Callback	回调地址，审核结果（Detail版本）以回调形式发送至您的回调地址，仅在异步审核时生效，支持以 http:// 或者 https:// 开头的地址，例如： http://www.callback.com 。	String	否

####

批量图片审核

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

```
public Guzzle\Service\Resource\Model detectImages(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为 https
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
// 获取图片base64编码
// $localImageFile = '/tmp/test.jpg';
// $img = file_get_contents($localImageFile);
// $imgInfo = getimagesize($localImageFile);
// $imgBase64Content = base64_encode($img);

$result = $cosClient->detectImages(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Inputs' => array(
array(
'Object' => 'test01.png', // 桶文件
// 'Interval' => '', // 可选 审核 GIF 时使用 截帧的间隔
// 'MaxFrames' => '', // 可选 针对 GIF 动图审核的最大截帧数量，需大于0。
// 'DataId' => 'aaa', // 可选 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
),
array(
'Url' => 'http://example.com/test.png', // 图片URL
// 'Interval' => 5, // 可选 审核 GIF 时使用 截帧的间隔
// 'MaxFrames' => 5, // 可选 针对 GIF 动图审核的最大截帧数量，需大于0。
// 'DataId' => 'bbb', // 可选 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
),
// array(
// 'Content' => $imgBase64Content, // 图片文件的内容，需要先经过 base64 编码。
//// 'Interval' => 5, // 可选 审核 GIF 时使用 截帧的间隔
//// 'MaxFrames' => 5, // 可选 针对 GIF 动图审核的最大截帧数量，需大于0。
//// 'DataId' => 'bbb', // 可选 图片标识，该字段在结果中返回原始内容，长度限制为512字节
// ),
),
// 'Conf' => array(
// 'BizType' => '', // 可选 定制化策略，不传走默认策略
// 'DetectType' => 'Porn,Ads', // 可选 不填走默认策略 或 定制化策略，四种参数：Porn,Ads等，可使用多种规则，注意规则间不要加空格
// 'Async' => 0, // 可选 是否异步进行审核，0：同步返回结果，1：异步进行审核。默认值为 0。
// 'Callback' => '', // 可选 审核结果（Detail版本）以回调形式发送至您的回调地址
// ) // 可选 在DetectType/BizType都不传的情况下，走默认策略及默认审核场景。
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

批量审核参数说明

Request 中的主要参数描述如下：

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。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	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。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	String	否
Content	Request.Input	图片文件的内容，需要先经过 base64 编码。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	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
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

版权所有：腾讯云计算（北京）有限责任公司

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，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	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

视频审核

最近更新时间：2022-07-25 08:53:32

简介

本文档提供关于视频审核的 API 概览和 SDK 示例代码。

> 注意：
> 需要 COS PHP SDK v2.4.4 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

```
public Guzzle\Service\Resource\Model detectVideo(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->detectVideo(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Input' => array(
'Object' => 'movie.mp4',
),
'Conf' => array(
'DetectType' => 'Porn,Ads', //可选2种参数：Porn,Ads，可使用多种规则，注意规则间不要加空格
// 'Callback' => 'https://example.com/callback', //回调地址
// 'BizType' => '', //审核策略
// 'DetectContent' => 1, //是否审核视频声音，默认0
// 'CallbackVersion' => 'Detail', //回调内容结构，Simple或Detail，默认Simple
```

```
'Snapshot' => array(
// 'Mode' => 'Average', //截帧模式。Interval,Average,Fps
// 'TimeInterval' => 50, //视频截帧数量
'Count' => '3', //视频截帧频率
),//截取截图
),
));
// 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
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	否
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
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

版权所有：腾讯云计算（北京）有限责任公司

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，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	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
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
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，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	否
Count	Request.Conf.Snapshot	视频截帧数量，范围为(0, 10000]。	Integer	是
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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYTA4YTlfZmNjYTNiMGFfNGM2MV8zMTM1ODQ=
[ContentType] => application/xml
[ContentLength] => 322
[JobsDetail] => Array
(
[Object] => movie.mp4
[JobId] => avd61e00d8116f11ec953452540024deb5
[State] => Submitted
[CreationTime] => 2021-09-09T21:14:17+08:00
)

[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/video/auditing
)
```

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

```
public Guzzle\Service\Resource\Model getDetectVideoResult(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->getDetectVideoResult(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'examplejobid', // JobId
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	需要查询的任务 ID	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYTBjMGVfZmNjYTNiMGFfNGM2Ml8zMTdjYjM=
[ContentType] => application/xml
[ContentLength] => 4941
[Key] => avd61e00d8116f11ec953452540024deb5
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/video/auditing/avd61e00d8116f11ec953452540024deb5
[Response] => Array
(
[JobsDetail] => Array
(
```

版权所有：腾讯云计算（北京）有限责任公司

[AdsInfo] => Array
(
[Count] => 0
[HitFlag] => 0
)

[AudioSection] => Array
(
[0] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Duration] => 30000
[OffsetTime] => 0
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Text] => Array
(
)

[Url] => https://audio-auditing-appid.cos.ap-guangzhou.myqcloud.com/_cms_segments%2Fcms%2Faudio%2Fw-audio-YToIq75SmCBmUefo%2F0.mp3?q-sign-algorithm=sha1&q-ak=AKIDJLGwMW6NTJ9mNj8DigYDiXOLtVpsbzGJ&q-sign-time=1631194126%3B1631201326&q-key-time=1631194126%3B1631201326&q-header-list=host&q-url-param-list=&q-signature=65af69dedc0cea3702059c7b124914311d73b1e7
)

[1] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Duration] => 30000
[OffsetTime] => 30000
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Text] => Array
(
)

[Url] => https://audio-auditing-appid.cos.ap-guangzhou.myqcloud.com/_cms_segments%2Fcms%2Faudio%2Fw-audio-YToIq75SmCBmUefo%2F30.mp3?q-sign-algorithm=sha1&q-ak=AKIDJLGwMW6NTJ9mNj8DigYDiXOLtVpsbzGJ&q-sign-time=1631194126%3B1631201326&q-key-time=1631194126%3B1631201326&q-header-list=host&q-url-param-list=&q-signature=ce3e8856e97b155a661108b0ae970f8fcfddf143

版权所有：腾讯云计算（北京）有限责任公司

```
)

[2] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Duration] => 2000
[OffsetTime] => 60000
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Text] => Array
(
)

[Url] => https://audio-auditing-appid.cos.ap-guangzhou.myqcloud.com/_cms_segments%2Fcms%2Faudio%2Fw-audio-YToIq75SmCBmUefo%2F60.mp3?q-sign-algorithm=sha1&q-ak=AKIDJLGwMW6NTJ9mNj8DigYDiXOLtVpsbzGJ&q-sign-time=1631194126%3B1631201326&q-key-time=1631194126%3B1631201326&q-header-list=host&q-url-param-list=&q-signature=019cf7a9250473e57e45e5d8eb3e569d4d2ecfe2
)

)

[CreationTime] => 2021-09-09T21:14:17+08:00
[JobId] => avd61e00d8116f11ec953452540024deb5
[Object] => movie.mp4
[PornInfo] => Array
(
[Count] => 0
[HitFlag] => 0
)

[Result] => 0
[Snapshot] => Array
(
[0] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
)

[Score] => 0
[SubLabel] => Array
(
)

)
```

[PornInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
)

[Score] => 0
[SubLabel] => Array
(
)

)

[SnapshotTime] => 0
[Text] => Array
(
)

[Url] => https://auditing-appid.cos.ap-chongqing.myqcloud.com/%2F872400-1251668577%2Fmovie.mp4%2Fjd634e582116f11eca3594388b5008703%2F0%2F0.jpg?q-sign-algorithm=sha1&q-ak=AKIDziAUWHqzwb2DsGIJzOD1hHajzyhyglHW&q-sign-time=1631194126%3B1631201326&q-key-time=1631194126%3B1631201326&q-header-list=host&q-url-param-list=&q-signature=d440335fccc8a6db8c278982b8b3fdda9719f970
)

[1] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
)

[Score] => 0
[SubLabel] => Array
(
)

)

[PornInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
)

[Score] => 0
[SubLabel] => Array
(
)

)

版权所有：腾讯云计算（北京）有限责任公司

```
[SnapshotTime] => 50000
[Text] => Array
(
)

[Url] => https://auditing-appid.cos.ap-chongqing.myqcloud.com/%2F872400-1251668577%2Fmovie.mp4%2Fjd634e582116f11eca3594388b5008703%2F50000%2F1.jpg?q-sign-algorithm=sha1&q-ak=AKIDziAUWHqzwb2DsGIJzOD1hHajzyhyglHW&q-sign-time=1631194126%3B1631201326&q-key-time=1631194126%3B1631201326&q-header-list=host&q-url-param-list=&q-signature=f93ae84af5bf5f6730ffeae9edb3332285b8169d
)

)

[SnapshotCount] => 2
[State] => Success

)

[RequestId] => NjEzYTBjMGVfZmNjYTNiMGFfNGM2MI8zMTdjYjM=
)

)
```

音频审核

最近更新时间：2022-07-25 08:53:36

简介

本文档提供关于音频审核的 API 概览和 SDK 示例代码。

> 注意：
> 需要 COS PHP SDK v2.4.4 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

```
public Guzzle\Service\Resource\Model detectAudio(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->detectAudio(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Input' => array(
'Object' => 'sound01.mp3',
),
'Conf' => array(
'DetectType' => 'Porn,Ads',//可选六种参数：Porn,Ads,Illegal,Abuse，可使用多种规则，注意规则间不要加空格
// 'CallbackVersion' => 'Detail', //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。
// 'Callback' => 'https://example.com/callback',//回调地址
// 'BizType' => '',//审核策略
),
```

```
));
// 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
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
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
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，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	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

节点名称（关键字）	父节点	描述	类型	是否必选
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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYTExN2RfNzgwYzdkNjRfMThiOV8zMjY2Zjc=
[ContentType] => application/xml
[ContentLength] => 324
[JobsDetail] => Array
(
[Object] => sound01.mp3
[JobId] => sa19232ce6117511ecb6485254009f6100
[State] => Submitted
[CreationTime] => 2021-09-09T21:51:57+08:00
)

[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/audio/auditing
)
```

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

```
public Guzzle\Service\Resource\Model getDetectAudioResult(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->getDetectAudioResult(array(
```

```
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	需要查询的任务 ID	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYjRjMWRfNzgwYzdkNjRfMThiOV8zNjI5OWI=
[ContentType] => application/xml
[ContentLength] => 2328
[Key] => sa19232ce6117511ecb6485254009f6100
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/audio/auditing/sa19232ce6117511ecb6485254009f6100
[Response] => Array
(
[JobsDetail] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
)

[Score] => 0
)

[AudioText] => Array
(
[0] =>
)

[CreationTime] => 2021-09-09T21:51:57+08:00
[JobId] => sa19232ce6117511ecb6485254009f6100
[Object] => sound01.mp3
[PornInfo] => Array
(
[HitFlag] => 0
[Label] => Array
(
```

```
)

[Score] => 0
)

[Result] => 0
[Section] => Array
(
[0] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Duration] => 30000
[OffsetTime] => 0
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Text] => Array
(
)

[Url] => https://audio-auditing-appid.cos.ap-guangzhou.myqcloud.com/_cms_segments%2Fcms%2Faudio%2Fw-audio-YToRfUy_b0-zRwQH%2F0.mp3?q-sign-algorithm=sha1&q-ak=AKIDJLGwMW6NTJ9mNj8DigYDiXOLtVpsbzGJ&q-sign-time=1631276061%3B1631283261&q-key-time=1631276061%3B1631283261&q-header-list=host&q-url-param-list=&q-signature=5b33e65e53f14ea7ba5559e63a40c920b500f109
)

[1] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Duration] => 15000
[OffsetTime] => 30000
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[Text] => Array
(
)

[Url] => https://audio-auditing-appid.cos.ap-guangzhou.myqcloud.com/_cms_segments%2Fcms%2Faudio%2Fw-audio-YToRfUy_b0-zRwQH%2F30.mp3?q-sign-algorithm=sha1&q-ak=AKIDJLGwMW6NTJ9mNj8DigYDiXOLtVpsbzGJ&q-sign-time=1631276061%3B1631283261&q-key-time=1631276061%3B1631283261&q-header-list=host&q-url-param-list=&q-signature=f8e6fe3b430eeac93fd5bf5950a09f9c4a570e26
```

```
)

)

[State] => Success
)

[RequestId] => NjEzYjRjMWRfNzgwYzdkNjRfMThiOV8zNjI5OWI=
)

)
```

文档审核

最近更新时间：2022-07-25 08:53:41

简介

本文档提供关于文档审核的 API 概览和 SDK 示例代码。

> 注意：
> 需要 COS PHP SDK v2.4.4 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

```
public Guzzle\Service\Resource\Model detectDocument(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->detectDocument(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Input' => array(
'Url' => 'https://example.com/test01.docx', //文档文件的链接地址
'Type' => 'docx', //文档文件类型
),
'Conf' => array(
'DetectType' => 'Porn,Ads', //可选参数：Porn,Ads，可使用多种规则，注意规则间不要加空格
// 'Callback' => 'https://example.com/callback/', //回调地址
// 'BizType' => '', //审核策略
),
```

```
));
// 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
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 存储桶中的文件名称，例如在目录 test 中的文件test.doc，则文件名称为 test/test.doc。Object 和 Url 只能选择其中一种。	String	否
Url	Request.Input	文档文件的链接地址，例如 http://www.example.com/doctest.doc，Object 和 Url 只能选择其中一种。	String	否
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
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
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，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

节点名称（关键字）	父节点	描述	类型	是否必选
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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzZTBjMmRfNzgwYzdkNjRfMThiOV8zZTY4ZGE=
[ContentType] => application/xml
[ContentLength] => 357
[JobsDetail] => Array
(
[Url] => https://example.com/test01.pdf
[JobId] => sd48e1ea1213d411ec953452540024deb5
[State] => Submitted
[CreationTime] => 2021-09-12T22:18:21+08:00
)

[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/document/auditing
)
```

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

```
public Guzzle\Service\Resource\Model getDetectDocumentResult(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
```

```
$result = $cosClient->getDetectDocumentResult(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => 'examplejobid', // JobId
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

参数说明

参数名称	类型	描述	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID。	是
Key	String	需要查询的任务 ID。	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzZTBjZWRfZmNjYTNiMGFfNGM2M18zZGUzNzc=
[ContentType] => application/xml
[ContentLength] => 1725
[Key] => sd48e1ea1213d411ec953452540024deb5
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/document/auditing/sd48e1ea1213d411ec953452540024deb5
[Response] => Array
(
[JobsDetail] => Array
(
[CreationTime] => 2021-09-12T22:18:21+08:00
[JobId] => sd48e1ea1213d411ec953452540024deb5
[Labels] => Array
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
)

)

[PageCount] => 1
[PageSegment] => Array
(
[Results] => Array
```

```
(
[AdsInfo] => Array
(
[HitFlag] => 0
[Score] => 0
[SubLabel] => Array
(
)

)

[PageNumber] => 1
[PornInfo] => Array
(
[HitFlag] => 0
[Score] => 0
[SubLabel] => Array
(
)

)

[Text] => Pdf测试文档 这是测试pdf文档，请勿纠结内容
[Url] => https://auditing-appid.cos.ap-chongqing.myqcloud.com/%2Fdocument%2F872400-1251668577%2Fsd48e1ea1213d411ec953452540024deb5%2Foutput%2F1.jpg?q-sign-algorithm=sha1&q-ak=AKIDziAUWHqzwb2DsGIJzOD1hHajzyhyglHW&q-sign-time=1631456493%3B1631463693&q-key-time=1631456493%3B1631463693&q-header-list=host&q-url-param-list=&q-signature=870d00e9c5a3e847bff53fa32e30bfc0d06bbcf9
)

)

[State] => Success
[Suggestion] => 0
[Url] => https://example.com/test01.pdf
)

[RequestId] => NjEzZTBjZWRfZmNjYTNiMGFfNGM2M18zZGUzNzc=
)

)
```

文本审核

最近更新时间：2022-07-25 08:53:45

简介

本文档提供关于文本审核的 API 概览和 SDK 示例代码。

> 注意：
> 需要 COS PHP SDK v2.4.4 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

```
public Guzzle\Service\Resource\Model detectText(array $args = array());
```

请求示例

示例一：审核文本内容

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$content = '敏感词';
$result = $cosClient->detectText(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Input' => array(
'Content' => base64_encode($content) //文本需base64_encode
),
'Conf' => array(
'DetectType' => 'Porn,Ads,Illegal,Abuse', //Porn,Ads,Illegal,Abuse类型
// 'BizType' => '', //审核策略
```

```
),
));
// 请求成功
print_r($result);
} catch (\Exception $e) {
// 请求失败
echo($e);
}
```

示例二：审核存储桶路径下的文本文件

```
<?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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->detectText(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Input' => array(
'Object' => 'test01.txt'// 对象名，请注意对象是否存在
),
'Conf' => array(
'DetectType' => 'Porn,Ads,Illegal,Abuse',//Porn,Ads,Illegal,Abuse类型
// 'Callback' => 'https://example.callback.com/test/', //回调URL
// 'CallbackVersion' => 'Detail', //回调内容的结构，有效值：Simple（回调内容包含基本信息）、Detail（回调内容包含详细信息）。默认为 Simple。
// 'BizType' => '', //审核策略
),
));
// 请求成功
print_r($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
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 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。	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
Url	Request.Input	文本文件的完整链接，例如：https://www.test.com/test.txt。	String	否
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

注意：

- 单次请求只能使用 Object 、Content、Url 中的一个。
- 当选择 Object、Url 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	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	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	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

返回结果示例

示例一：审核文本内容

```
//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。
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYjUxMDhfZmNjYTNiMGFfNGM2NF8zNWY4Njg=
[ContentType] => application/xml
[ContentLength] => 878
[JobsDetail] => Array
(
[Code] => Success
[Message] => Array
(
)

[JobId] => st8ffc0551123311eca33f525400f3e40d
[State] => Success
[CreationTime] => 2021-09-10T20:35:20+08:00
[Content] => 5pWP5oSf6K+N
[Result] => 0
[SectionCount] => 1
[PornInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[AdsInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[IllegalInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[AbuseInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[Section] => Array
(
[0] => Array
(
)

)

)

[Bucket] => examplebucket-1250000000
```

```
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/text/auditing
)
```

示例二：审核存储桶路径下的文本文件

```
//当选择 Object 时，审核结果为异步返回，可通过 查询文本审核任务接口获取返回结果。
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYjUxY2VfZmNjYTNiMGFfNGM0MF8zNTE0OWI=
[ContentType] => application/xml
[ContentLength] => 324
[JobsDetail] => Array
(
[Object] => exampleobject
[JobId] => st05c02c3c123411ec953452540024deb5
[State] => Submitted
[CreationTime] => 2021-09-10T20:38:38+08:00
)

[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/text/auditing
)
```

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

```
public Guzzle\Service\Resource\Model getDetectTextResult(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 审核时必须为https
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));

try {
$result = $cosClient->getDetectTextResult(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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	是否必填
Bucket	String	存储桶名称，格式：BucketName-APPID	是
Key	String	需要查询的任务 ID	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjEzYjU0NDhfNzgwYzdkNjRfMThiOV8zNjQ0Y2Q=
[ContentType] => application/xml
[ContentLength] => 725
[JobsDetail] => Array
(
[Code] => Success
[Message] =>
[JobId] => st7a813188123511ecb6485254009f6100
[State] => Success
[CreationTime] => 2021-09-10T20:49:03+08:00
[Object] => exampleobject
[SectionCount] => 1
[Result] => 0
[PornInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[AdsInfo] => Array
(
[HitFlag] => 0
[Count] => 0
)

[Section] => Array
(
[0] => Array
(
)

)

)

[Key] => st7a813188123511ecb6485254009f6100
```

```
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/text/auditing/st7a813188123511ecb6485254009f6100
)
```

任务接口

动图任务接口

最近更新时间：2022-12-14 17:49:52

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Animation',
'QueueId' => 'p81e648af2aee49688xxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'TemplateId' => 't1de276cbdab16xxxxxxxxxxxxxxxxxxxxx',
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
```

```
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Animation',
'QueueId' => 'p81e648af2aee49688xxxxxxxxxxxxxxxx',
'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' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'Animation.gif',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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
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。

版权所有：腾讯云计算（北京）有限责任公司

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

```
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
)

)

)
```

截图任务接口

最近更新时间：2022-12-14 17:50:00

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Snapshot',
'QueueId' => 'asdadadfafsdkjhfjghdfjg',
'CallBack' => 'https://example.com/callback',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'TemplateId' => 'asdfafiahfiushdfisdhfuis',
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'snapshot-${Number}.jpg',
```

```
// 'SpriteObject' => 'sprite-${Number}.jpg',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
// 'CallBackMqConfig' => array(
// 'MqRegion' => '',
// 'MqMode' => '',
// 'MqName' => '',
// ),
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Snapshot',
'QueueId' => 'asdadadfafsdkjhfjghdfjg',
'CallBack' => 'https://example.com/callback',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'snapshot-${Number}.jpg',
),
'Snapshot' => array(
'Mode' => 'Average',
'Start' => 3,
'TimeInterval' => '',
```

版权所有：腾讯云计算（北京）有限责任公司

```
'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，默认为0
),
));
// 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：2022-12-14 17:50:10

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'video01.mkv',
),
'Transcode' => array(
'Container' => array(
'Format' => 'mp4'
),
'Video' => array(
'Codec' => 'H.264',
'Profile' => 'high',
'Bitrate' => '1000',
'Preset' => 'medium',
'Width' => '1280',
'Fps' => '30',
),
'Audio' => array(
'Codec' => 'aac',
'Samplerate' => '44100',
'Bitrate' => '128',
'Channels' => '4',
),
'TransConfig' => array(
'AdjDarMethod' => 'scale',
'IsCheckReso' => 'false',
'ResoAdjMethod' => '1',
),
'TimeInterval' => array(
'Start' => '0',
'Duration' => '60',
),
),
),
));
// 请求成功
print_r($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

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 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
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：2022-12-14 17:50:17

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
));
```

```
// 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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：

版权所有：腾讯云计算（北京）有限责任公司

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	限制
ConcatFragment	Request.Operation. ConcatTemplate	拼接节点	Container 数组	否	无	支持多个文件，按照文件顺序拼接
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	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	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	无
AudioMix	Request.Operation. ConcatTemplate	混音参数, 详情见 AudioMix	Container 数组	否	无	仅在 Audio.Remove 为 false 时生效
Index	Request.Operation. ConcatTemplate	Input 节点位于 ConcatFragment 序列索引	String	否	0	不能大于 ConcatFragment 长度
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	true、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

节点名称（关键字）	父节点	描述	类型	是否必选
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] => NjI2MjFhOTHASIUDHAUISDHI8yMjMxNmE=
[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
)

)

)
```

智能封面任务接口

最近更新时间：2022-12-14 17:50:24

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'SmartCover',
'QueueId' => 'p81e648afxxxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
// 'TemplateId' => '', // 使用模版
'SmartCover' => array(
'Format' => '',
'Width' => '',
'Height' => '',
'Count' => '',
'DeleteDuplicates' => '',
),
```

```
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'SmartCover-${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，默认为0
),
'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	1. 值范围：[128，4096] 2. 单位：px
Height	Request.Operation.SmartCover	封面图片高度	String	是	无	1. 值范围：[128，4096] 2. 单位：px
Count	Request.Operation.SmartCover	封面数量	String	否	3	1. 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

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] => 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：2022-04-29 10:39:03

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Transcode', // 可选，单一任务时，优先以Operation.Tag为准，当Operation无Tag参数时，该参数生效
'QueueId' => 'paaf4fce5521a40888a303xxxxxxxxxxxxx',
'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-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'Object' => 'video01.mp4',
),
'WatermarkTemplateId' => array(
't112d18d9b2a9b430e91dxxxxxxxxxxxxx',
),
),
array(
'Tag' => 'Transcode',
'TemplateId' => 't04e1ab86554984f1aa17xxxxxxxxxxxxx',
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'video02.mp4',
),
'WatermarkTemplateId' => array(
't1bf713bb5c6a5496e859axxxxxxxxxxxxx',
),
),
),
));
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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] => 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] => t185z90x8c9z0da5sd67a5a19c
)

[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] => t0bz98xc789zxs9d80a8sd90asd37f83c
[TemplateName] => FLV-HD
[WatermarkTemplateId] => t18zc8z90csd876a78sdas87d8a19c
)

[Progress] => 0
[QueueId] => p81zxc908zaf2aee496a09s8d90as8d90
[StartTime] => -
[State] => Submitted
[Tag] => Transcode
)

)

)

)
```

视频增强任务接口

最近更新时间：2022-12-14 17:50:41

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoProcess',
'QueueId' => 'p81e648afxxxxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'TemplateId' => 't13466f1ea41a14c0xxxxxxxxxxxxx', // 视频增强模板 ID
'TranscodeTemplateId' => 't0b6a845f5e42847bd81xxxxxxxxxxxx', // 转码模板 ID
'WatermarkTemplateId' => 't185e2e24551b24259a0xxxxxxxxxxxx', // 水印模板 ID
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'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，默认为0
),
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoProcess',
'QueueId' => 'p81e648afxxxxxxxxxxxxxxxxx',
'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' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'VideoProcess.flv',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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 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
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

版权所有：腾讯云计算（北京）有限责任公司

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
(
)

)

[RequestId] => NjI2MjNjYHAOIHDOIADJHOIA18yMjFIMWE=
[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
```

版权所有：腾讯云计算（北京）有限责任公司

```
)

[JobId] => jfd27519z8zx90c8z90c7809z9f35eb
[Message] =>
[Operation] => Array
(
[Output] => Array
(
[Bucket] => examplebucket-1250000000
[Object] => VideoProcess.flv
[Region] => ap-beijing
)

[TemplateId] => t13466f1eazx08c09zx8c09zx8e9cd
[TemplateName] => video-process-1
[TranscodeTemplateId] => t0b612860z087xc09zx8c09xz8c095dca38
[WatermarkTemplateId] => t185e2e24zxc780z8c0z98c323c6428a19c
[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：2022-12-14 17:50:48

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoMontage',
'QueueId' => 'p81e648af2aeexxxxxxxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'TemplateId' => 't1fcc3770199e04737axxxxxxxxxxxxxx',
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoMontage',
'QueueId' => 'p81e648af2aeexxxxxxxxxxxxxxxxxxx',
'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' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'VideoMontage.mp4',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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

```
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] => 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
)

)

)
```

人声分离任务接口

最近更新时间：2022-12-14 17:50:54

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
));
// 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
));
// 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

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

```
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] => 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
)

)

)
```

SDR to HDR 任务接口

最近更新时间：2022-12-14 17:51:02

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/60754
$result = $cosClient->createMediaSDRtoHDRJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/c
```

```
os5/bucket
'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，默认为0
),
'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：
>
> 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] => t0b612860a293f410785ba7s8d09a8d09a38
[WatermarkTemplateId] => t185e2e24551b242d09a80d8a0d80428a19c
[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：2022-12-14 17:51:09

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/65999
$result = $cosClient->createMediaDigitalWatermarkJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
'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	数字水印嵌入的字符串信息	string	是	长度不超过64个字符，仅支持中文、英文、数字、_、-和*

节点名称（关键字）	父节点	描述	类型	是否必选	限制
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] => NjI2N2JlMDFfZmNjYAJSDIOAJDOIAJD9IZDYxOA==
[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(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/66007
```

```
$result = $cosClient->createMediaExtractDigitalWatermarkJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'ExtractDigitalWatermark',
'QueueId' => 'p81e648af2aee496885707caxxxxxxxxx',
'Input' => array(
'Object' => 'DigitalWatermark.mp4'
),
'Operation' => array(
'ExtractDigitalWatermark' => array(
'Type' => 'Text',
'Version' => 'V1',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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
(
)

)

[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：2022-12-14 17:51:17

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'SuperResolution',
'QueueId' => 'p81e648af2aee49688570xxxxxxxxxxx',
'Input' => array(
'Object' => 'video01.mp4'
),
'Operation' => array(
'TemplateId' =>'t19ea5e0c0b7054d7b904axxxxxxxxxx',
'TranscodeTemplateId' =>'t0b612860a293f41078xxxxxxxxxx',
'WatermarkTemplateId' =>'t185e2e24551b24259a02xxxxxxxxxx',
'DigitalWatermark' => array(
'Message' => 'xxx',
'Type' => 'Text',
'Version' => 'V1',
```

```
'IgnoreError' => 'true',
),
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'SuperResolution.flv',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'SuperResolution',
'QueueId' => 'p81e648af2aee49688570xxxxxxxxxx',
'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' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'SuperResolution.flv',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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

节点名称（关键字）	父节点	描述	类型	是否必选
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选
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 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
(
)

)

[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] => t19ea5e0c0b7b904a8sd09a8sd09a80sd10
[WatermarkTemplateId] => t19ea5e0c0b7b904a8sd09a8asdasddsd10
[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：2022-12-14 17:51:25

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
'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

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] => 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
)

)

)
```

图片处理任务接口

最近更新时间：2022-12-14 17:51:37

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 = 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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
'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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'PicProcess',
'QueueId' => 'pcf4d6d9e5e734asd0as8d09as8d09a8d0',
'Input' => array(
'Object' => 'test01.png'
),
'Operation' => array(
'PicProcess' => array(
'IsPicInfo' => '',
'ProcessRule' => '',
),
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Object' => 'picprocess.jpg',
),
// 'UserData' => 'xxx', // 透传用户信息
// 'JobLevel' => '0', // 任务优先级，级别限制：0 、1 、2。级别越大任务优先级越高，默认为0
),
'CallBack' => '',
));
// 请求成功
print_r($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
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：2022-12-14 17:51:44

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/67186
$result = $cosClient->createMediaSegmentJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'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，默认为0
),
'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：

版权所有：腾讯云计算（北京）有限责任公司

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 HLS 加密	String	否	false	1. true/false 2. Segment.Format 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
(
)

)

[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
)

)

)
```

视频质量评分任务接口

最近更新时间：2022-12-14 17:51:52

简介

本文档提供关于视频质量评分任务接口的 API 概览和 SDK 示例代码。

API	操作描述
提交视频质量评分任务	提交视频质量评分任务

提交视频质量评分任务

功能说明

提交一个视频质量分析任务。

方法原型

```
public Guzzle\Service\Resource\Model createMediaQualityEstimateJobs(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 {
// 提交视频质量评分任务 https://cloud.tencent.com/document/product/460/76906
$result = $cosClient->createMediaQualityEstimateJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'QualityEstimate',
'QueueId' => 'p81e648af2aee4968857xxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'test.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，默认为0
),
'CallBack' => 'http://xxx.com/callback',
'CallBackFormat' => 'JSON',
// 'CallBackType' => '',
// '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	创建任务的 Tag：QualityEstimate	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 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
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

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

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] => NjM4ZGFkMjVfNzgwYzdASUDIOAUSJDOI2QyMg==
[ContentType] => application/xml
[ContentLength] => 688
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
[Response] => Array
(
[JobsDetail] => Array
(
[Code] => Success
[CreationTime] => 2022-12-05T16:34:45+0800
[EndTime] => -
[Input] => Array
(
[BucketId] => examplebucket-1250000000
[Object] => video01.mp4
[Region] => ap-guangzhou
)

[JobId] => jac0581107a8s09d8a09s8d0as476a55
[Operation] => Array
(
[JobLevel] => 0
[UserData] => xxx
)

[QueueId] => pc6f4d289c9bcasd7a098sd7as0a
[StartTime] => -
[State] => Submitted
[Tag] => QualityEstimate
)

)

)
```

音视频流分离任务接口

最近更新时间：2022-12-14 17:51:58

简介

本文档提供关于音视频流分离任务接口的 API 概览和 SDK 示例代码。

API	操作描述
提交音视频流分离任务	提交音视频流分离任务

提交音视频流分离任务

功能说明

提交一个音视频流分离任务。

方法原型

```
public Guzzle\Service\Resource\Model createMediaStreamExtractJobs(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 {
// 提交一个音视频流分离任务 https://cloud.tencent.com/document/product/460/76911
$result = $cosClient->createMediaStreamExtractJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'StreamExtract',
'QueueId' => 'p81e648af2aee4968857xxxxxxxxxxxxxxx',
'Input' => array(
'Object' => 'test.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，默认为0
'Output' => array(
'Bucket' => 'examplebucket-1250000000',
'Region' => $region,
'StreamExtracts' => array(
array(
'Index' => '0',
```

```
'Object' => 'output/out0.mp4',
),
array(
'Index' => '1',
'Object' => 'output/out1.mp4',
),
),
),
),
'CallBack' => 'http://xxx.com/callback',
'CallBackFormat' => 'JSON',
// 'CallBackType' => '',
// '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	创建任务的 Tag：StreamExtract	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
UserData	Request.Operation	透传用户信息，可打印的 ASCII 码，长度不超过1024	String	否

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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

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
StreamExtract	Request.Operation.Output	结果文件配置信息，最多支持10个流配置	Container	是

Container 类型 StreamExtract 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
Index	Request.Operation.Output.StreamExtract	流的编号，对应媒体信息中的 Response.MediaInfo.Stream.Video.Index 和 Response.MediaInfo.Stream.Audio.Index，详见 获取媒体信息接口，当输入媒体格式为 mxf 格式时，只支持提取 video 流	String	是
Object	Request.Operation.Output.StreamExtract	结果输出文件名	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] => NjM4ZGFlZjBfZAOUSDAUSODIUAOISDk5OA==
[ContentType] => application/xml
[ContentLength] => 905
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
[Response] => Array
(
[JobsDetail] => Array
(
[Code] => Success
[CreationTime] => 2022-12-05T16:42:24+0800
[EndTime] => -
[Input] => Array
(
```

```
[BucketId] => examplebucket-1250000000
[Object] => video01.mp4
[Region] => ap-guangzhou
)

[JobId] => jbd77f6707478asua9s8d09ad804218e
[Operation] => Array
(
[JobLevel] => 0
[Output] => Array
(
[Bucket] => examplebucket-1250000000
[Region] => ap-guangzhou
[StreamExtract] => Array
(
[0] => Array
(
[Index] => 0
[Output] => tmp/out0.mp4
)

[1] => Array
(
[Index] => 1
[Output] => tmp/out1.mp4
)

)

)

[UserData] => xxx
)

[QueueId] => pc6f4d289c9bc4as8d09as8d09asdfb5a
[StartTime] => -
[State] => Submitted
[Tag] => StreamExtract
)

)

)
```

音频降噪任务接口

最近更新时间：2022-12-14 17:52:04

简介

本文档提供关于音频降噪任务接口的 API 概览和 SDK 示例代码。

API	操作描述
提交音频降噪任务	提交音频降噪任务

提交音频降噪任务

功能说明

提交一个音频降噪任务。

方法原型

```
public Guzzle\Service\Resource\Model createMediaNoiseReductionJobs(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的 region 可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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->createMediaNoiseReductionJobs(array(
'Bucket' => 'examplebucket-1250000000', //存储桶名称，由 BucketName-Appid 组成，可以在 COS 控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'NoiseReduction',
'QueueId' => 'p81e648af2aee4968857xxxxxxxxxxxxxxx',
'CallBack' => '',
'Input' => array(
'Object' => 'sound01.mp3',
),
'Operation' => array(
'Output' => array(
'Region' => $region,
'Bucket' => 'examplebucket-1250000000',
'Object' => 'NoiseReduction.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，默认为0
),
```

```
));
// 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：NoiseReduction	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	执行音频降噪任务的媒体文件名 1.目前只支持文件大小在10M之内的音频 2. 如果输入为视频文件或者多通道的音频，只会保留单通道的音频流 3. 目前暂不支持 m3u8格式输入	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
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 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] => NjM4ZGE3Y2JfNzgwYASODJHOASDJOASTk2Yw==
[ContentType] => application/xml
[ContentLength] => 754
[Bucket] => examplebucket-1250000000
[Location] => examplebucket-1250000000.ci.ap-guangzhou.myqcloud.com/jobs
[Response] => Array
(
[JobsDetail] => Array
(
[Code] => Success
[CreationTime] => 2022-12-05T16:11:55+0800
[EndTime] => -
[Input] => Array
(
[BucketId] => examplebucket-1250000000
[Object] => sound01.mp3
[Region] => ap-guangzhou
)

[JobId] => j7b807dd674741sa098d09a8sd09asd560
[Operation] => Array
(
[JobLevel] => 0
[Output] => Array
(
[Bucket] => examplebucket-1250000000
[Object] => tmp/NoiseReduction.mp3
[Region] => ap-guangzhou
)

[UserData] => xxx
)

[QueueId] => pc6fxd289c9a8s0d8as09d8a09d80sad
[StartTime] => -
[State] => Submitted
[Tag] => NoiseReduction
)

)

)
```

工作流接口
删除工作流

最近更新时间：2022-07-25 08:54:26

简介

本文档提供关于删除工作流的 API 概览和 SDK 示例代码。

API	操作描述
删除工作流	用于删除工作流。

删除工作流

功能说明

用于删除工作流。

方法原型

```
public Guzzle\Service\Resource\Model deleteWorkflow(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/53990 删除工作流
$result = $cosClient->deleteWorkflow(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // WorkflowId
));
// 请求成功
print_r($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	需要被删除的工作流 ID	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhOTU1ODZfNzAHOISDUOAID8xNGYxYTY=
[ContentType] => application/xml
[ContentLength] => 188
[WorkflowId] => wa9ef96cd98d09as8d01dasd9809
[Key] => wa9ef96cd98d09as8d01dasd9809
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/workflow/wa9ef96cd98d09as8d01dasd9809
)
```

版权所有：腾讯云计算（北京）有限责任公司

批量触发存量任务

最近更新时间：2022-07-25 08:54:30

简介

本文档提供关于批量触发存量任务的 API 概览和 SDK 示例代码。

API	操作描述
手动触发工作流	用于手动触发工作流。
触发批量存量任务	用于提交一个存量触发工作流任务。
批量拉取存量任务	用于拉取符合条件的任务。
查询存量任务	用于查询指定存量触发工作流的任务。
取消存量任务	用于取消一个存量触发工作流任务。

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 Guzzle\Service\Resource\Model triggerWorkflow(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 默认http
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
// https://cloud.tencent.com/document/product/436/54641 手动触发工作流
$result = $cosClient->triggerWorkflow(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'workflowId' => 'w9938ed4b1435448783xxxxxxxxxxxxx',
'object' => 'test01.png',
// 'name' => 'xxx',
));
// 请求成功
print_r($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
workflowId	需要触发的工作流 ID	String	是
object	需要进行工作流处理的对象名称	String	是
name	存量触发任务名称，支持中文、英文、数字、一和_，长度限制128字符，默认为空	String	否

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhN2ZkYmZfNzgwYAASDKJAPODJMPFNFM=
[ContentType] => application/xml
[ContentLength] => 188
[InstanceId] => i7fb6e17cebds8v7x897cv89x7v98d6f
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/triggerworkflow
)
```

触发批量存量任务

功能说明

用于提交一个存量触发工作流任务。

方法原型

```
public Guzzle\Service\Resource\Model createInventoryTriggerJob(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/71516 触发批量存量任务
$result = $cosClient->createInventoryTriggerJob(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'Name' => '存量触发任务名称',
'Input' => array(
// 'Manifest' => '',
// 'UrlFile' => '',
// 'Prefix' => '',
'Object' => 'test01.png',
),
'Operation' => array(
'WorkflowIds' => 'w9938ed4b1435448783xxxxxxxxxxxx',
// 'TimeInterval' => array(
// 'Start' => '',
// 'End' => '',
// ),
),
));
// 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
Name	Request	存量触发任务名称，支持中文、英文、数字、—和_，长度限制128字符	String	是
Input	Request	待操作的媒体信息	Container	是
Operation	Request	操作规则	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
Manifest	Request.Input	COS 清单列表文件名	String	否
UrlFile	Request.Input	URL 文件名	String	否
Prefix	Request.Input	Object 前缀	String	否
Object	Request.Input	媒体文件名	String	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
WorkflowIds	Request.Operation	触发的工作流 ID	String	是
TimeInterval	Request.Operation	按时间过滤的触发范围	Container	否

Container 类型 TimeInterval 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
Start	Request.TimeInterval	开始时间	String	否	0	扫描对象的上传时间 %Y-%m-%dT%H:%m:%S%z
End	Request.TimeInterval	结束时间	String	否	当前时间	扫描对象的上传时间 %Y-%m-%dT%H:%m:%S%z

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
[uri:GuzzleHttp\Psr7\Stream:private] => php://temp
[customMetadata:GuzzleHttp\Psr7\Stream:private] => Array
(
)

)

[RequestId] => NjJhOTg3ODNfZmNjYTNiOHAODISHOdlMmI=
[ContentType] => application/xml
[ContentLength] => 576
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/inventorytriggerjob
[Response] => Array
(
[JobsDetail] => Array
(
[Code] => Success
[CreationTime] => 1-01-01T08:05:43+0805
[EndTime] => -
[Input] => Array
(
[Object] => test01.png
)

[JobId] => b33872336ec7b11ecas0d80asd30d6f
[Name] => 存量触发任务名称
[Operation] => Array
(
[TimeInterval] => Array
(
[End] => 2022-06-15T15:17:23+08:00
)

)

[StartTime] => 1-01-01T08:05:43+0805
[State] => Running
[WorkflowIds] => w9938ed4b14aas890d8as09ddd
)

)

)
```

批量拉取存量任务

功能说明

用于拉取符合条件的任务。

方法原型

```
public Guzzle\Service\Resource\Model describeInventoryTriggerJobs(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/71518 批量拉取存量任务
$result = $cosClient->describeInventoryTriggerJobs(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
// 'NextToken' => '',
// 'Size' => '',
// 'OrderByTime' => '',
// 'States' => '',
// 'StartCreationTime' => '',
// 'EndCreationTime' => '',
// 'WorkflowId' => '',
// 'JobId' => '',
// 'Name' => '',
));
// 请求成功
print_r($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
NextToken	无	请求的上下文，用于翻页。上次返回的值	String	否
Size	无	拉取的最大任务数。默认为10。最大为100	Integer	否
OrderByTime	无	Desc 或者 Asc。默认为 Desc	string	否

节点名称（关键字）	父节点	描述	类型	必选
States	无	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
StartCreationTime	无	拉取创建时间大于等于该时间的任务。格式为：%Y-%m-%dT%H:%m:%S%z	String	否
EndCreationTime	无	拉取创建时间小于等于该时间的任务。格式为：%Y-%m-%dT%H:%m:%S%z	String	否
WorkflowId	无	工作流 ID	string	否
JobId	无	存量触发任务 ID	string	否
Name	无	存量触发任务名称	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] => NjJhOTg4YzRfZmNjAIODUOIDUxNWU0OTM=
[ContentType] => application/xml
[ContentLength] => 3808
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/inventorytriggerjob
[Response] => Array
(
[JobsDetail] => Array
(
[0] => Array
(
[Code] => Success
[CreationTime] => 2022-06-15T15:17:23+0800
[EndTime] => 2022-06-15T15:21:17+0800
[Input] => Array
(
[Object] => test01.png
)

[JobId] => b33872336ec7b11ecas8d90as8d09ad6f
[Name] => 存量触发任务名称
[Operation] => Array
(
[TimeInterval] => Array
(
```

```
[End] => 2022-06-15T15:17:23+08:00
)

)

[StartTime] => 2022-06-15T15:17:23+0800
[State] => Success
[WorkflowIds] => w9938edb1asd80a9s8d09asddebdd
)

[1] => Array
(
[Code] => Success
[CreationTime] => 2022-06-14T11:17:19+0800
[EndTime] => 1-01-01T08:05:43+0805
[Input] => Array
(
[Object] => test01.png
)

[JobId] => b7fca9011ec9e215a09s8d09as8d6f
[Name] => 存量触发任务名称-5
[Operation] => Array
(
[TimeInterval] =>
)

[StartTime] => 2022-06-14T11:17:19+0800
[State] => Success
[WorkflowIds] => w993asd81as98d09a8d762bddebdd
)

[2] => Array
(
[Code] => Success
[CreationTime] => 2022-05-31T17:01:43+0800
[EndTime] => -
[Input] => Array
(
[Object] => test01.png
)

[JobId] => w993asd81as98d09a8d762bddebdd
[Name] => 存量触发任务名称-3
[Operation] => Array
(
[TimeInterval] => Array
(
[End] => 2022-05-31T17:01:43+08:00
)

)

[StartTime] => 2022-05-31T17:01:43+0800
[State] => Cancel
[WorkflowIds] => w993asd81as98d09a8d76as890d0debdd
```

```
)
)

[NextToken] => 1930
[RequestId] => NjJhOTg4YzRfZmNjYTNAOISUDOIASUDWU0OTM=
)

)
```

查询存量任务

功能说明

用于查询指定存量触发工作流的任务。

方法原型

```
public Guzzle\Service\Resource\Model describeInventoryTriggerJob(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/71517 查询存量任务
$result = $cosClient->describeInventoryTriggerJob(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // jobId
));
// 请求成功
print_r($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

参数名称	类型	描述	是否必填
Key	String	需要查询的任务 ID	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] => NjJhOThhNjhfNzgwYzAUOIDUSIOUDIOQ3NGE=
[ContentType] => application/xml
[ContentLength] => 670
[Key] => b33872336ec7b11as8d9000030d6f
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/inventorytriggerjob/b33872336ec7b11as8d9000030d6f
[Response] => Array
(
[JobsDetail] => Array
(
[Code] => Success
[CreationTime] => 2022-06-15T15:17:23+0800
[EndTime] => 2022-06-15T15:21:17+0800
[Input] => Array
(
[Object] => test01.png
)

[JobId] => b33872336ec7b11as8d9000030d6f
[Name] => 存量触发任务名称
[Operation] => Array
(
[TimeInterval] => Array
(
[End] => 2022-06-15T15:17:23+08:00
)

)

[StartTime] => 2022-06-15T15:17:23+0800
[State] => Success
[WorkflowIds] => w9938eda8s09d8a9s0d809a5762bddebdd
)
```

```
[RequestId] => NjJhOThhNjhfNAOUSDOIDUo3NGE=
)

)
```

取消存量任务

功能说明

用于取消一个存量触发工作流任务。

方法原型

```
public Guzzle\Service\Resource\Model cancelInventoryTriggerJob(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/71515 取消存量任务
$result = $cosClient->cancelInventoryTriggerJob(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // jobId
));
// 请求成功
print_r($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	需要取消的任务 ID	是

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhOThjOWFfNzgwYzOUADOIID8xNWU3ODY=
[ContentType] => application/xml
[ContentLength] => 0
[Key] => b3c9das80d98a00f2ffd9
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/inventorytriggerjob/b3c9das80d98a00f2ffd9
)
```

搜索工作流

最近更新时间：2022-07-25 08:54:35

简介

本文档提供关于搜索工作流的 API 概览和 SDK 示例代码。

API	操作描述
搜索工作流	用于搜索工作流。

搜索工作流

功能说明

用于搜索工作流。

方法原型

```
public Guzzle\Service\Resource\Model describeWorkflow(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/53991 搜索工作流
$result = $cosClient->describeWorkflow(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
// 'Ids' => '',
// 'Name' => '',
// 'PageNumber' => '',
// 'PageSize' => '',
));
// 请求成功
print_r($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：

参数名称（关键字）	描述	类型	是否必选
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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhOTU0MWZfZmNjYTADUSIANDlINGI=
[ContentType] => application/xml
[ContentLength] => 2022
[TotalCount] => 1
[PageNumber] => 1
[PageSize] => 10
[MediaWorkflowList] => Array
(
[0] => Array
(
[BucketId] => examplebucket-125000000
[CreateTime] => 2022-05-24T15:54:35+0800
[Name] => workflow-1
[State] => Paused
[Topology] => Array
(
[Dependencies] => Array
(
[PicProcess_165337887999] => End
[Start] => PicProcess_165337887999
)

[Nodes] => Array
(
[PicProcess_1653378870999] => Array
(
[Operation] => Array
(
[IsTerminal] => false
[Output] => Array
(
[AuObject] =>
[Bucket] => examplebucket-125000000
[Object] => ${InputPath}${InputName}_${RunId}.${ext}
[Region] => ap-guangzhou
[SpriteObject] =>
)

[TemplateId] => t16as98d0asdf76c8d9asd789ad78a09
[TranscodeTemplateId] =>
[WatermarkTemplateId] =>
)
```

```
[Type] => PicProcess
)

[Start] => Array
(
[Input] => Array
(
[ExtFilter] => Array
(
[AllFile] => false
[Audio] => false
[ContentType] => false
[Custom] => false
[CustomExts] =>
[Image] => true
[State] => On
[Video] => false
)

[NotifyConfig] => Array
(
[Event] => TaskFinish,WorkflowFinish
[ResultFormat] => JSON
[State] => On
[Type] => Url
[Url] => http://www.xxx.com/xxx
)

[ObjectPrefix] => /workflow/
[PicProcessQueueId] => pcf4d6d9d0b8as09d88a09d06ad0
[QueueId] => p81eaee496885707d8a09s8de086
)

[Type] => Start
)

)

)

[UpdateTime] => 2022-05-24T15:54:35+0800
[WorkflowId] => w9938eas908d0a98d090as8d09abdd
[type] => 0
[ver] =>

)

)

[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/workflow
)
```

获取工作流实例列表

最近更新时间：2022-07-25 08:54:40

简介

本文档提供关于获取工作流实例列表的 API 概览和 SDK 示例代码。

API	操作描述
获取工作流实例列表	用于获取工作流实例列表。

获取工作流实例列表

功能说明

用于获取工作流实例列表。

方法原型

```
public Guzzle\Service\Resource\Model getWorkflowInstances(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 默认http
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
// https://cloud.tencent.com/document/product/436/53993 获取工作流实例列表
$result = $cosClient->getWorkflowInstances(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'workflowId' => 'w9938ed4b1435448783xxxxxxxxxxxxx',
// 'name' => '',
// 'orderByTime' => '',
// 'size' => '',
// 'states' => '',
// 'startCreationTime' => '',
// 'endCreationTime' => '',
// 'nextToken' => '',
));
// 请求成功
print_r($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（关键字）	描述	类型	是否必选
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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhODAxZThfZmNjAJSIOPDJAOIJDOIAD5NWQ=
[ContentType] => application/xml
[ContentLength] => 1517
[NextToken] => 214283246
[WorkflowExecutionList] => Array
(
[0] => Array
(
[RunId] => i7fb6e17ceb9011ec9eas7d9asf9sa8f90sdf6f
[WorkflowId] => w9938ed4b1435448783a2asd809a8ds0a9ddd
[State] => Success
[CreateTime] => 2022-06-14T11:17:19+08:00
[Object] => test01.png
)

[1] => Array
(
[RunId] => i7fb6e17ceb9011ec9eas7d9asf9sa8f90sdf6f
[WorkflowId] => w9938ed4b1435448783a2asd809a8ds0a9ddd
[State] => Success
[CreateTime] => 2022-05-31T17:01:41+08:00
[Object] => test01.png
)

[2] => Array
(
[RunId] => i7fb6e17ceb9011ec9eas7d9asf9sa8f90sdf6f
[WorkflowId] => w9938ed4b1435448783a2asd809a8ds0a9ddd
[State] => Success
[CreateTime] => 2022-05-31T11:48:41+08:00
[Object] => test01.png
```

```
)

[3] => Array
(
[RunId] => i7fb6e17ceb9011ec9eas7d9asf9sa8f90sdf6f
[WorkflowId] => w9938ed4b1435448783a2asd809a8ds0a9ddd
[State] => Success
[CreateTime] => 2022-05-31T11:33:29+08:00
[Object] => test01.png
)

)

[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/workflowexecution
)
```

获取工作流实例详情

最近更新时间：2022-07-25 08:54:44

简介

本文档提供关于获取工作流实例详情的 API 概览和 SDK 示例代码。

API	操作描述
获取工作流实例详情	用于获取工作流实例详情。

获取工作流实例详情

功能说明

用于获取工作流实例详情。

方法原型

```
public Guzzle\Service\Resource\Model getWorkflowInstance(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$cosClient = new Qcloud\Cos\Client(
array(
'region' => $region,
'schema' => 'https', // 默认http
'credentials' => array(
'secretId' => $secretId,
'secretKey' => $secretKey)));
try {
// https://cloud.tencent.com/document/product/436/53992 获取工作流实例详情
$result = $cosClient->getWorkflowInstance(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // RunId
));
// 请求成功
print_r($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

参数名称	类型	描述	是否必填
Key	String	需要查询的工作流实例 ID	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] => NjJhODJjOTZfNzgwYzdkNASDOIDUIOSDASDSLKJLDKGYyOTA=
[ContentType] => application/xml
[ContentLength] => 1947
[Key] => i7fb6e17ceb90zx8cv09x7v98xc79v8x030d6f
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/workflowexecution/i7fb6e17ceb90zx8cv09x7v98xc79v8x030d6f
[Response] => Array
(
[RequestId] => NjJhODJjOTZfNzgwYzdkNASDOIDUIOSDASDSLKJLDKGYyOTA=
[WorkflowExecution] => Array
(
[CreateTime] => 2022-06-14T11:17:19+08:00
[Object] => test01.png
[RunId] => i7fb6e17ceb90zx8cv09x7v98xc79v8x030d6f
[State] => Success
[Tasks] => Array
(
[Code] => Success
[CreateTime] => 2022-06-14T11:17:19+08:00
[EndTime] => 2022-06-14T11:17:19+08:00
[JobId] => c7fe16e8eeb90bcv098b90cb80957a7abbe
[Name] => PicProcess_16533788711116
[StartTime] => 2022-06-14T11:17:19+08:00
[State] => Success
[Type] => PicProcess
)

[Topology] => Array
(
[Dependencies] => Array
(
[PicProcess_16533788711116] => End
[Start] => PicProcess_16533788711116
)
```

```
[Nodes] => Array
(
[PicProcess_16533788711116] => Array
(
[Operation] => Array
(
[Output] => Array
(
[Bucket] => examplebucket-125000000
[Object] => ${InputPath}${InputName}_${RunId}.${ext}
[Region] => ap-guangzhou
)

[TemplateId] => t1648745f76c354e8a8c90zxd8cv98a09
)

[Type] => PicProcess
)

[Start] => Array
(
[Input] => Array
(
[ExtFilter] => Array
(
[Image] => true
[State] => On
)

[NotifyConfig] => Array
(
[Event] => TaskFinish,WorkflowFinish
[ResultFormat] => JSON
[State] => On
[Type] => Url
[Url] => http://127.0.0.1/xxx
)

[ObjectPrefix] => /workflow/
[PicProcessQueueId] => pcf4d6e734d0bas8d09a8d09ad06ad0
[QueueId] => pcf4d6e734asdad09a8d09ad06ad0
)

[Type] => Start
)

)

)

[WorkflowId] => w9938ed4b1435448asd890a8d09addebdd
[WorkflowName] => workflow-1
)

)
```

```
)
```

模板接口
动图模板接口

最近更新时间：2022-07-25 08:55:11

简介

本文档提供关于动图模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增动图模板	用于新增动图模板。
更新动图模板	用于更新动图模板。

新增动图模板

功能说明

用于新增动图模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaAnimationTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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->createMediaAnimationTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Animation',
'Name' => 'Animation-Template-Name',
'Container' => array(
'Format' => '',
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'AnimateOnlyKeepKeyFrame' => '',
'AnimateTimeIntervalOfFrame' => '',
'AnimateFramesPerSecond' => '',
'Quality' => '',
),
'TimeInterval' => array(
'Start' => '',
'Duration' => '',
```

```
),
));
// 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	模板类型：Animation	String	是
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

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	是否必选
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	是否必选	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 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 。若 AnimateOnlyKeepKeyFrame 设置为 true 时，则不考虑 AnimateTimeIntervalOfFrame、AnimateFramesPerSecond；若 AnimateOnlyKeepKeyFrame 设置为 false 时，则必须填写 AnimateTimeIntervalOfFrame 或 AnimateFramesPerSecond	String	否	false	true、false 动图保留关键帧参数优先级：AnimateFramesPerSecond > AnimateOnlyKeepKeyFrame > AnimateTimeIntervalOfFrame

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
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	是否必选	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

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] => NjJhODRiZDBfNzgwYzdlOSJAIODJOIASDThmZDE=
[ContentType] => application/xml
[ContentLength] => 964
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODRiZDBfNzgwYzdlOSJAIODJOIASDThmZDE=
[Template] => Array
(
[TemplateId] => t1562e64115f68416e8as90d8a90d8dc66b
[Name] => Animation-Template-Name
[State] => Normal
[Tag] => Animation
```

```
[CreateTime] => 2022-06-14T16:50:24+0800
[UpdateTime] => 2022-06-14T16:50:24+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[TransTpl] => Array
(
[Container] => Array
(
[Format] => gif
)

[Video] => Array
(
[Codec] => gif
[Width] => 400
[Height] => 400
[Fps] => 30
[AnimateOnlyKeepKeyFrame] => true
[AnimateTimeIntervalOfFrame] => 3
[AnimateFramesPerSecond] => 3
[Quality] => 20
)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 30
)

)

)

)

)
```

更新动图模板

功能说明

用于更新动图模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaAnimationTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
```

```
$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://cloud.tencent.com/document/product/436/54028 更新动图模板
$result = $cosClient->updateMediaAnimationTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'Animation',
'Name' => 'Animation-Template-Name',
'Container' => array(
'Format' => '',
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'AnimateOnlyKeepKeyFrame' => '',
'AnimateTimeIntervalOfFrame' => '',
'AnimateFramesPerSecond' => '',
'Quality' => '',
),
'TimeInterval' => array(
'Start' => '',
'Duration' => '',
),
));
// 请求成功
print_r($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	Request	模板类型：Animation	String	是
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

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

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 帧率超过60时，设置为60 用户可以设置 fps，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	false	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

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] => NjJhODRjNjRfZmNjYTNiMAOIDOIASDUJOIWIyOWY=
[ContentType] => application/xml
[ContentLength] => 966
[Key] => t1562e64115f68416eas9d8a09d809addc66b
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t1562e64115feas9d8a809addc66b
[Response] => Array
(
[RequestId] => NjJhODRjNjRfZmNjYTNiMAOIDOIASDUJOIWIyOWY=
[Template] => Array
(
[TemplateId] => t1562e6411eas9d8a09d809addc66b
[Name] => Animation-Template-Name
[State] => Normal
[Tag] => Animation
[CreateTime] => 2022-06-14T16:50:24+0800
[UpdateTime] => 2022-06-14T16:52:52+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[TransTpl] => Array
(
[Container] => Array
(
[Format] => gif
)

[Video] => Array
(
[Codec] => gif
[Width] => 400
[Height] => 400
[Fps] => 30
[AnimateOnlyKeepKeyFrame] => true
[AnimateTimeIntervalOfFrame] => 3
[AnimateFramesPerSecond] => 3
[Quality] => 20
)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 30
)

)

)

)

)
```

截图模板接口

最近更新时间：2022-07-25 08:55:15

简介

本文档提供关于截图模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增截图模板	用于新增截图模板。
更新截图模板	用于更新截图模板。

新增截图模板

功能说明

用于新增截图模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaSnapshotTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54029 新增截图模板
$result = $cosClient->createMediaSnapshotTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Snapshot',
'Name' => 'Snapshot-Template-Name',
'Snapshot' => array(
'Mode' => '',
'Start' => '',
'TimeInterval' => '',
'Count' => '',
'Width' => '',
'Height' => '',
),
));
// 请求成功
print_r($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	模板类型：Snapshot	String	是
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
Snapshot	Request	截图	Container	是

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	默认值	限制
Mode	Request.Snapshot	截图模式	String	否	Interval	值范围：{Interval, Average} 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	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
TimeInterval	Request.Snapshot	截图时间间隔	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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhODMwYWZfNzgwYzAIUSIODZCHIOUHDIMGQ5ZjI=
[ContentType] => application/xml
[ContentLength] => 879
[Template] => Array
(
[State] => Normal
[BucketId] => examplebucket-125000000
[Category] => Custom
[Tag] => Snapshot
[Name] => Snapshot-Template-Name
[TemplateId] => t11c298b96a8zxc890z8c90z8c09699
```

版权所有：腾讯云计算（北京）有限责任公司

```
[UpdateTime] => 2022-06-14T14:54:39+0800
[CreateTime] => 2022-06-14T14:54:39+0800
[Snapshot] => Array
(
[IsCheckCount] => false
[BlackLevel] => 0
[PixelBlackThreshold] => 0
[IsCheckBlack] => false
[SnapshotOutMode] => OnlySnapshot
[Mode] => Interval
[Start] => 0
[TimeInterval] => 3
[Count] => 10
[Width] => 400
[Height] => 400
)

)

[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
)
```

更新截图模板

功能说明

用于更新截图模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaSnapshotTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54032 更新截图模板
$result = $cosClient->updateMediaSnapshotTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'Snapshot',
```

```
'Name' => 'Snapshot-Template-Name',
'Snapshot' => array(
'Mode' => '',
'Start' => '',
'TimeInterval' => '',
'Count' => '',
'Width' => '',
'Height' => '',
),
));
// 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	Request	模板类型：Snapshot	String	是
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
Snapshot	Request	截图	Container	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	必选	默认值	限制
Mode	Request.Snapshot	截图模式	String	是	Interval	值范围：{Interval, Average} 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	截图时间间隔	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

返回结果示例

```
GuzzleHttp\Command\Result Object
(
[RequestId] => NjJhODMxYjFfZmNjYTNASHCU67JClOZJmQ=
[ContentType] => application/xml
[ContentLength] => 881
[Template] => Array
(
[State] => Normal
[BucketId] => examplebucket-125000000
[Category] => Custom
[Tag] => Snapshot
[Name] => Snapshot-Template-Name
[TemplateId] => t11c298b96azx9c8z9x08c0z98c09699
[UpdateTime] => 2022-06-14T14:58:57+0800
[CreateTime] => 2022-06-14T14:54:39+0800
[Snapshot] => Array
(
[IsCheckCount] => false
[BlackLevel] => 0
[PixelBlackThreshold] => 0
[IsCheckBlack] => false
[SnapshotOutMode] => OnlySnapshot
[Mode] => Interval
[Start] => 0
[TimeInterval] => 3
[Count] => 10
[Width] => 400
[Height] => 400
)

)

[Key] => t11c298b96azx9c8z9x08c0z98c09699
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t11c298b9f976f4f6a86f6d506f17b9699
)
```

水印模板接口

最近更新时间：2022-07-25 08:55:19

简介

本文档提供关于水印模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增水印模板	用于新增水印模板。
更新水印模板	用于更新水印模板。

新增水印模板

功能说明

用于新增水印模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaWatermarkTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54033 新增水印模板
// 文本水印
$result = $cosClient->createMediaWatermarkTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Watermark',
'Name' => 'Watermark-Template-Name',
'Watermark' => array(
'Type' => 'Text',
'Pos' => 'TopRight',
'LocMode' => 'Absolute',
'Dx' => '128',
'Dy' => '128',
'StartTime' => '',
'EndTime' => '',
'Text' => array(
'FontSize' => '30',
```

```
'FontType' => 'simfang',
'FontColor' => '0x000000',
'Transparency' => '30',
'Text' => '水印内容',
),
),
));
// 请求成功
print_r($result);

// 图片水印
$result = $cosClient->createMediaWatermarkTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Watermark',
'Name' => 'Watermark-Template-Name',
'Watermark' => array(
'Type' => 'Image',
'Pos' => 'TopRight',
'LocMode' => 'Absolute',
'Dx' => '128',
'Dy' => '128',
'StartTime' => '',
'EndTime' => '',
'Image' => array(
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.png',
'Mode' => 'Proportion',
'Width' => '10',
'Height' => '',
'Transparency' => '100',
'Background' => '',
),
),
));
// 请求成功
print_r($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	Request	模板类型: Watermark	String	是
Name	Request	模板名称，仅支持中文、英文、数字、_、-和*	String	是
Watermark	Request	水印信息	Container	是

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

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	基准位置	String	是	无	TopRight、TopLeft、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	1. 在图片水印中，如果 Background 为 true，当 locMode 为 Relativity 时，为%，值范围：[-300 0]；当 locMode 为 Absolute 时，为 px，值范围：[-4096 0] 2. 在图片水印中，如果 Background 为 false，当 locMode 为 Relativity 时，为%，值范围：[0 100]；当 locMode 为 Absolute 时，为 px，值范围：[0 4096] 3. 在文字水印中，当 locMode 为 Relativity 时，为%，值范围：[0 100]；当 locMode 为 Absolute 时，为 px，值范围：[0 4096]
Dy	Request.Watermark	垂直偏移	String	是	无	1. 在图片水印中，如果 Background 为 true，当 locMode 为 Relativity 时，为%，值范围：[-300 0]；当 locMode 为 Absolute 时，为 px，值范围：[-4096 0] 2. 在图片水印中，如果 Background 为 false，当 locMode 为 Relativity 时，为%，值范围：[0 100]；当 locMode 为 Absolute 时，为 px，值范围：[0 4096] 3. 在文字水印中，当 locMode 为 Relativity 时，为%，值范围：[0 100]；当 locMode 为 Absolute 时，为 px，值范围：[0 4096]
StartTime	Request.Watermark	水印开始时间	String	否	0	1. [0 视频时长] 2. 单位为秒 3. 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	1. [0 视频时长] 2. 单位为秒 3. 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（需要 Urlencode 后传入）	String	是	无	同 bucket 的水印图片地址
Mode	Request.Watermark.Image	尺寸模式	String	是	无	1. Original：原有尺寸 2. Proportion：按比例 3. Fixed：固定大小
Width	Request.Watermark.Image	宽	String	否	无	1. 当 Mode 为 Original 时，不支持设置水印图宽度 2. 当 Mode 为 Proportion，单位为%，背景图值范围：[100 300]；前景图值范围：[1 100]，相对于视频宽，最大不超过4096px 3. 当 Mode 为 Fixed，单位为 px，值范围：[8，4096] 4.若只设置 Width 时，按照水印图比例计算 Height

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Height	Request.Watermark.Image	高	String	否	无	1. 当 Mode 为 Original 时，不支持设置水印图高度 2. 当 Mode 为 Proportion，单位为%，背景图值范围：[100 300]；前景图值范围：[1 100]，相对于视频高，最大不超过4096px 3. 当 Mode 为 Fixed，单位为 px，值范围：[8，4096] 4.若只设置 Height 时，按照水印图比例计算 Width
Transparency	Request.Watermark.Image	透明度	String	是	无	值范围：[1 100]，单位%
Background	Request.Watermark.Image	是否背景图	String	否	false	true、false

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 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%
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

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] => NjJhOTRhNTlJAOISDUJIOASDUIOxNDc2N2I=
[ContentType] => application/xml
[ContentLength] => 853
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhOTRhNTlJAOISDUJIOASDUIOxNDc2N2I=
[Template] => Array
(
[TemplateId] => t183b6fd90369a908sd09a8sd09ase1340
[Name] => Watermark-Template-Name
[State] => Normal
[Tag] => Watermark
[CreateTime] => 2022-06-15T10:56:25+0800
[UpdateTime] => 2022-06-15T10:56:25+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[Watermark] => Array
(
[Type] => Text
[Pos] => TopRight
[LocMode] => Absolute
[Dx] => 128
[Dy] => 128
[StartTime] => 0
[EndTime] => 300
[Text] => Array
(
[FontType] => simfang
[FontSize] => 30
[FontColor] => 0x000000
[Text] => 水印内容
[Transparency] => 30
)

)

)
```

```
)

)
```

更新水印模板

功能说明

用于更新水印模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaWatermarkTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54036 更新水印模板
// 文本
$result = $cosClient->updateMediaWatermarkTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'Watermark',
'Name' => 'Watermark-Template-Name',
'Watermark' => array(
'Type' => 'Text',
'Pos' => 'TopRight',
'LocMode' => 'Absolute',
'Dx' => '128',
'Dy' => '128',
'StartTime' => '',
'EndTime' => '',
'Text' => array(
'FontSize' => '30',
'FontType' => 'simfang',
'FontColor' => '0x000000',
'Transparency' => '30',
'Text' => '水印内容',
),
),
```

```
));
// 请求成功
print_r($result);

// 图片
$result = $cosClient->updateMediaWatermarkTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'Watermark',
'Name' => 'Watermark-Template-Name2-1',
'Watermark' => array(
'Type' => 'Image',
'Pos' => 'TopRight',
'LocMode' => 'Absolute',
'Dx' => '128',
'Dy' => '128',
'StartTime' => '',
'EndTime' => '',
'Image' => array(
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.png',
'Mode' => 'Proportion',
'Width' => '10',
'Height' => '',
'Transparency' => '100',
'Background' => '',
),
),
));
// 请求成功
print_r($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	Request	模板类型：Watermark	String	是
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
Watermark	Request	水印信息	Container	是

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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Pos	Request.Watermark	基准位置	String	是	无	TopRight：右上 TopLeft：左上 BottomRight：右下 BottomLeft：左下
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	1. 在图片水印中，如果 Background 为 true，当 locMode 为 Relativity 时，单位为%，值范围：[-300 0]；当 locMode 为 Absolute 时，单位为 px，值范围：[-4096 0] 2. 在图片水印中，如果 Background 为 false，当 locMode 为 Relativity 时，单位为%，值范围：[0 100]；当 locMode 为 Absolute 时，单位为 px，值范围：[0 4096] 3. 在文字水印中，当 locMode 为 Relativity 时，单位为%，值范围：[0 100]；当 locMode 为 Absolute 时，单位为 px，值范围：[0 4096]
Dy	Request.Watermark	垂直偏移	String	是	无	1. 在图片水印中，如果 Background 为 true，当 locMode 为 Relativity 时，单位为%，值范围：[-300 0]；当 locMode 为 Absolute 时，单位为 px，值范围：[-4096 0] 2. 在图片水印中，如果 Background 为 false，当 locMode 为 Relativity 时，单位为%，值范围：[0 100]；当 locMode 为 Absolute 时，单位为 px，值范围：[0 4096] 3. 在文字水印中，当 locMode 为 Relativity 时，单位为%，值范围：[0 100]；当 locMode 为 Absolute 时，单位为 px，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	水印图片地址，如果水印图片为私有对象时，请携带签名信息
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 时，不支持设置水印图宽度 当 Mode 为 Proportion，单位为%，背景图值范围：[100 300]；前景图值范围：[1 100] 当 Mode 为 Fixed，单位为 px，值范围：[8，4096]，若只设置 Width 时，按照视频原始比例计算 Height

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Height	Request.Watermark.Image	高	String	否	无	当 Mode 为 Original 时，不支持设置水印图高度 当 Mode 为 Proportion，单位为%，背景图值范围：[100 300]；前景图值范围：[1 100] 当 Mode 为 Fixed，单位为px，值范围：[128，4096]，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：[0 100]，单位为%
Background	Request.Watermark.Image	是否背景图	String	否	false	true、false

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：[0 100]，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：[0 100]，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

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] => NjJhOTRiNzJfNzgwYzdkNAIODUAIODUNGZhMWQ=
[ContentType] => application/xml
[ContentLength] => 855
[Key] => t183b6fd9039167b2db0dfe1340
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t183b6fd9039167b2db0dfe1340
[Response] => Array
(
[RequestId] => NjJhOTRiNzJfNzgwYzdkNAIODUAIODUNGZhMWQ=
[Template] => Array
(
[TemplateId] => t183b6fd9039167b2db0dfe1340
[Name] => Watermark-Template-Name
[State] => Normal
[Tag] => Watermark
[CreateTime] => 2022-06-15T10:56:25+0800
[UpdateTime] => 2022-06-15T11:01:06+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[Watermark] => Array
(
[Type] => Text
[Pos] => TopRight
[LocMode] => Absolute
[Dx] => 128
[Dy] => 128
[StartTime] => 0
[EndTime] => 300
[Text] => Array
(
[FontType] => simfang
[FontSize] => 30
[FontColor] => 0x000000
[Text] => 水印内容
[Transparency] => 30
)

)

)

)
```

```
)
```

转码模板接口

最近更新时间：2022-07-25 08:55:23

简介

本文档提供关于转码模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增转码模板	用于新增转码模板。
更新转码模板	用于更新转码模板。

新增转码模板

功能说明

用于新增转码模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaTranscodeTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54037 新增转码模板
$result = $cosClient->createMediaTranscodeTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Transcode',
'Name' => 'Transcode-Template-Name',
'Container' => array(
'Format' => '',
'ClipConfig' => array(
'Duration' => '',
),
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'Remove' => '',
'Profile' => '',
'Bitrate' => '',
'Crf' => '',
```

```
'Gop' => '',
'Preset' => '',
'Bufsize' => '',
'Maxrate' => '',
'Pixfmt' => '',
'LongShortMode' => '',
'Rotate' => '',
),
'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' => '',
),
),
));
// 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
Tag	Request	模板类型：Transcode	String	是	无
Name	Request	模板名称，仅支持中文、英文、数字、_、-和*	String	是	无
Container	Request	容器格式	Container	是	无
Video	Request	视频信息	Container	否	不传 Video，相当于删除视频信息

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选	限制
TimeInterval	Request	时间区间	Container	否	无
Audio	Request	音频信息	Container	否	不传 Audio，相当于删除音频信息
TransConfig	Request	转码配置	Container	否	无

Container 类型 Container 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
Format	Request.Container	容器格式：mp4、flv、hls、ts、mp3、aac、WebM、dash	String	是
ClipConfig	Request.Container	分片配置，当 format 为 hls 和 dash 时有效	Container	否

Container 类型 ClipConfig 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
Duration	Request.Container.ClipConfig	分片时长，默认5s	String	否

设定 container，音频视频支持的格式如下表所示：

Container	Audio Codecs	Video Codecs
mp4/hls/mkv	aac、mp3	H.264、H.265
ts/flv	aac、mp3	H.264
aac	aac	不支持
mp3	mp3	不支持
flac	flac	不支持
amr	amr	不支持
WebM	Vorbis、opus	VP8、VP9
dash	aac	H.264

Container 类型 Video 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
Codec	Request.Video	编解码格式	String	否	H.264，当 format 为 WebM 时，为 VP8	H.264H.265VP8VP9
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 必须为偶数
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 必须为偶数
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Profile	Request.Video	编码级别	String	否	high	支持 baseline、main、high、auto 当 Pixfmt 为 auto 时，该参数仅能设置为 auto 当设置为其他选项时，参数值将被设置为 autobaseline：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 的设置失效当 Bitrate 为空时，默认为25
Gop	Request.Video	关键帧间最大帧数	String	否	无	值范围：[1, 100000]
Preset	Request.Video	视频算法器预置	String	否	medium，当 Codec 为 VP8 时，为 good	H.264 支持该参数，取值 veryfast、fast、medium、slow、slower VP8 支持该参数，取值 good、realtime H.265 和 VP9 不支持该参数
Bufsize	Request.Video	缓冲区大小	String	否	无	值范围：[1000, 128000] 单位：KbCodec 为 VP8/VP9时不支持此参数
Maxrate	Request.Video	视频码率峰值	String	否	无	值范围：[10, 50000] 单位：KbpsCodec 为 VP8/VP9时不支持此参数
Pixfmt	Request.Video	视频颜色格式	String	否	无	H.264支持：yuv420p、yuv422p、yuv444p、yuvj420p、yuvj422p、yuvj444p、autoH.265支持：yuv420p、yuv420p10le、autoCodec 为 VP8/VP9时不支持此参数
LongShortMode	Request.Video	长短边自适应	String	否	false	true、falseCodec 为 VP8/VP9 时不支持此参数
Rotate	Request.Video	旋转角度	String	否	无	值范围：[0, 360)单位：度

Container 类型 TimeInterval 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
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	视频原始时长	[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 Audio 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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Codec	Request.Audio	编解码格式	String	否	aac，当 format 为 WebM 时，为 Vorbis	取值 aac、mp3、flac、amr、Vorbis、opus
Samplerate	Request.Audio	采样率	String	否	44100，当 Codec 为 opus 时，默认值为48000	单位：Hz可选 8000、11025、12000、16000、22050、24000、32000、44100、48000、88200、96000 不同的封装，mp3 支持不同的采样率，如下表所示 当 Codec 设置为 amr 时，只支持8000 当 Codec 设置为 opus 时，仅支持8000，16000，24000，48000
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/flac，支持1、2、4、5、6、8当 Codec 设置为 mp3/opus 时，支持1、2 当 Codec 设置为 Vorbis 时，只支持2当 Codec 设置为 amr，只支持1当封装格式为dash时，不支持8
Remove	Request.Audio	是否删除源音频流	String	否	false	取值 true、false
KeepTwoTracks	Request.Audio	保持双音轨	String	否	false	取值 true、false。当 Video.Codec 为H.265 时，此参数无效。
SwitchTrack	Request.Audio	转换轨道	String	否	false	取值 true、false。当 Video.Codec 为H.265 时，此参数无效。
SampleFormat	Request.Audio	采样位宽	String	否	无	当 Codec 设置为 aac，支持 fltp 当 Codec 设置为 mp3，支持 fltp、s16p、s32p 当 Codec 设置为 flac 支持s16、s32 当 Codec 设置为 amr，支持s16 当 Video.Codec 为H.265时，此参数无效

说明：

Y 表示支持这种采样率，N 表示不支持。

Audio.Codec 采样率支持情况，如下表所示：

Audio.Codec	aac	amr	flac	opus	Vorbis	mp3
8000	Y	Y	Y	Y	Y	封装不同，支持不同
11025	Y	N	Y	N	Y	封装不同，支持不同
12000	Y	N	Y	N	Y	封装不同，支持不同
16000	Y	N	Y	Y	Y	封装不同，支持不同
22050	Y	N	Y	N	Y	封装不同，支持不同
24000	Y	N	Y	Y	Y	封装不同，支持不同
32000	Y	N	Y	N	Y	封装不同，支持不同
44100	Y	N	Y	N	Y	封装不同，支持不同
48000	Y	N	Y	Y	Y	封装不同，支持不同
88200	Y	N	Y	N	Y	封装不同，支持不同
96000	Y	N	Y	N	Y	封装不同，支持不同

版权所有：腾讯云计算（北京）有限责任公司

当 Audio.Codec 为 mp3 时，Container.Format 的采样率兼容情况如下所示：

封装格式/音频采样率	8000	11025	12000	16000	22050	24000	32000	44100	48000	88200	96000
flv	N	N	N	N	Y	N	N	Y	N	N	N
mp4	N	N	N	Y	Y	Y	Y	Y	Y	N	N
hls/ts/mp3/mkv	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N

Container 类型 TransConfig 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
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，根据此参数做分辨率的相应调整 当视频编解码格式为H265时，该参数无效
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	分辨率调整方式	String	否	0	取值0、1；0表示使用原视频分辨率；1表示返回转码失败当 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 当视频编解码格式为H265时，该参数无效
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 当视频编解码格式为H265时，该参数无效
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
DeleteMetadata	Request.TransConfig	是否删除文件中的 MetaData 信息	String	否	false	true、false 当为 false 时，保留源文件信息
IsHdr2Sdr	Request.TransConfig	是否开启 HDR 转 SDR	String	否	false	true/false
HlsEncrypt	Request.TransConfig	hls 加密配置	Container	否	无	无

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	是否必选	默认值	限制
IsHlsEncrypt	Request.TransConfig.HlsEncrypt	是否开启 HLS 加密	String	否	false	true/false当 Container.Format 为 hls 时支持加密
UriKey	Request.TransConfig.HlsEncrypt	HLS 加密的 key	String	否	无	当 IsHlsEncrypt 为 true 时，该参数才有意义

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] => NjJhODM3OTRfZmNjYTNiMCUOACNOISTY3ODQ=
[ContentType] => application/xml
[ContentLength] => 2021
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODM3OTRfZmNjYTNiMCUOACNOISTY3ODQ=
[Template] => Array
(
[TemplateId] => t102b2c576bdzx8c90z8c09z8c0203a4c61a
[Name] => Transcode-Template-Name
[State] => Normal
[Tag] => Transcode
[TransTpl] => Array
(
[Container] => Array
(
[Format] => hls
[ClipConfig] => Array
(
[Duration] => 5
)
)

[Video] => Array
(
[Codec] => H.264
[Profile] => high
[Bitrate] => 1000
[Crf] => 1
[Width] => 1280
[Height] => 720
[Fps] => 30
[Gop] => 1
[Preset] => medium
[Bufsize] => 1000
[Maxrate] => 30
[Remove] => false
```

```
[Pixfmt] => yuv420p
[LongShortMode] => true
[Rotate] => 0
)

[Audio] => Array
(
[Codec] => aac
[Profile] =>
[Samplerate] => 44100
[Bitrate] => 128
[Channels] => 4
[Remove] => false
[KeepTwoTracks] => false
[SwitchTrack] => false
[SampleFormat] => fltp
)

[TransConfig] => Array
(
[IsCheckReso] => false
[IsCheckVideoBitrate] => true
[IsCheckAudioBitrate] => false
[ResoAdjMethod] => 1
[VideoBitrateAdjMethod] => 0
[AudioBitrateAdjMethod] => 0
[AdjDarMethod] => scale
[IsStreamCopy] => false
[DeleteMetadata] => false
[IsHdr2Sdr] => false
[HlsEncrypt] => Array
(
[IsHlsEncrypt] => false
[UriKey] => https://www.xxx.com/xxx
)

)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 60
)

)

[CreateTime] => 2022-06-14T15:24:04+0800
[UpdateTime] => 2022-06-14T15:24:04+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
)

)

)
```

更新转码模板

功能说明

用于更新转码模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaTranscodeTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54040 更新转码模板
$result = $cosClient->updateMediaTranscodeTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'Transcode',
'Name' => 'Transcode-Template-Name',
'Container' => array(
'Format' => '',
'ClipConfig' => array(
'Duration' => '',
),
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'Remove' => '',
'Profile' => '',
'Bitrate' => '',
'Crf' => '',
'Gop' => '',
'Preset' => '',
'Bufsize' => '',
'Maxrate' => '',
'Pixfmt' => '',
'LongShortMode' => '',
'Rotate' => '',
),
```

版权所有：腾讯云计算（北京）有限责任公司

```
'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' => '',
),
),
));
// 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	限制
Tag	Request	模板类型：Transcode	String	是	无
Name	Request	模板名称 仅支持中文、英文、数字、_、-和*	String	是	无
Container	Request	容器格式	Container	是	无
Video	Request	视频信息	Container	否	不传 Video，相当于删除视频信息
TimeInterval	Request	时间区间	Container	否	无
Audio	Request	音频信息	Container	否	不传 Audio，相当于删除音频信息
TransConfig	Request	转码配置	Container	否	无

Container 类型 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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选
Format	Request.Container	容器格式：mp4、flv、hls、ts、mp3、aac、WebM、dash、mov、avi	String	是
ClipConfig	Request.Container	分片配置，当 format 为 hls 和 dash 时有效	Container	否

Container 类型 ClipConfig 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
Duration	Request.Container.ClipConfig	分片时长，默认5s	String	否

设定 container，音频视频支持的格式如下表所示：

Container	Audio Codecs	Video Codecs
mp4/hls/mkv	aac、mp3	H.264、H.265
ts/flv	aac、mp3	H.264
aac	aac	不支持
mp3	mp3	不支持
flac	flac	不支持
amr	amr	不支持
WebM	Vorbis、opus	VP8、VP9
dash	aac	H.264

Container 类型 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	否	H.264，当 format 为 WebM 时，为 VP8	1. H.264 2. H.265 3. VP8 4. VP9
Width	Request.Video	宽	String	否	视频原始宽度	1. 值范围：[128, 4096] 2. 单位：px 3. 若只设置 Width 时，按照视频原始比例计算 Height 4. 必须为偶数
Height	Request.Video	高	String	否	视频原始高度	1. 值范围：[128, 4096] 2. 单位：px 3. 若只设置 Height 时，按照视频原始比例计算 Width 4. 必须为偶数
Fps	Request.Video	帧率	String	否	无	1. 值范围：(0, 60] 2. 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	1. 支持 baseline、main、high、 auto 2. 当 Pixfmt 为 auto 时，该参数仅能设置为 auto，当设置为其他选项时，参数值将被设置为 auto 3. baseline：适合移动设备 4. main：适合标准分辨率设备 5. high：适合高分辨率设备 6. 仅H.264支持此参数

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Bitrate	Request.Video	视频输出文件的码率	String	否	无	1. 值范围：[10, 50000] 2. 单位：Kbps
Crf	Request.Video	码率-质量控制因子	String	否	无	1. 值范围：(0, 51] 2. 如果设置了 Crf，则 Bitrate 的设置失效 3. 当 Bitrate 为空时，默认为25
Gop	Request.Video	关键帧间最大帧数	String	否	无	值范围：[1, 100000]
Preset	Request.Video	视频算法器预置	String	否	medium，当 Codec 为 VP8 时，为 good	1. H.264支持该参数，取值 veryfast、fast、medium、slow、slower 2. VP8 支持该参数，取值 good、realtime 3. H.265和VP9不支持该参数
Bufsize	Request.Video	缓冲区大小	String	否	无	1. 值范围：[1000, 128000] 2. 单位：Kb 3. Codec 为 VP8/VP9时不支持此参数
Maxrate	Request.Video	视频码率峰值	String	否	无	1. 值范围：[10, 50000] 2. 单位：Kbps 3. Codec 为 VP8/VP9时不支持此参数
Pixfmt	Request.Video	视频颜色格式	String	否	无	1. H.264支持：yuv420p、yuv422p、yuv444p、yuvj420p、yuvj422p、yuvj444p、auto 2. H.265支持：yuv420p、yuv420p10le、auto 3. Codec 为 VP8/VP9时不支持此参数
LongShortMode	Request.Video	长短边自适应	String	否	false	1. true、false 2. Codec 为 VP8/VP9时不支持此参数
Rotate	Request.Video	旋转角度	String	否	无	1. 值范围：[0, 360) 2. 单位：度

Container 类型 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 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，当 format 为 WebM 时，为 Vorbis	取值 aac、mp3、flac、amr、Vorbis、opus
Samplerate	Request.Audio	采样率	String	否	44100，当 Codec 为 opus 时，默认值为 48000	单位：Hz，可选 8000、11025、12000、16000、22050、24000、32000、44100、48000、88200、96000 不同的封装，mp3 支持不同的采样率，如下表所示 当 Codec 设置为 amr 时，只支持8000 当 Codec 设置为 opus 时，仅支持8000，16000，24000，48000
Bitrate	Request.Audio	原始音频码率	String	否	无	单位：Kbps 值范围：[8，1000]

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Channels	Request.Audio	声道数	String	否	无	当 Codec 设置为 aac/flac，支持1、2、4、5、6、8 当 Codec 设置为 mp3/opus 时，支持1、2 当 Codec 设置为 Vorbis 时，只支持2当 Codec 设置为 amr，只支持1当封装格式为dash时，不支持8
Remove	Request.Audio	是否删除源音频流	String	否	false	取值 true、false
KeepTwoTracks	Request.Audio	保持双音轨	String	否	false	取值 true、false。当 Video.Codec 为H.265时，此参数无效
SwitchTrack	Request.Audio	转换轨道	String	否	false	取值 true、false。当 Video.Codec 为H.265时，此参数无效
SampleFormat	Request.Audio	采样位宽	String	否	无	当 Codec 设置为 aac, 支持 fltp 当 Codec 设置为 mp3, 支持 fltp、s16p、s32p 当 Codec 设置为 flac, 支持s16、s32当 Codec 设置为 amr, 支持s16 当 Video.Codec 为H.265时，此参数无效

说明：

Y 表示支持这种采样率，N 表示不支持。

Audio.Codec 采样率支持情况，如下表所示：

Audio.Codec	aac	amr	flac	opus	Vorbis	mp3
8000	Y	Y	Y	Y	Y	封装不同，支持不同
11025	Y	N	Y	N	Y	封装不同，支持不同
12000	Y	N	Y	N	Y	封装不同，支持不同
16000	Y	N	Y	Y	Y	封装不同，支持不同
22050	Y	N	Y	N	Y	封装不同，支持不同
24000	Y	N	Y	Y	Y	封装不同，支持不同
32000	Y	N	Y	N	Y	封装不同，支持不同
44100	Y	N	Y	N	Y	封装不同，支持不同
48000	Y	N	Y	Y	Y	封装不同，支持不同
88200	Y	N	Y	N	Y	封装不同，支持不同
96000	Y	N	Y	N	Y	封装不同，支持不同

当 Audio.Codec 为 mp3 时，Container.Format 的采样率兼容情况如下所示：

封装格式/音频采样率	8000	11025	12000	16000	22050	24000	32000	44100	48000	88200	96000
flv	N	N	N	N	Y	N	N	Y	N	N	N
mp4	N	N	N	Y	Y	Y	Y	Y	Y	N	N
hls/ts/mp3/mkv	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N

Container 类型 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

版权所有：腾讯云计算（北京）有限责任公司

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，根据此参数做分辨率的相应调整
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	分辨率调整方式	String	否	0	取值0、1；0表示使用原视频分辨率；1表示返回转码失败 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 当视频编解码格式为H265时，该参数无效
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 当视频编解码格式为H265时，该参数无效
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
DeleteMetadata	Request.TransConfig	是否删除文件中的 MetaData 信息	String	否	false	true、false 当为 false 时，保留源文件信息
IsHdr2Sdr	Request.TransConfig	是否开启 HDR 转 SDR	String	否	false	true、false
HlsEncrypt	Request.TransConfig	hls 加密配置	Container	否	无	无

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.TransConfig.HlsEncrypt	是否开启 HLS加密	String	否	false	true、false 当Video.Codec 为H264或H265，且 Container.Format 为 hls 时支持加密
UriKey	Request.TransConfig.HlsEncrypt	HLS加密的 key	String	否	无	当 IsHlsEncrypt 为 true 时，该参数才有意义

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
```

版权所有：腾讯云计算（北京）有限责任公司

```
[customMetadata:GuzzleHttp\Psr7\Stream:private] => Array
(
)

)

[RequestId] => NjJhODNhNDRfNzgwYzdkNjZXIUClOZClOxMWFjNTg=
[ContentType] => application/xml
[ContentLength] => 2023
[Key] => t102b2c576bd09e61as98d90a8d90a4c61a
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t102b2c576bdf144d09e61402203a4c61a
[Response] => Array
(
[RequestId] => NjJhODNhNDRfNzgwYzdkNjZXIUClOZClOxMWFjNTg=
[Template] => Array
(
[TemplateId] => t102b2c576bd09e61as98d90a8d90a4c61a
[Name] => Transcode-Template-Name
[State] => Normal
[Tag] => Transcode
[TransTpl] => Array
(
[Container] => Array
(
[Format] => hls
[ClipConfig] => Array
(
[Duration] => 5
)

)

[Video] => Array
(
[Codec] => H.264
[Profile] => high
[Bitrate] => 1000
[Crf] => 1
[Width] => 1280
[Height] => 720
[Fps] => 30
[Gop] => 1
[Preset] => medium
[Bufsize] => 1000
[Maxrate] => 30
[Remove] => false
[Pixfmt] => yuv420p
[LongShortMode] => true
[Rotate] => 0
)

[Audio] => Array
(
[Codec] => aac
[Profile] =>
```

```
[Samplerate] => 44100
[Bitrate] => 128
[Channels] => 4
[Remove] => false
[KeepTwoTracks] => false
[SwitchTrack] => false
[SampleFormat] => fltp
)

[TransConfig] => Array
(
[IsCheckReso] => false
[IsCheckVideoBitrate] => true
[IsCheckAudioBitrate] => false
[ResoAdjMethod] => 1
[VideoBitrateAdjMethod] => 0
[AudioBitrateAdjMethod] => 0
[AdjDarMethod] => scale
[IsStreamCopy] => false
[DeleteMetadata] => false
[IsHdr2Sdr] => false
[HlsEncrypt] => Array
(
[IsHlsEncrypt] => false
[UriKey] => https://www.xxx.com/xxx
)

)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 60
)

)

[CreateTime] => 2022-06-14T15:24:04+0800
[UpdateTime] => 2022-06-14T15:35:32+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
)

)

)
```

拼接模板接口

最近更新时间：2022-07-25 08:55:29

简介

本文档提供关于拼接模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增拼接模板	用于新增拼接模板。
更新拼接模板	用于更新拼接模板。

新增拼接模板

功能说明

用于新增拼接模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaConcatTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54041 新增拼接模板
$result = $cosClient->createMediaConcatTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'Concat',
'Name' => 'Concat-Template-Name',
'ConcatTemplate' => array(
'ConcatFragments' => array(
array(
'Mode' => 'Start',
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video01.mp4',
),
array(
'Mode' => 'End',
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video02.mp4',
),
),
```

```
'Audio' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
'Video' => array(
'Codec' => 'h.264',
'Width' => '',
'Height' => '',
'Fps' => '',
'Bitrate' => '',
'Remove' => 'false',
),
'Container' => array(
'Format' => 'mp4',
),
),
));
// 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	Request	模板类型：Concat	String	是
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
ConcatTemplate	Request	拼接模板	Container	是

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	必选	默认值	限制
ConcatFragment	Request.ConcatTemplate	拼接节点	Container	是	无	无
Audio	Request.ConcatTemplate	音频参数	Container	否	媒体原始值	目标文件不需要 Audio 信息，需要设置 Audio.Remove 为 true
Video	Request.ConcatTemplate	视频参数	Container	否	媒体原始值	目标文件不需要 Video 信息，需要设置 Video.Remove 为 true
Container	Request.ConcatTemplate	封装格式	Container	是	无	无

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	必选	默认值	限制
Url	Request.ConcatTemplate. ConcatFragment	拼接对象地址	String	是	无	同 bucket 对象文件
Mode	Request.ConcatTemplate. ConcatFragment	节点类型	String	是	无	Start：开头 End：结尾

版权所有：腾讯云计算（北京）有限责任公司

Container 类型 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.ConcatTemplate.Audio	编解码格式	String	是	文件原编码	取值 aac、mp3
Samplerate	Request.ConcatTemplate.Audio	采样率	String	否	文件原采样率	单位：Hz 可选 11025、22050、32000、44100、48000、96000 不同的封装，mp3 支持不同的采样率，如下表所示
Bitrate	Request.ConcatTemplate.Audio	音频码率	String	否	文件原音频码率	单位：Kbps 值范围：[8，1000]
Channels	Request.ConcatTemplate.Audio	声道数	String	否	文件原声道数	当 Codec 设置为 aac，支持1、2、4、5、6、8 当 Codec 设置为 mp3，支持1、2

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
mp3	Y	Y	Y	Y	Y	N

Container 类型 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.ConcatTemplate. Container	容器格式：mp4，flv，hls，ts, mp3, aac	String	是

Container 类型 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.ConcatTemplate.Video	编解码格式	String	是	H.264	H.264
Width	Request.ConcatTemplate.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.ConcatTemplate.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.ConcatTemplate.Video	帧率	String	否	视频原始帧率	值范围：(0，60] 单位：fps
Bitrate	Request.ConcatTemplate.Video	视频输出文件的码率	String	否	视频原始码率	值范围：[10，50000] 单位：Kbps
Remove	Request.ConcatTemplate.Video	是否删除视频流	String	否	false	取值 true、false

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] => NjJhODRmNWJfNzgwYzdAJIOSJDOIJSDOxMTg1MGI=
[ContentType] => application/xml
[ContentLength] => 1434
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODRmNWJfNzgwYzdAJIOSJDOIJSDOxMTg1MGI=
[Template] => Array
(
[TemplateId] => t1dc71b980da09sd80s9aba4b159
[Name] => Concat-Template-Name
[State] => Normal
[Tag] => Concat
[CreateTime] => 2022-06-14T17:05:32+0800
[UpdateTime] => 2022-06-14T17:05:32+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[ConcatTemplate] => Array
(
[ConcatFragment] => Array
(
[0] => Array
(
[Mode] => Start
[Url] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video01.mp4

)

[1] => Array
(
[Mode] => End
[Url] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video02.mp4
)

)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 88
[Channels] => 2
[Remove] => false
)
```

```
[Video] => Array
(
[Codec] => H.264
[Width] => 400
[Height] => 400
[Fps] => 30
[Bitrate] => 1000
[Remove] => false
)

[Container] => Array
(
[Format] => mp4
)

[DirectConcat] => false
)

)

)

)
```

更新拼接模板

功能说明

用于更新拼接模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaConcatTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/54044 更新拼接模板
$result = $cosClient->updateMediaConcatTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
```

```
'Key' => '', // TemplateId
'Tag' => 'Concat',
'Name' => 'Concat-Template-Name',
'ConcatTemplate' => array(
'ConcatFragments' => array(
array(
'Mode' => 'Start',
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video01.mp4',
),
array(
'Mode' => 'End',
'Url' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video02.mp4',
),
),
'Audio' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
'Video' => array(
'Codec' => 'h.264',
'Width' => '',
'Height' => '',
'Fps' => '',
'Bitrate' => '',
'Remove' => 'false',
),
'Container' => array(
'Format' => 'mp4',
),
),
));
// 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	Request	模板类型：Concat	String	是
Name	Request	模板名称仅支持中文、英文、数字、_、-和*	String	是
ConcatTemplate	Request	拼接模板	Container	是

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	必选	默认值	限制
ConcatFragment	Request.ConcatTemplate	拼接节点	Container	是	无	无

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Audio	Request.ConcatTemplate	音频参数	Container	否	无	目标文件不需要 Audio 信息，需要设置 Audio.Remove 为 true
Video	Request.ConcatTemplate	视频参数	Container	否	无	目标文件不需要 Video 信息，需要设置 Video.Remove 为 true
Container	Request.ConcatTemplate	封装格式	Container	是	无	无

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	必选	默认值	限制
Url	Request.ConcatTemplate. ConcatFragment	拼接对象地址	String	是	无	同 bucket 对象文件
Mode	Request.ConcatTemplate. ConcatFragment	节点类型	String	是	无	Start：开头 End：结尾

Container 类型 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.ConcatTemplate. Audio	编解码格式	String	否	文件原编码	取值 aac、mp3
Samplerate	Request.ConcatTemplate. Audio	采样率	String	否	文件原采样率	单位：Hz 可选 11025、22050、32000、44100、48000、96000 不同的封装，mp3 支持不同的采样率，如下表所示
Bitrate	Request.ConcatTemplate. Audio	音频码率	String	否	文件原音频码率	单位：Kbps 值范围：[8，1000]
Channels	Request.ConcatTemplate. Audio	声道数	String	否	文件原声道数	当 Codec 设置为 aac，支持1、2、4、5、6、8 当 Codec 设置为 mp3，支持1、2

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
mp3	Y	Y	Y	Y	Y	N

Container 类型 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.ConcatTemplate. Container	容器格式：mp4，flv，hls，ts, mp3, aac	String	是

Container 类型 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.ConcatTemplate. Video	编解码格式	String	否	H.264	H.264
Width	Request.ConcatTemplate. 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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
Height	Request.ConcatTemplate.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.ConcatTemplate.Video	帧率	String	否	视频原始帧率	值范围：(0，60] 单位：fps
Bitrate	Request.ConcatTemplate.Video	视频输出文件的码率	String	否	视频原始码率	值范围：[10，50000] 单位：Kbps
Remove	Request.ConcatTemplate.Video	是否删除视频流	String	否	false	取值 true、false

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] => NjJhODUxMTdfZmNjYTNiMAIOSDJIOSDTY5M2U=
[ContentType] => application/xml
[ContentLength] => 1436
[Key] => t1dc71b2bc221as8d90as8d0959
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t1dc71b2bc221as8d90as8d0959
[Response] => Array
(
[RequestId] => NjJhODUxMTdfZmNjYTNiMAIOSDJIOSDTY5M2U=
[Template] => Array
(
[TemplateId] => t1dc71b2bc221as8d90as8d0959
[Name] => Concat-Template-Name
[State] => Normal
[Tag] => Concat
[CreateTime] => 2022-06-14T17:05:32+0800
[UpdateTime] => 2022-06-14T17:12:55+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[ConcatTemplate] => Array
(
[ConcatFragment] => Array
(
```

```
[0] => Array
(
[Mode] => Start
[Url] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video01.mp4
)

[1] => Array
(
[Mode] => End
[Url] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/video02.mp4
)

)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 88
[Channels] => 2
[Remove] => false
)

[Video] => Array
(
[Codec] => H.264
[Width] => 400
[Height] => 400
[Fps] => 30
[Bitrate] => 1000
[Remove] => false
)

[Container] => Array
(
[Format] => mp4
)

[DirectConcat] => false
)

)

)

)
```

极速高清转码模板接口

最近更新时间：2022-07-25 08:55:33

简介

本文档提供关于极速高清转码模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增极速高清转码模板	用于新增极速高清转码模板。
更新极速高清转码模板	用于更新极速高清转码模板。

新增极速高清转码模板

功能说明

用于新增极速高清转码模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaHighSpeedHdTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58307 新增极速高清转码模板
$result = $cosClient->createMediaHighSpeedHdTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'HighSpeedHd',
'Name' => 'HighSpeedHd-Template-Name',
'Container' => array(
'Format' => '',
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'Profile' => '',
'Bitrate' => '',
'Crf' => '',
```

```
'Gop' => '',
'Preset' => '',
'Bufsize' => '',
'Maxrate' => '',
'HlsTsTime' => '',
'Pixfmt' => '',
),
'TimeInterval' => array(
'Start' => '',
'Duration' => '',
),
'Audio' => array(
'Codec' => '',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
'TransConfig' => array(
'IsCheckReso' => '',
'ResoAdjMethod' => '',
'IsHdr2Sdr' => '',
),
));
// 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
Tag	Request	模板类型：HighSpeedHd	String	是	无
Name	Request	模板名称。仅支持中文、英文、数字、_、-和*	String	是	无
Container	Request	容器格式	Container	是	无
Video	Request	视频信息	Container	是	无
TimeInterval	Request	时间区间	Container	否	无
Audio	Request	音频信息	Container	是	无
TransConfig	Request	转码配置	Container	否	无

Container 类型 Container 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
Format	Request.Container	容器格式：mp4，flv，hls	String	是

设定 container，音频视频支持的格式如下表：

Container	Audio Codecs	Video Codecs
mp4/hls	AAC、MP3	H.264、H.265

版权所有：腾讯云计算（北京）有限责任公司

Container	Audio Codecs	Video Codecs
flv	AAC、MP3	H.264

Container 类型 Video 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
Codec	Request.Video	编解码格式	String	否	H.264	H.264、H.265
Width	Request.Video	宽	String	否	视频原 始宽度	1. 值范围：[128，4096] 2. 单位：px 3. 若只设置 Width 时，按照 视频原始比例计算 Height 4. 必须为偶数
Height	Request.Video	高	String	否	视频原 始高度	1. 值范围：[128，4096] 2. 单位：px 3. 若只设置 Height 时，按照 视频原始比例计算 Width 4. 必须为偶数
Fps	Request.Video	帧率	String	否	无	1. 值范围：(0，60] 2. 单位：fps
Profile	Request.Video	编码级别	String	否	high	1. 支持 baseline、main、high 2. baseline：适合移动设备 3. main：适合标准分辨率设备 4. high：适合高分辨率设备 5. 仅H.264支持此参数
Bitrate	Request.Video	视频输出文件的码率	String	否	无	1. 值范围：[10，50000] 2. 单位：Kbps
Crf	Request.Video	码率-质量控制因子	String	否	无	1. 值范围：(0，51] 2. 如果设置了 Crf，则 Bitrate 的设置失效 3. 当 Bitrate 为空时，默认为25
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	1. 仅H.264支持该参数 2. 取值 veryfast、fast、medium、slow、slower
Bufsize	Request.Video	缓冲区大小	String	否	无	1. 值范围：[1000，128000] 2. 单位：Kb
Maxrate	Request.Video	视频码率峰值	String	否	无	1. 值范围：[10，50000] 2. 单位：Kbps
HlsTsTime	Request.Video	hls 分片时间	String	否	5	1. (0 视频时长] 2. 单位为秒
Pixfmt	Request.Video	视频颜色格式	String	否	无	yuv420p

Container 类型 TimeInterval 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
Start	Request.TimeInterval	开始时间	String	否	无	1. [0 视频时长] 2. 单位为秒 3. 支持 float 格式，执行精度精确到毫秒
Duration	Request.TimeInterval	持续时间	String	否	无	1. [0 视频时长] 2. 单位为秒 3. 支持 float 格式，执行精度精确到毫秒

版权所有：腾讯云计算（北京）有限责任公司

Container 类型 Audio 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
Codec	Request.Audio	编解码格式	String	否	aac	取值 aac、mp3
Samplerate	Request.Audio	采样率	String	否	44100	1. 单位：Hz 2. 可选 11025、22050、32000、44100、48000、96000 3. 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	1. 单位：Kbps 2. 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 2. 当 Codec 设置为 mp3，支持1、2

说明：
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
hls	Y	Y	Y	Y	Y	N

Container 类型 TransConfig 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
IsCheckReso	Request.TransConfig	是否检查分辨率	String	否	false	1. true、false 2. 当为 false 时，按照配置参数转码
ResoAdjMethod	Request.TransConfig	分辨率调整方式	String	否	0	取值0、1；0 表示使用原视频分辨率；1表示返回转码失败 2. 当 IsCheckReso 为 true 时生效
IsHdr2Sdr	Request.TransConfig	是否开启 HDR 转 SDR	String	否	false	true/false

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] => NjJhODQ2ZDJfNzgwYzdAJSIODJIAODJhMzk=
[ContentType] => application/xml
[ContentLength] => 1990
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODQ2ZDJfNzgwYzdAJSIODJIAODJhMzk=
[Template] => Array
(
[TemplateId] => t1a1c25825ded1zx8c09zx8c04007ac
[Name] => HighSpeedHd-Template-Name
[State] => Normal
[Tag] => HighSpeedHd
[CreateTime] => 2022-06-14T16:29:06+0800
[UpdateTime] => 2022-06-14T16:29:06+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[HighSpeedHd] => Array
(
[Container] => Array
(
[Format] => hls
[ClipConfig] => Array
(
[Duration] => 30
)
)

[Video] => Array
(
[Codec] => H.264
[Profile] => main
[Bitrate] => 300
[Crf] => 25
[Width] => 400
[Height] => 400
[Fps] => 30
[Gop] => 300
[Preset] => veryfast
[Bufsize] => 1000
[Maxrate] => 300
[Remove] => false
[HlsTsTime] => 30
[Pixfmt] => yuv420p
[LongShortMode] => false
)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 88
[Channels] => 2
[Remove] => false
[KeepTwoTracks] => false
```

```
[SwitchTrack] => false
)

[TransConfig] => Array
(
[IsCheckReso] => true
[IsCheckVideoBitrate] => false
[IsCheckAudioBitrate] => false
[ResoAdjMethod] => 0
[VideoBitrateAdjMethod] => 0
[AudioBitrateAdjMethod] => 0
[AdjDarMethod] => none
[IsStreamCopy] => false
[DeleteMetadata] => false
[IsHdr2Sdr] => true
)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 30
)

)

)

)

)
```

更新极速高清转码模板

功能说明

用于更新极速高清转码模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaHighSpeedHdTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58310 更新极速高清转码模板
$result = $cosClient->updateMediaHighSpeedHdTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'HighSpeedHd',
'Name' => 'HighSpeedHd-Template-Name',
'Container' => array(
'Format' => '',
),
'Video' => array(
'Codec' => '',
'Width' => '',
'Height' => '',
'Fps' => '',
'Profile' => '',
'Bitrate' => '',
'Crf' => '',
'Gop' => '',
'Preset' => '',
'Bufsize' => '',
'Maxrate' => '',
'HlsTsTime' => '',
'Pixfmt' => '',
),
'TimeInterval' => array(
'Start' => '',
'Duration' => '',
),
'Audio' => array(
'Codec' => '',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
'TransConfig' => array(
'IsCheckReso' => '',
'ResoAdjMethod' => '',
'IsHdr2Sdr' => '',
),
));
// 请求成功
print_r($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
Tag	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.Tag

节点名称（关键字）	父节点	描述
Name	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.Name
TimeInterval	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.TimeInterval
Container	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.Container
Video	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.Video
Audio	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.Audio
TransConfig	Request	同极速高清模板 CreateMediaTemplate 接口中的 Request.TransConfig

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] => NjJhODQ4MTBfZmNjYTNiAJHIOSDJAOIDxMTliZjM=
[ContentType] => application/xml
[ContentLength] => 1992
[Key] => t1a1c25825dedb4as8d90a8d09007ac
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t1a1c25825dedb4as8d90a8d09007ac
[Response] => Array
(
[RequestId] => NjJhODQ4MTBfZmNjYTNiAJHIOSDJAOIDxMTliZjM=
[Template] => Array
(
[TemplateId] => t1a1c25825dedb4as8d90a8d09007ac
[Name] => HighSpeedHd-Template-Name
[State] => Normal
[Tag] => HighSpeedHd
[CreateTime] => 2022-06-14T16:29:06+0800
[UpdateTime] => 2022-06-14T16:34:24+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[HighSpeedHd] => Array
(
[Container] => Array
(
[Format] => hls
[ClipConfig] => Array
(
```

```
[Duration] => 30
)

)

[Video] => Array
(
[Codec] => H.264
[Profile] => high
[Bitrate] => 300
[Crf] => 25
[Width] => 400
[Height] => 400
[Fps] => 30
[Gop] => 300
[Preset] => veryfast
[Bufsize] => 1000
[Maxrate] => 300
[Remove] => false
[HlsTsTime] => 30
[Pixfmt] => yuv420p
[LongShortMode] => false
[Rotate] =>
)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 88
[Channels] => 2
[Remove] => false
[KeepTwoTracks] => false
[SwitchTrack] => false
)

[TransConfig] => Array
(
[IsCheckReso] => true
[IsCheckVideoBitrate] => false
[IsCheckAudioBitrate] => false
[ResoAdjMethod] => 0
[VideoBitrateAdjMethod] => 0
[AudioBitrateAdjMethod] => 0
[AdjDarMethod] => none
[IsStreamCopy] => false
[DeleteMetadata] => false
[IsHdr2Sdr] => true
)

[TimeInterval] => Array
(
[Start] => 0
[Duration] => 30
)
```

```
)
)
)
)
```

精彩集锦模板接口

最近更新时间：2022-07-25 08:55:38

简介

本文档提供关于精彩集锦模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增精彩集锦模板	用于新增精彩集锦模板。
更新精彩集锦模板	用于更新精彩集锦模板。

新增精彩集锦模板

功能说明

用于新增精彩集锦模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaVideoMontageTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58314 新增精彩集锦模板
$result = $cosClient->createMediaVideoMontageTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoMontage',
'Name' => 'VideoMontage-Template-Name',
'Duration' => '',
'Container' => array(
'Format' => 'mp4',
),
'Video' => array(
'Codec' => 'H.264',
'Width' => '',
'Height' => '',
'Fps' => '',
'Bitrate' => '',
'Crf' => '',
```

```
),
'Audio' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
'Remove' => '',
),
'AudioMix' => array(
'AudioSource' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.mp3',
'MixMode' => 'Once',
'Replace' => 'true',
),
));
// 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
Tag	Request	模板类型: VideoMontage	String	是	无
Name	Request	模板名称 仅支持中文、英文、数字、_、-和*	String	是	无
Duration	Request	集锦时长	String	否	1. 默认自动分析时长 2. 单位为秒 3. 支持 float 格式，执行精度精确到毫秒
Container	Request	容器格式	Container	是	无
Video	Request	视频信息	Container	是	无
Audio	Request	音频信息	Container	否	无
AudioMix	Request	混音参数	Container	否	仅在 Audio.Remove 为 false 时生效

Container 类型 Container 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
Format	Request.Container	容器格式: mp4、flv、hls、ts、mkv	String	是

设定 container，音频视频支持的格式如下表：

Container	Audio Codecs	Video Codecs
mp4/ts/hls/mkv	AAC、MP3	H.264、H.265
flv	AAC、MP3	H.264

Container 类型 Video 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

节点名称（关键字）	父节点	描述	类型	是否必选	默认值	限制
Codec	Request.Video	编解码格式	String	否	H.264	1. H.264 2. H.265
Width	Request.Video	宽	String	否	视频原始宽度	1. 值范围：[128，4096] 2. 单位：px 3. 若只设置 Width 时，按照视频原始比例计算 Height 4. 必须为偶数
Height	Request.Video	高	String	否	视频原始高度	1. 值范围：[128，4096] 2. 单位：px 3. 若只设置 Height 时，按照视频原始比例计算 Width 4. 必须为偶数
Fps	Request.Video	帧率	String	否	无	1. 值范围：(0，60] 2. 单位：fps
Bitrate	Request.Video	视频输出文件的码率	String	否	无	1. 值范围：[10，50000] 2. 单位：Kbps
Crf	Request.Video	码率-质量控制因子	String	否	无	1. 值范围：(0, 51] 2. 如果设置了 Crf，则 Bitrate 的设置失效 3. 当 Bitrate 为空时，默认为25

Container 类型 Audio 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
Codec	Request.Audio	编解码格式	String	否	aac	取值 aac、mp3
Samplerate	Request.Audio	采样率	String	否	44100	1. 单位：Hz 2. 可选 11025、22050、32000、44100、48000、96000 3. 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	1. 单位：Kbps 2. 值范围：[8，1000]
Channels	Request.Audio	声道数	String	否	无	1. 当 Codec 设置为 aac，支持1、2、4、5、6、8 2. 当 Codec 设置为 mp 3. 支持1、2
Remove	Request.Audio	是否删除音频流	String	否	false	取值 true、false

说明：
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	N	Y	N	Y	N	N
mp4	N	Y	Y	Y	Y	N
hls/ts/mkv	Y	Y	Y	Y	Y	N

Container 类型 AudioMix 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

节点名称（关键字）	父节点	描述	类型	必选	默认值	限制
AudioSource	Request.AudioMix	混音文件地址	String	是	无	需与 Input 媒体文件存储于同一 bucket
MixMode	Request.AudioMix	混音模式	String	否	Repeat	1. Repeat：背景音循环 2. Once：背景音一次播放
Replace	Request.AudioMix	是否用混音音轨媒体替换 Input 媒体文件的原音频	String	否	false	true/false

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] => NjJhODZkNGRfZmNAOISUDOIASUDIOMjI0MzU=
[ContentType] => application/xml
[ContentLength] => 1115
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODZkNGRfZmNAOISUDOIASUDIOMjI0MzU=
[Template] => Array
(
[TemplateId] => t1107eb6ca4da547f3as8d90a8d067938
[Name] => VideoMontage-Template-Name
[State] => Normal
[Tag] => VideoMontage
[CreateTime] => 2022-06-14T19:13:17+0800
[UpdateTime] => 2022-06-14T19:13:17+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VideoMontage] => Array
(
[Duration] => 10
[Container] => Array
(
[Format] => mp4
)

[Video] => Array
```

```
(
[Codec] => H.264
[Bitrate] => 100
[Width] => 400
[Height] => 400
[Fps] => 60
[Crf] => 30
)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 100
[Channels] => 2
[Remove] => false
)

[AudioMix] => Array
(
[AudioSource] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.mp3
[MixMode] => Once
[Replace] => true
)

)

)

)

)
```

更新精彩集锦模板

功能说明

用于更新精彩集锦模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaVideoMontageTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58311 更新精彩集锦模板
$result = $cosClient->updateMediaVideoMontageTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'VideoMontage',
'Name' => 'VideoMontage-Template-Name',
'Duration' => '',
'Container' => array(
'Format' => 'mp4',
),
'Video' => array(
'Codec' => 'H.264',
'Width' => '',
'Height' => '',
'Fps' => '',
'Bitrate' => '',
'Crf' => '',
),
'Audio' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
'Remove' => '',
),
'AudioMix' => array(
'AudioSource' => 'https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.mp3',
'MixMode' => 'Once',
'Replace' => 'true',
),
));
// 请求成功
print_r($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
Tag	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Tag
Name	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Name
Container	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Container
Duration	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Duration
Video	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Video

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述
Audio	Request	同精彩集锦模板 CreateMediaTemplate 接口中的 Request.Audio

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] => NjJhODZlYTlfZmNjYTNiMGAUJSDOISDUoTQ=
[ContentType] => application/xml
[ContentLength] => 1116
[Key] => t1107eb6ca4f3aas8d0as9d80938
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t1107eb6ca4f3aas8d0as9d80938
[Response] => Array
(
[RequestId] => NjJhODZlYTlfZmNjYTNiMGAUJSDOISDUoTQ=
[Template] => Array
(
[TemplateId] => t1107eb6ca4f3aas8d0as9d80938
[Name] => VideoMontage-Template-Name-ss
[State] => Normal
[Tag] => VideoMontage
[CreateTime] => 2022-06-14T19:13:17+0800
[UpdateTime] => 2022-06-14T19:19:05+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VideoMontage] => Array
(
[Duration] => 10
[Container] => Array
(
[Format] => mp4
)

[Video] => Array
(
[Codec] => H.264
[Bitrate] => 100
[Width] => 400
[Height] => 400
[Fps] => 60
```

```
[Crf] => 30
)

[Audio] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 100
[Channels] => 2
[Remove] => false
)

[AudioMix] => Array
(
[AudioSource] => https://examplebucket-125000000.cos.ap-guangzhou.myqcloud.com/test01.mp3
[MixMode] => Once
[Replace] => true
)

)

)

)

)
```

人声分离模板接口

最近更新时间：2022-07-25 08:55:44

简介

本文档提供关于人声分离模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增人声分离模板	用于新增人声分离模板。
更新人声分离模板	用于更新人声分离模板。

新增人声分离模板

功能说明

用于新增人声分离模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaVoiceSeparateTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58315 新增人声分离模板
$result = $cosClient->createMediaVoiceSeparateTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VoiceSeparate',
'Name' => 'VoiceSeparate-Template-Name',
'AudioMode' => 'IsAudio',
'AudioConfig' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
));
// 请求成功
print_r($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
Tag	Request	模板类型: VoiceSeparate	String	是	无
Name	Request	模板名称 仅支持中文、英文、数字、_、-和*	String	是	无
AudioMode	Request	输出音频	String	是	IsAudio：输出人声 IsBackground：输出背景声 AudioAndBackground：输出人声和背景声
AudioConfig	Request	音频配置	Container	是	无

Container 类型 AudioConfig 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
Codec	Request.Audio	编解码格式	String	否	aac	取值 aac、mp3、flac、amr
Samplerate	Request.Audio	采样率	String	否	44100	1. 单位：Hz 2. 可选 8000、11025、22050、32000、44100、48000、96000 3. 当 Codec 设置为 aac/flac 时，不支持8000 4. 当 Codec 设置为 mp3 时，不支持8000和96000 5. 当 Codec 设置为 amr 时，只支持8000
Bitrate	Request.Audio	原始音频码率	String	否	无	1. 单位：Kbps 2. 值范围：[8，1000]
Channels	Request.Audio	声道数	String	否	无	当 Codec 设置为 aac/flac，支持1、2、4、5、6、8 2. 当 Codec 设置为 mp3，支持1、2 3. 当 Codec 设置为 amr，只支持1

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] => NjJhODcyMGVfNzgwYzAOIJDIOUJDIODJY=
```

```
[ContentType] => application/xml
[ContentLength] => 705
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODcyMGVfNzgwYzAOIJDIOUJDIODJY=
[Template] => Array
(
[TemplateId] => t13eedc8532b0644cas8d09as8d0338f
[Name] => VoiceSeparate-Template-Name
[State] => Normal
[Tag] => VoiceSeparate
[CreateTime] => 2022-06-14T19:33:34+0800
[UpdateTime] => 2022-06-14T19:33:34+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VoiceSeparate] => Array
(
[AudioMode] => IsAudio
[AudioConfig] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 300
[Channels] => 2
)

)

)

)

)
```

更新人声分离模板

功能说明

用于更新人声分离模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaVoiceSeparateTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/58318 更新人声分离模板
$result = $cosClient->updateMediaVoiceSeparateTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'VoiceSeparate',
'Name' => 'VoiceSeparate-Template-Name',
'AudioMode' => 'IsAudio',
'AudioConfig' => array(
'Codec' => 'aac',
'Samplerate' => '',
'Bitrate' => '',
'Channels' => '',
),
));
// 请求成功
print_r($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
Tag	Request	同人声分离模板 CreateMediaTemplate 接口中的 Request.Tag
Name	Request	同人声分离模板 CreateMediaTemplate 接口中的 Request.Name
AudioMode	Request	同人声分离模板 CreateMediaTemplate 接口中的 Request.AudioMode
AudioConfig	Request	同人声分离模板 CreateMediaTemplate 接口中的 Request.AudioConfig

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] => NjJhODczMDVfZmNjYTNiMGFJAOIDUJIOADJjNWU=
[ContentType] => application/xml
[ContentLength] => 707
[Key] => t13eedc8532b0as8d09asd15338f
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t13eedc8532b0as8d09asd15338f
[Response] => Array
(
[RequestId] => NjJhODczMDVfZmNjYTNiMGFJAOIDUJIOADJjNWU=
[Template] => Array
(
[TemplateId] => t13eedc8532b0as8d09asd15338f
[Name] => VoiceSeparate-Template-Name
[State] => Normal
[Tag] => VoiceSeparate
[CreateTime] => 2022-06-14T19:33:34+0800
[UpdateTime] => 2022-06-14T19:37:41+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VoiceSeparate] => Array
(
[AudioMode] => IsAudio
[AudioConfig] => Array
(
[Codec] => aac
[Samplerate] => 11025
[Bitrate] => 300
[Channels] => 2
)

)

)

)

)
```

视频增强模板接口

最近更新时间：2022-07-25 08:55:50

简介

本文档提供关于视频增强模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增视频增强模板	用于新增视频增强模板。
更新视频增强模板	用于更新视频增强模板。

新增视频增强模板

功能说明

用于新增视频增强模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaVideoProcessTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/60745 新增视频增强模板
$result = $cosClient->createMediaVideoProcessTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'VideoProcess',
'Name' => 'VideoProcess-Template-Name',
'ColorEnhance' => array(
'Enable' => '',
'Contrast' => '',
'Correction' => '',
'Saturation' => '',
),
'MsSharpen' => array(
'Enable' => '',
'SharpenLevel' => '',
),
));
```

```
// 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
Tag	Request	模板类型：VideoProcess	String	是	无
Name	Request	模板名称 仅支持中文、英文、数字、_、- 和*	String	是	无
ColorEnhance	Request	色彩增强	Container	否，ColorEnhance 和 MsSharpen 不能同时为空	无
MsSharpen	Request	细节增强	Container	否，ColorEnhance 和 MsSharpen 不能同时为空	无

Container 类型 ColorEnhance 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
Enable	Request.ColorEnhance	色彩增强是否开启	String	否	false	true、false
Contrast	Request.ColorEnhance	对比度	String	否	无	值范围：[0, 100]，空字符串（表示自动分析）
Correction	Request.ColorEnhance	色彩校正	String	否	无	值范围：[0, 1000]，空字符串（表示自动分析）
Saturation	Request.ColorEnhance	饱和度	String	否	无	值范围：[0, 300]，空字符串（表示自动分析）

Container 类型 MsSharpen 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
Enable	Request.MsSharpen	细节增强是否开启	String	否	false	true、false
SharpenLevel	Request.MsSharpen	增强等级	String	否	无	值范围：[0, 10]，空字符串（表示自动分析）

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] => NjJhODUzYTBfNzgwYzdkAJIODUJIOxMTIjNDI=
[ContentType] => application/xml
[ContentLength] => 767
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODUzYTBfNzgwYzdkAJIODUJIOxMTIjNDI=
[Template] => Array
(
[TemplateId] => t17855aab7a9s0d8a90sd80accac5
[Name] => VideoProcess-Template-Name
[State] => Normal
[Tag] => VideoProcess
[CreateTime] => 2022-06-14T17:23:44+0800
[UpdateTime] => 2022-06-14T17:23:44+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VideoProcess] => Array
(
[ColorEnhance] => Array
(
[Enable] => true
[Contrast] => 10
[Correction] => 10
[Saturation] => 30
)

[MsSharpen] => Array
(
[Enable] => true
[SharpenLevel] => 5
)

)

)

)

)
```

更新视频增强模板

功能说明

用于更新视频增强模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaVideoProcessTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->updateMediaVideoProcessTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'VideoProcess',
'Name' => 'VideoProcess-Template-Name',
'ColorEnhance' => array(
'Enable' => '',
'Contrast' => '',
'Correction' => '',
'Saturation' => '',
),
'MsSharpen' => array(
'Enable' => '',
'SharpenLevel' => '',
),
));
// 请求成功
print_r($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
Tag	Request	同视频增强模板 CreateMediaTemplate 接口中的 Request.Tag
Name	Request	同视频增强模板 CreateMediaTemplate 接口中的 Request.Name
ColorEnhance	Request	同视频增强模板 CreateMediaTemplate 接口中的 Request.ColorEnhance
MsSharpen	Request	同视频增强模板 CreateMediaTemplate 接口中的 Request.MsSharpen

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] => NjJhODU0YzZfNzgAUOIDUAOIDl8xMWEzMGU=
[ContentType] => application/xml
[ContentLength] => 769
[Key] => t17855aae70b9799ce761accac5
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t17855aae70b9799ce761accac5
[Response] => Array
(
[RequestId] => NjJhODU0YzZfNzgAUOIDUAOIDl8xMWEzMGU=
[Template] => Array
(
[TemplateId] => t17855aae70b9799ce761accac5
[Name] => VideoProcess-Template-Name
[State] => Normal
[Tag] => VideoProcess
[CreateTime] => 2022-06-14T17:23:44+0800
[UpdateTime] => 2022-06-14T17:28:38+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[VideoProcess] => Array
(
[ColorEnhance] => Array
(
[Enable] => true
[Contrast] => 10
[Correction] => 10
[Saturation] => 30
)

[MsSharpen] => Array
(
[Enable] => true
[SharpenLevel] => 5
)

)

)
```

```
)

)
```

超分辨率模板接口

最近更新时间：2022-07-25 08:55:55

简介

本文档提供关于超分辨率模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增超分辨率模板	用于新增超分辨率模板。
更新超分辨率模板	用于更新超分辨率模板。

新增超分辨率模板

功能说明

用于新增超分辨率模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaSuperResolutionTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->createMediaSuperResolutionTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'SuperResolution',
'Name' => 'SuperResolution-Template-Name',
'Resolution' => '',
'EnableScaleUp' => '',
'Version' => '',
));
// 请求成功
print_r($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

节点名称（关键字）	父节点	描述	类型	是否必选	限制
Tag	Request	模板类型：SuperResolution	String	是	无
Name	Request	模板名称：仅支持中文、英文、数字、_、-和*	String	是	无
Resolution	Request	分辨率选项	String	是	1. sdtohd：标清到超清 2. hdto4k：高清到4K
EnableScaleUp	Request	自动缩放开关，默认关闭	String	否	true、false
Version	Request	版本, 默认值 Base	String	否	1.Base:基础版 2.Enhance:增强版

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] => NjJhODc0ODdfNzgwYzdJAOIDJAOIJDIzODI=
[ContentType] => application/xml
[ContentLength] => 636
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODc0ODdfNzgwYzdJAOIDJAOIJDIzODI=
[Template] => Array
(
[TemplateId] => t14e9d3as8d908a90d8a455636
[Name] => SuperResolution-Template-Name
[State] => Normal
[Tag] => SuperResolution
[CreateTime] => 2022-06-14T19:44:07+0800
[UpdateTime] => 2022-06-14T19:44:07+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[SuperResolution] => Array
(
[Resolution] => sdtohd
[EnableScaleUp] => true
[Version] => Enhance
)

)
```

```
)

)
```

更新超分辨率模板

功能说明

用于更新超分辨率模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaSuperResolutionTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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 = $cosClient->updateMediaSuperResolutionTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'SuperResolution',
'Name' => 'SuperResolution-Template-Name',
'Resolution' => '',
'EnableScaleUp' => '',
'Version' => '',
));
// 请求成功
print_r($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
Tag	Request	同超分辨率模板 CreateMediaTemplate 接口中的 Request.Tag

节点名称（关键字）	父节点	描述
Name	Request	同超分辨率模板 CreateMediaTemplate 接口中的 Request.Name
Resolution	Request	同超分辨率模板 CreateMediaTemplate 接口中的 Request.Resolution
EnableScaleUp	Request	同超分辨率模板 CreateMediaTemplate 接口中的 Request.EnableScaleUp
Version	Request	同超分辨率模板CreateMediaTemplate接口中的Request.Version

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] => NjJhODc1NTBfNzgwYzdkAJOSJDOIDRkYjY=
[ContentType] => application/xml
[ContentLength] => 638
[Key] => t14e9d1151b43641455636
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t14e9d1151b43641455636
[Response] => Array
(
[RequestId] => NjJhODc1NTBfNzgwYzdkAJOSJDOIDRkYjY=
[Template] => Array
(
[TemplateId] => t14e9d1151b43641455636
[Name] => SuperResolution-Template-Name
[State] => Normal
[Tag] => SuperResolution
[CreateTime] => 2022-06-14T19:44:07+0800
[UpdateTime] => 2022-06-14T19:47:28+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[SuperResolution] => Array
(
[Resolution] => sdtohd
[EnableScaleUp] => true
[Version] => Enhance
)

)

)
```

```
)
```

图片处理模板接口

最近更新时间：2022-07-25 08:55:59

简介

本文档提供关于图片处理模板接口的 API 概览和 SDK 示例代码。

API	操作描述
新增图片处理模板	用于新增图片处理模板。
更新图片处理模板	用于更新图片处理模板。

新增图片处理模板

功能说明

用于新增图片处理模板。

方法原型

```
public Guzzle\Service\Resource\Model createMediaPicProcessTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/67225 新增图片处理模板
$result = $cosClient->createMediaPicProcessTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => 'PicProcess',
'Name' => 'PicProcess-Template-Name',
'PicProcess' => array(
'IsPicInfo' => '',
'ProcessRule' => 'imageMogr2/rotate/90',
),
));
// 请求成功
print_r($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
Tag	Request	模板类型：PictureProcess	String	是	无
PicProcess	Request	容器格式	Container	是	无

Container 类型 PicProcess 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
IsPicInfo	Request.Container	是否返回原图信息	String	否	false	true、false
ProcessRule	Request.Container.Rule	图片处理规则	String	是	无	1. 基础图片处理参见 基础图片处理 文档 2. 图片压缩参见 图片压缩 文档 3. 盲水印参见 盲水印 文档

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] => NjJhODc2NDRfZmNjYTNiOAUIODUDmZmYjQ=
[ContentType] => application/xml
[ContentLength] => 594
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template
[Response] => Array
(
[RequestId] => NjJhODc2NDRfZmNjYTNiOAUIODUDmZmYjQ=
[Template] => Array
(
[TemplateId] => t10e5880d9efdfd09eb19basdasd
[Name] => PicProcess-Template-Name
[State] => Normal
[Tag] => PicProcess
[CreateTime] => 2022-06-14T19:51:32+0800
[UpdateTime] => 2022-06-14T19:51:32+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[PicProcess] => Array
(
[IsPicInfo] => true
[ProcessRule] => imageMogr2/rotate/90
)
```

```
)

)

)
```

更新图片处理模板

功能说明

用于更新图片处理模板。

方法原型

```
public Guzzle\Service\Resource\Model updateMediaPicProcessTemplate(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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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://cloud.tencent.com/document/product/436/67228 更新图片处理模板
$result = $cosClient->updateMediaPicProcessTemplate(array(
'Bucket' => 'examplebucket-125000000', //存储桶名称，由BucketName-Appid 组成，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Key' => '', // TemplateId
'Tag' => 'PicProcess',
'Name' => 'PicProcess-Template-Name',
'PicProcess' => array(
'IsPicInfo' => '',
'ProcessRule' => 'imageMogr2/rotate/90',
),
));
// 请求成功
print_r($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
Tag	Request	模板类型：PictureProcess	String	是	无
PicProcess	Request	容器格式	Container	是	无

Container 类型 PicProcess 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
IsPicInfo	Request.Container	是否返回原图信息	String	否	false	true、false
ProcessRule	Request.Container.Rule	图片处理规则	String	是	无	1. 基础图片处理参见 基础图片处理 文档 2. 图片压缩参见 图片压缩 文档 3. 盲水印参见 盲水印 文档

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] => NjJhODc2Y2JfNzgwYzAIUJOIUDJIOSxMjdmZDQ=
[ContentType] => application/xml
[ContentLength] => 596
[Key] => t10e5880d9efdfd61bb11d09eb19b
[Bucket] => examplebucket-125000000
[Location] => examplebucket-125000000.ci.ap-guangzhou.myqcloud.com/template/t10e5880d9efdfd61bb11d09eb19b
[Response] => Array
(
[RequestId] => NjJhODc2Y2JfNzgwYzAIUJOIUDJIOSxMjdmZDQ=
[Template] => Array
(
[TemplateId] => t10e5880d9efdfd61bb11d09eb19b
[Name] => PicProcess-Template-Name
[State] => Normal
[Tag] => PicProcess
[CreateTime] => 2022-06-14T19:51:32+0800
[UpdateTime] => 2022-06-14T19:53:47+0800
[BucketId] => examplebucket-125000000
[Category] => Custom
[PicProcess] => Array
(
[IsPicInfo] => true
[ProcessRule] => imageMogr2/rotate/90
)
```

```
)
)
)
```

搜索模板接口

最近更新时间：2022-07-26 10:16:40

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://console.cloud.tencent.com/cam/capi
$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，https://console.cloud.tencent.com/cam/capi
$region = "ap-beijing"; //替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
$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，可以在COS控制台查看 https://console.cloud.tencent.com/cos5/bucket
'Tag' => '', // 模板 Tag：Animation、Snapshot、Watermark、Transcode、Concat、HighSpeedHd、VideoMontage、VoiceSeparate、VideoProcess、PicProcess
'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
)
```

Python SDK
快速入门

最近更新时间：2023-01-04 20:39:34

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。如果您仍在使用 JSON 版本 SDK，请 升级到 XML Python SDK。

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
```

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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid
```

```
组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
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=token, Proxies=proxies)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
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，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

endpoint = 'cos.accelerate.myqcloud.com' # 替换为用户的 endpoint 或者 cos全局加速域名，如果使用桶的全球加速域名，需要先开启桶的全球加速功能，请参见https://cloud.tencent.com/document/product/436/38864
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Endpoint=endpoint, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid
```

```
组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
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，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

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https://cloud.tencent.com/document/product/436/36638
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Domain=domain, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
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，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

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https://cloud.tencent.com/document/product/436/18670
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Endpoint=endpoint, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
```

```
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，https://console.cloud.tencent.com/cam/capi
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，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

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，参见https://cloud.tencent.com/document/product/436/18670
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Domain=domain, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
```

```
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
```

```
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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，请在参数 Bucket 中带上 Appid。Bucket 由 BucketName-Appid 组成
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，https://console.cloud.tencent.com/cam/capi
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，https://console.cloud.tencent.com/cam/capi
region = 'ap-beijing' # 替换为用户的 region，已创建桶归属的region可以在控制台查看，https://console.cloud.tencent.com/cos5/bucket
# COS支持的所有region列表参见https://cloud.tencent.com/document/product/436/6224
token = None # 如果使用永久密钥不需要填入token，如果使用临时密钥需要填入，临时密钥生成和使用指引参见https://cloud.tencent.com/document/product/436/14048
scheme = 'https' # 指定使用 http/https 协议来访问 COS，默认为 https，可不填

config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
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：2022-12-12 16:22:35

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":"imageView2/format/png"}]}')
```

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":"imageView2/format/png"}]}'
    )
```

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
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
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.jpg',
'ETag': '"aff1b996bcc63a0f0259df0de2fa989f38c5ce7e"',
'ImageInfo': {
'Format': 'JPEG',
'Width': '300',
'Height': '168',
'Quality': '74',
'Ave': '0x1a3451',
'Orientation': '0'
}
},
'ProcessResults': {
'Object': {
'Key': 'format.png',
'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/format.png',
'Format': 'png',
'Width': '300',
'Height': '168',
'Size': '77063',
'Quality': '74',
'ETag': '"a07dc5bcfa238e7d23d2d5884da4ac328aaaa9c6"'
}
}
}
```

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

版权所有：腾讯云计算（北京）有限责任公司

节点名称	类型	描述
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

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

云上数据处理

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":"imageView2/format/png"}]}')
```

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":"imageView2/format/png"}]}')
```

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.jpg',
'ETag': '"aff1b996bcc63a0f0259df0de2fa989f38c5ce7e"',
'ImageInfo': {
'Format': 'JPEG',
'Width': '300',
'Height': '168',
'Quality': '74',
'Ave': '0x1a3451',
'Orientation': '0'
}
},
'ProcessResults': {
```

```
'Object': {
'Key': 'format.png',
'Location': 'examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/format.png',
'Format': 'png',
'Width': '300',
'Height': '168',
'Size': '77063',
'Quality': '74',
'ETag': '"a07dc5bcfa238e7d23d2d5884da4ac328aaaa9c6"'
}
}
}
```

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

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：

版权所有：腾讯云计算（北京）有限责任公司

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

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：2022-04-22 12:17:37

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：2022-04-22 12:17:41

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, **kwargs)
```

请求示例

```
# 添加盲水印
watermark_url = 'http://{bucket}.cos.{region}.myqcloud.com/watermark.png'.format(bucket='examplebucket-1250000000', region=region)
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": "watermark/3/type/1/image/' +
watermark_url_base64 + '" }]}'
)

# 提取盲水印
sample_url = 'http://{bucket}.cos.{region}.myqcloud.com/sample.png'.format(bucket='examplebucket-1250000000', region=region)
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": "watermark/4/type/1/image/' +
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型	是否必填
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：2022-04-22 12:17:46

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',
'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

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

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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型
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
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：2022-04-22 12:17:50

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='auto', Mode='exactframe', **kwargs)
```

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	是否必填	类型
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

内容审核

图片审核

最近更新时间：2022-10-14 16:05:03

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/tttt.jpg',
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
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。

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/porn_ocr.png',
'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/pron_plitics_ads_terrism.png',
}]
)
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

版权所有：腾讯云计算（北京）有限责任公司

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。 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。	Array	否

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。

视频审核

最近更新时间：2022-10-14 16:05:12

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
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

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：2022-08-04 10:55:32

简介

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：2022-10-14 16:05:29

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

参数名称	描述	类型	是否必选
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：2022-10-14 16:05:38

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
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。

版权所有：腾讯云计算（北京）有限责任公司

网页审核

最近更新时间：2022-08-04 10:55:44

简介

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型	是否必选
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。

文档处理
文档转码

最近更新时间：2022-04-29 17:39:31

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

参数名称	描述	类型	是否必选
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，即转换全部页。	Int	否
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

参数名称	描述	类型	是否必选
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：2022-09-27 16:29:41

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=None, PassWord=None, Comment=0, Sheet=1,
ExcelPaperDirection=0, ExcelRow=0, ExcelCol=0, ExcelPaperSize=None, TxtPagination=False,
ImageParams=None, Quality=100, Scale=100, ImageDpi=96, **kwargs)
```

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
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

版权所有：腾讯云计算（北京）有限责任公司

异步处理任务接口

最近更新时间：2022-09-27 16:29:46

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, OutputRegion, OutputObject, SrcType=None, TgtType=None, StartPage=None, EndPage=-1, SheetId=0, PaperDirection=0, PaperSize=0, DocPassword=None, Comments=None, PageRanges=None, ImageParams=None, Quality=100, Zoom=100, ImageDpi=96, PicPagination=0, **kwargs)
```

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='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxf1',
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
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
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


```
{
'JobsDetail': {
'Code': 'Success',
'CreationTime': '2022-09-14T00:00:00+0800',
'EndTime': '-',
'Input': {
'BucketId': 'test-125xxxxxxxxxx',
'Object': '1.txt',
'Region': 'ap-chongqing'
},
'JobId': 'dcf48xxxxxxxxxxxxxxxxxxxxxxxxxxc523',
'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': 'p4bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx57f1',
'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='d18a9xxxxxxxxxxxxxxxxxxxff1aa',
)
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': 'ap-chongqing'},
'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'}, {'PageNo': '3', 'TgtUri': '/test_doc/normal/abc_3.jpg'}],
'SuccPageCount': '2',
'TaskId': None,
'TgtType': None,
'TotalPageCount': '7'
},
'JobLevel': '0',
'Output': {'Bucket': 'testpic-1253960454', 'Object': '/test_doc/normal/abc_${Number}.jpg', 'Region': 'ap-chongqing'}
},
```

版权所有：腾讯云计算（北京）有限责任公司

```
'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=None, OrderByTime='Desc', States='All', Size=10, NextToken='', **kwargs):
```

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': 'ap-chongqing'},
'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'}, {'PageNo': '3', 'TgtUri': '/test_doc/normal/abc_3.jpg'}],
'SuccPageCount': '2',
'TaskId': None,
'TgtType': None,
'TotalPageCount': '7'
},
'JobLevel': '0',
'Output': {'Bucket': 'testpic-1253960454', 'Object': '/test_doc/normal/abc_${Number}.jpg', 'Region': 'ap-chongqing'}
},
'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：2022-09-27 16:29:50

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='', PageSize='', **kwargs):
```

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
```

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': 'p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxxf1',
'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='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxxf1',
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': 'p4bdf22746b0xxxxxxxxxxxxxxxxxx',
'Name': 'doc-queue',
'State': 'Active',
'NotifyConfig': {
'Url': 'http://www.demo.callback.com',
'Event': 'TaskFinish',
'Type': 'Url',
'State': 'On',
'ResultFormat':
'JSON',
'MqMode': None,
```

版权所有：腾讯云计算（北京）有限责任公司

```
'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：2022-09-27 16:29:54

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', DstType='html', HtmlParams=None, HtmlWaterword=None,
HtmlFillStyle=None,
HtmlFront=None, HtmlRotate=None, HtmlHorizontal=None, HtmlVertical=None, **kwargs):
```

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

版权所有：腾讯云计算（北京）有限责任公司

名称	参数说明	类型	是否必选
HtmlVertical	水印文字垂直间距，单位 px，默认为100	String	否

查询文档预览开通状态

最近更新时间：2022-09-27 16:29:59

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='', PageSize='', **kwargs)
```

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

版权所有：腾讯云计算（北京）有限责任公司

内容识别
语音识别
任务接口

最近更新时间：2022-09-27 16:27:55

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, OutputRegion, OutputObject, TemplateId=None,
SpeechRecognition=None, CallBack=None, CallBackFormat=None, CallBackType=None, CallBackMqConfig=None, **kwargs)
```

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': 'ap-chongqing'},
'JobId': 'sf1398ed0343911eda6d733209028ff12',
'Message': None,
'Operation': {
'JobLevel': '0',
'Output': {'Bucket': 'test-12539xxxxx', 'Object': 'result.txt', 'Region': 'ap-chongqing'},
'TemplateId': 't1ada6f282d29742db83244e085exxxxxx',
'TemplateName': 'name'
},
'QueueId': 'p7369ebcf8f93414c8250c966xxxxxxxx',
'StartTime': '-',
'State': 'Submitted',
'Tag': 'SpeechRecognition'
}
}
```

参数名称	描述	类型

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='s18a9xxxxxxxxxxxxxxxxxxxxff1aa',
)
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': 'ap-chongqing'},
'JobId': 'sabfb88d6343a11ed8c375fe3d77ef1d7',
'Message': None,
```

```
'Operation': {
'JobLevel': '0',
'Output': {'Bucket': 'testpic-1253960454', 'Object': 'result.txt', 'Region': 'ap-chongqing'},
'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=None, OrderByTime='Desc', States='All', Size=10, NextToken='', **kwargs):
```

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='p4bdxxxxxxxxxxxxxxxxxxxx57f1',
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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	描述	类型
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': '2022-09-14T22:36:51+0800',
'EndTime': '2022-09-14T22:37:01+0800',
'Input': {'BucketId': 'test-125396xxxx', 'Object': 'gaobai.mp4', 'Region': 'ap-chongqing'},
'JobId': 'sabfb88d6343a11ed8c375fe3dxxxxxxx',
'Message': None,
'Operation': {
'JobLevel': '0',
'Output': {'Bucket': 'test-125396xxxx', 'Object': 'result.txt', 'Region': 'ap-chongqing'},
'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：2022-09-27 16:29:11

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='', PageSize='', **kwargs)
```

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': 'NjMyMWE2ZjZfZWM0YTYyNjRxxxxxxxxxxx',
'PageNumber': '1',
'PageSize': '1',
'AsrBucketList': [{
'Name': 'test-125xxxxxx',
```

```
'CreateTime': '2022-08-11T00:00:00+0800',
'Region': 'ap-chongqing',
'AliasBucketId': None,
'BucketId': 'test-125xxxxxxx'
}]
}
```

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：2022-09-27 16:29:27

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='', PageSize='', **kwargs):
```

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': 'p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxxf1',
'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='p4bdf22xxxxxxxxxxxxxxxxxxxxxxxxxf1',
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': 'p4bdf22746b0xxxxxxxxxxxxxxxxxx',
'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：2022-04-22 12:18:15

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/cover/0"}]}'
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

模板接口
语音识别

最近更新时间：2022-09-27 16:28:13

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, ConvertNumMode=0, SpeakerDiarization=0,
SpeakerNumber=0, FilterPunc=0, OutputFileType='txt', **kwargs)
```

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

节点名称（关键字）	描述	类型	必选
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； • 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。	int	否
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='t1bdxxxxxxxxxxxxxxxxx94a9',
)
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='', PageNumber=1, PageSize=10, **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
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': 'NjMyMjIjNTIfMTIwNjUzMDIfMmUzYV8xMWNh',
'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'
}
}
]
}
```

响应字段详情详见获取语音识别模板响应信息

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, ChannelNum,
ResTextFormat, FilterDirty=0, FilterModal=0, ConvertNumMode=0, SpeakerDiarization=0,
SpeakerNumber=0, FilterPunc=0, OutputFileType='txt', **kwargs)
```

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。

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',
'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

小程序 SDK
快速入门

最近更新时间：2023-01-06 10:49:36

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 Android App、iOS App，需要使用对应的 Android 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

> 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。

```
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量，请求时会用地域域名
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用putObject,sdk版本至少需要v1.3.0
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
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 invalid');
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：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使用 putObject,SDK 版本至少需要v1.3.0
// 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/master/server/
// 服务端其他语言参考 COS STS SDK ：https://github.com/tencentyun/qcloud-cos-sts-sdk
// STS 详细文档指引看：https://cloud.tencent.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 invalid');
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：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使用 putObject,SDK 版本至少需要v1.3.0
// 必选参数
getAuthorization: function (options, callback) {
// 服务端示例：https://github.com/tencentyun/qcloud-cos-sts-sdk/edit/master/scope.md
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 invalid');
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：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求时重复使用
});
}
});
}
});
```

> **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使用 putObject,SDK 版本至少需要v1.3.0
// 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://cloud.tencent.com/document/product/436/6474
// 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 / 等
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('authorization invalid');
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 sessionToken 传给 SecurityToken
});
}
});
},
});
```

- **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.cloud.tencent.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使用putObject,sdk版本至少需要v1.3.0
});
```

构造函数参数说明

参数名	参数描述	类型	是否必填

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
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

版权所有：腾讯云计算（北京）有限责任公司

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，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);
}
```

版权所有：腾讯云计算（北京）有限责任公司

> ⚠ 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);
});
```

版权所有：腾讯云计算（北京）有限责任公司

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用putObject
});

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.jpg）支持中文，必须字段 */
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使用简单上传。可自行设置，非必须 */
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);
});
```

图片处理
基础图片处理

最近更新时间：2022-04-22 12:18:28

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

```
<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", "rule": "imageMogr2/thumbnail/200x/"}]}',
},
},
(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": "imageMogr2/thumbnail/200x/"}]}',
},
},
(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：2022-04-22 12:18:32

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": "desample_photo.jpg", "rule": "watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEy
NTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn"}]}',
},
},
(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/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAwMDQucGljc2gubXlxY2xvdWQuY29tL3NodWI5aW4uanBn`,
},
(err, data) => {
console.log(err || data);
},
);
},
});
```

媒体处理
媒体 bucket 接口

最近更新时间：2022-04-22 12:18:38

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'ap-shanghai,ap-beijing'，非必须 */
// bucketNames: 'test-1250000000', /** 存储桶名称，精确搜索，例如'test-1250000000'，支持多个值用逗号分隔如'test1-1250000000,test2-1250000000'，非必须 */
// 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隔如'test1,test2'，非必须 */
}
}, 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

参数名称	参数描述	类型	是否必填
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

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：2022-04-22 12:18:42

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
- - - - - 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

版权所有：腾讯云计算（北京）有限责任公司

参数名	参数描述	类型
- - - - 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
- - - - - 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：2022-04-22 12:18:46

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

版权所有：腾讯云计算（北京）有限责任公司

参数名称	参数描述	类型	是否必填
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

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：2022-12-27 15:18:58

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 小程序 SDK 版本需要大于等于 v1.1.1。

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。Content，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。Content，Object 和 Url 只能选择其中一种。	String	否
Content	Request.Input	图片文件的内容，需要先经过 base64 编码。Content，Object 和 Url 只能选择其中一种，传入多个时仅一个生效，按 Content，Object， Url 顺序。	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：2022-06-29 14:41:25

简介

本文档提供关于视频审核的 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-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	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，支持 float 格式，执行精度精确到毫秒。	Float	否

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：2022-06-29 14:41:28

简介

本文档提供关于音频审核的 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

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：2022-10-14 16:06:23

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 小程序 SDK 版本需要大于等于 v1.1.1。

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（Input传入Object）
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。

文档审核

最近更新时间：2022-10-14 16:06:35

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。

> 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。

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 postDocumentAuditing() {
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
var url = 'https://' + host + '/document/auditing';
var body = COS.util.json2xml({
Request: {
Input: {
Object: 'test.xlsx', /* 需要审核的文档文件，存储桶里的路径 */
},
Conf: {
BizType: '',
DetectType: 'Porn',
}
}
});
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'POST',
Url: url,
```

```
Key: '/document/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
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
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
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

返回结果说明

详情请参见 提交文档审核任务。

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。

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 getDocumentAuditingResult() {
var jobId = 'sd7815c21caff611eca12f525400d88xxx'; // jobId可以通过提交文档审核任务返回
var host = config.Bucket + '.ci.' + config.Region + '.myqcloud.com';
```

```
var url = 'https://' + host + '/document/auditing/' + jobId;
cos.request({
Bucket: config.Bucket,
Region: config.Region,
Method: 'GET',
Key: '/document/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 查询文档审核任务结果。

网页审核

最近更新时间：2022-10-14 16:06:43

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 小程序 SDK 版本需要大于等于 v1.1.1。

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，可以调用查询文本审核结果api查询
},
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

版权所有：腾讯云计算（北京）有限责任公司

节点名称（关键字）	父节点	描述	类型	是否必选
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示的 html 内容。取值为 true 或者 false，默认为 false。	Boolean	否

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。

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：2022-06-08 17:55:12

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：

- 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