对象存储

附录

腾讯云

【版权声明】

【商标声明】

【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或95716。

文档目录

C++ SDK
iOS SDK
Java SDK
JavaScript SDK
PHP SDK
Python SDK

附录
历史版本
历史版本地域列表

最近更新时间：2020-04-15 16:20:16

> **注意**
> 您目前查阅的是历史版本地域文档，已不再更新和维护，我们建议您查阅新版 地域和访问域名 文档。

COS 支持多地域存储，不同地区默认访问域名不同。建议根据自己的业务场景选择就近的地域存储，可以提高对象上传、下载速度。

可用地域及访问域名

> **注意**
> - 默认域名在创建好存储桶后，可通过 对象存储控制台 的存储桶【域名管理】查看。
> - bucketname 是在创建存储桶时为存储桶命名的名称，可通过 对象存储控制台 的存储桶【基础配置】查看。
> - APPID 是在成功申请腾讯云账户后，系统分配的账户标识之一，可通过 腾讯云控制台 【账号信息】查看。

地域	地域简称	默认下载域名	上传域名
北京一区（华北）	tj	<bucketname-APPID>.costj.myqcloud.com	tj.file.myqcloud.com
北京	bj	<bucketname-APPID>.cosbj.myqcloud.com	bj.file.myqcloud.com
上海（华东）	sh	<bucketname-APPID>.cossh.myqcloud.com	sh.file.myqcloud.com
广州（华南）	gz	<bucketname-APPID>.cosgz.myqcloud.com	gz.file.myqcloud.com
成都（西南）	cd	<bucketname-APPID>.coscd.myqcloud.com	cd.file.myqcloud.com
中国香港	hk	<bucketname-	hk.file.myqcloud.c

		APPID>.coshk.myqcloud.com	om
新加坡	sgp	<bucketname-APPID>.cossgp.myqcloud.com	sgp.file.myqcloud.com
多伦多	ca	<bucketname-APPID>.cosca.myqcloud.com	ca.file.myqcloud.com
法兰克福	ger	<bucketname-APPID>.cosger.myqcloud.com	ger.file.myqcloud.com

例如：
用户在所属地域广州创建了一个存储桶，存储桶名中用户自定义字符串部分为 example，系统自动为用户生成的数字串 APPID 为 1234567890，则其默认下载域名为：

```
example-1234567890.cosgz.myqcloud.com
```

内网访问判断方法

相同地域内腾讯云产品访问，将会自动使用内网连接，不产生流量费用。因此选购腾讯云不同产品时，建议尽量选择相同地域，减少您的费用。进一步确认是否内网访问可以参考如下方法：
以腾讯 CVM 访问 COS 为例，判断是否使用内网访问 COS ，可以在 CVM 上 使用 `nslookup` 命令解析 COS 域名，若返回内网 IP，则表明 CVM 和 COS 之间是内网访问，否则为外网访问。
内网 IP 地址一般形如 `10.*.*.*` 、 `100.*.*.*` VPC网络一般为 `169.*.*.*` 等。
假设 `mybucket-1258888888.cos.ap-guangzhou.myqcloud.com` 为目标存储桶地址，其下方的 `Address: 10.148.214.13` 表示从内网访问。

```
nslookup mybucket-1258888888.cos.ap-guangzhou.myqcloud.com

Server:         10.138.224.65
Address:        10.138.224.65#53

Name:   mybucket-1258888888.cos.ap-guangzhou.myqcloud.com
Address: 10.148.214.13
Name:   mybucket-1258888888.cos.ap-guangzhou.myqcloud.com
Address: 10.148.214.14
```

默认 CDN 加速域名

最近更新时间：2022-11-25 17:28:57

> **注意**
> 自2022年5月9日起，对象存储（Cloud Object Storage，COS）服务将不再支持新增默认 CDN 加速域名。您已开启、或曾经开启的默认 CDN 加速域名不会受到影响，可以继续使用，但建议您使用自定义 CDN 加速域名代替默认 CDN 加速域名。关于自定义 CDN 加速域名的操作指引，请参见 开启自定义 CDN 加速域名 文档。

配置说明

默认 CDN 加速域名是 COS 为存储桶自动分配的一个 CDN 加速域名，形如 `BucketName-APPID.file.myqcloud.com`，当开启了默认 CDN 加速，则可通过该域名获得访问加速体验。

操作步骤

1. 选择存储桶

登录 对象存储控制台 ，在左侧导航栏中，单击**存储桶列表**，并单击需要加速的存储桶，进入存储桶管理页。

2. 进入配置页面

> **注意**
> 从未使用过腾讯云 CDN 服务的用户将无法进入**域名管理**，需先前往 CDN 控制台 开通 CDN 服务。

单击左侧的**域名与传输管理 > 默认 CDN 加速域名**，找到**默认 CDN 加速域名**配置项。默认 CDN 加速域名的初始状态为**关闭**。单击**编辑**，将当前状态设置为**开启**，然后按照以下配置项说明进行配置。

- **加速域名**：默认 CDN 加速域名是 COS 为存储桶自动分配的一个 CDN 加速域名，形如 `BucketName-APPID.file.myqcloud.com` ，当开启了默认 CDN 加速，则可通过该域名获得访问加速体验。
- **加速地域**：支持中国境内、中国境外以及全球加速，其中全球加速指支持所有地域之间的存储桶加速。
- **源站类型**：默认是**默认源站**，若作为源站的存储桶开启了静态网站，并且希望为静态网站加速，可以将源站类型设置为**静态网站源站**，具体详情请参阅 CDN 加速概述 。
- **源站域名**：即 COS 源站域名，是创建存储桶时，由系统根据存储桶名称和地域自动生成的，要与默认 CDN 加速域名区分开。

3. 开启回源鉴权（可选）

> **注意**
> 对于私有读存储桶，开启回源鉴权和 CDN 服务授权会使 CDN 边缘节点访问源站时无需携带签名，CDN 缓存资源会进行公网分发，导致数据的安全性受到影响，因此强烈建议开启 CDN 鉴权。

回源鉴权是用来验证 CDN 边缘节点的服务身份以阻止非法访问，具体情况如下：

- 公有读存储桶：CDN 边缘节点无需任何授权即可直接访问存储桶，无需开启回源鉴权。
- 私有读存储桶：CDN 边缘节点需经回源鉴权验证服务身份，验证通过的 CDN 边缘节点才能访问存储桶中对象。

（1）完成 CDN 服务授权

> **说明**
> 开启回源鉴权前，需添加 CDN 服务授权。

操作步骤如下：

添加 CDN 服务授权是为了授予 CDN 边缘节点对存储桶进行操作的服务身份，具体情况如下：

- 公有读存储桶：CDN 边缘节点无需任何授权即可直接访问存储桶，无需添加 CDN 服务授权。
- 私有读存储桶：CDN 边缘节点需被授予特殊的服务身份才能访问存储桶，单击**添加 CDN 服务授权**授予 CDN 边缘节点服务身份，选择**我同意以上授权**，并单击**确定**。

完成 CDN 服务授权后，CDN 边缘节点可以对存储桶进行 Get Object、Head Object、Options Object 这三项操作。系统会自动写入存储桶访问策略，策略示例如下。之后 CDN 节点在需要回源时无需再进行其他操作。

```
{
 "Statement": [
  {
   "Action": [
    "name/cos:GetObject",
    "name/cos:HeadObject",
    "name/cos:OptionsObject"
   ],
   "Effect": "allow",
   "Principal": {
    "qcs": [
     "qcs::cam::uin/100000000001:service/cdn"
    ]
   },
   "Resource": [
    "qcs::cos:ap-chengdu:uid/1250000000:examplebucket-1250000000/*"
   ]
  }
 ],
 "version": "2.0"
}
```

（2）检测到完成授权后，即可单击开启**回源鉴权**。

4. 开启 CDN 加速

单击**保存**后，可以看到默认 CDN 加速域名在部署中（预计5分钟左右配置完成）。

5. 配置 CDN 鉴权

> **注意**
> 开启了默认 CDN 加速后，任何人都可以通过此域名直接访问源站，如果您的数据有一定的私密性，请您务必**开启鉴权配置**来保护您的源站数据。

- 开启默认 CDN 加速域名和回源鉴权后，默认 CDN 加速域名管理界面下会出现一条 CDN 鉴权状态提示，可通过提示上的**鉴权配置**直接跳转到对应域名的**访问控制**页面进行配置。

- 或者进入 CDN 控制台，单击**域名管理**，然后选择单击默认 CDN 加速域名，最后单击**访问控制>鉴权配置**。具体配置步骤详见 鉴权配置 。

6. 关闭功能

完成以上步骤后，即开启默认 CDN 加速完成。如需关闭默认 CDN 加速，可通过以下方式关闭：

- 在默认 CDN 加速域名管理界面，单击**编辑**，把状态从**开启**改成**关闭**，单击**保存**，大约需要5分钟时间进行部署。部署完成之后在 CDN 控制台上该域名的状态由**已启动**变为**已关闭**。
- 在 CDN 控制台 可以对域名进行关闭/删除操作，详情请见 域名操作 。

> **注意**
> 在 CDN 控制台删除的只是默认 CDN 加速域名在 CDN 加速时的记录，并不会将默认 CDN 加速域名真正抹去，当需要再次开启 CDN 加速时，可在 COS 控制台重新开启默认 CDN 加速域名。

历史版本 API
简介

最近更新时间：2020-12-22 11:40:09

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

腾讯云对象存储（COS）服务的 RESTful API 是一种轻量级的，无连接状态的，可以直接通过 HTTP/HTTPS 的请求和响应来和腾讯云对象存储（COS）后台进行交互操作的 Web 服务。
JSON API 是腾讯云 COS 服务在推出 XML API 前为用户提供接入使用 COS 的 API 接口，上传域名为 [Region].file.myqcloud.com。JSON API 接口与标准 XML 的 API 底层架构相同，数据互通，可以交叉使用，但是接口不兼容，域名不一致。
腾讯云 COS 的 XML API 服务推出后，推荐您使用 XML API 接口， JSON API 接口日后将保持维护状态，可以正常使用但是不发展新特性。

> **注意**
> 目前 COS 的可用地域（Region）根据 XML API和 JSON API 有不同的取值，在使用不同的 API 和对应的 SDK 时要求使用对应的地域字段，详细信息请参见 历史版本地域列表 文档。
> 为了能更高效的使用腾讯云对象存储服务的 JSON API ，请在查阅其他的 API 文档之前，先详细阅读 签名算法 。

术语信息

文中会出现的一些主要概念和术语：

名称	描述
AppID	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
SecretID	开发者拥有的项目身份识别 ID，用以身份认证
SecretKey	开发者拥有的项目身份密钥
Bucket	COS 中用于存储数据的容器
Object	COS 中存储的具体文件，是存储的基本实体

快速入门

要使用对象存储 API，需要先执行以下步骤：

1. 购买腾讯云对象存储（COS）服务。
2. 在腾讯云 对象存储控制台 里创建一个 Bucket。
3. 在控制台 个人 API 密钥 页面里获取 APPID、SecretID、SecretKey 内容。
4. 编写一个请求签名算法程序（或使用任何一种服务端 SDK）。
5. 计算签名，调用 API 执行操作。

API 概览

最近更新时间：2020-02-12 11:49:02

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

腾讯云对象存储服务（COS）相关 JSON API 接口及说明如下：

目录操作

API	请求方式	功能说明
创建目录 API	POST	在 Bucket 中创建一个新目录
查询目录属性 API	GET	查询目录的属性信息
列出目录 API	GET	列出指定目录下的所有文件目录
删除目录 API	POST	删除指定目录

文件操作

API	请求方式	功能说明
简单上传文件 API	POST	上传 ≤ 20MB 的文件到 Bucket
初始化分片上传 API	POST	初始化文件分片上传流程
逐个上传分片 API	POST	对大于 20 MB 的文件进行逐片上传
结束分片上传API	POST	结束文件分片上传流程
查询上传分片 API	GET	查询分片的断点和 session
查询文件属性 API	GET	查询文件的属性信息
更新文件属性 API	POST	对 COS 中某个文件的属性进行设置或修改
复制文件 API	POST	将 COS 中指定文件复制一份
下载文件 API	GET	将 COS 中指定文件下载到本地
移动文件 API	POST	将文件移动或重命名
删除文件 API	POST	将 COS 中某个文件删除

签名算法

最近更新时间：2023-01-12 12:19:06

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

腾讯云对象存储提供多种语言 SDK，SDK 内置了生成签名的方法，详情请参见各语言 SDK 的**生成签名**章节。

> **注意**
> - 建议用户 使用临时密钥 生成签名，通过临时授权的方式进一步提高签名的安全性。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。

签名鉴权

为什么需要鉴权?

业务端购买腾讯云（对象存储）服务，来作为其业务组成的一部分（云存储或内容分发加速），并最终提供给用户。所以当用户访问腾讯云服务时，需要先征得业务端的“许可”。那么，用户如何才能向腾讯云证明自己已经得到“访问许可”了呢?
简单的说，腾讯云和业务端约定了一套规则，通过这套规则，业务端可以产生一个“钥匙”。当用户需要访问业务端购买的腾讯云服务时，业务端会将这个钥匙传递给用户，然后用户拿着这个钥匙去向腾讯云“证明”自己已经拿到了“访问许可”。腾讯云收到用户的请求后，会先鉴别“钥匙”是否有效，若有效，则允许访问云服务，若无效，则返回 Authorization 错误消息。
这套规则，就是腾讯云的“鉴权系统”，这把“钥匙”，就是下文要解释的“签名”。

签名有哪几种?

签名本质上是一个加密的字符串，目前在腾讯云中，分为**多次有效签名**和**单次有效签名**:
多次有效签名：签名中部分不绑定文件 fileid ，部分绑定文件 fileid ，开启token防盗链的下载、文件简单上传、文件分片上传可以绑定 fileid，同时支持前缀匹配。多次有效签名需要设置一个大于当前时间的有效期，有效期内此签名可多次使用，有效期最长可设置三个月。
单次有效签名：签名中绑定文件 fileid ，有效期必须设置为0，此签名只可使用一次，且只能应用于被绑定的文件或文件夹操作。

哪些场景需要用到签名?

腾讯云对象存储（COS）对签名的适用场景做了如下限制：

场景	适用签名	签名中是否需要绑定 fileid

下载（不开启 token 防盗链）	不验证签名	–
下载（开启 token 防盗链）	多次有效签名	可选
简单上传文件	多次有效签名	可选
分片上传文件	多次有效签名	可选
查询目录、文件属性	多次有效签名	否
创建目录	多次有效签名	否
删除目录、文件	**单次有效签名**	是
更新文件属性	**单次有效签名**	是
移动（重命名）文件	**单次有效签名**	是

如何使用签名来完成鉴权?

腾讯云对象存储（COS）通过签名来验证请求的合法性。业务端通过将签名授权给用户端，使其具备上传下载及管理指定资源的权限。
当用户使用 RESTful API 对腾讯云对象存储资源进行操作时，必须要在每个 Http/Https 请求的 Header 中填写 Host 和 Authorization，其中的 Authorization 参数，即为“签名算法”最终得到的 sign 字符串。

参数名称	必选	类型	描述
Host	是	String	文件云服务器域名，固定为[region].file.myqcloud.com
Authorization	是	String	用户的有效签名，用于鉴权

腾讯云对象存储目前提供了 COS 签名工具 供用户生成签名，如有需要可前往使用。

签名算法

腾讯云的各类 SDK 已经提供了标准的签名计算方法，业务端只需填写相关参数，即可得到签名字符串。用户也可以根据 RESTful API 自行计算签名。
对于签名的计算过程，可简单分为如下三个步骤：**获取签名所需信息，拼接明文字符串，将明文字符串转化为签名。**

获取签名所需信息

生成签名所需信息包括项目 ID（App Id），空间名称（Bucket，文件资源的组织管理单元），项目的 Secret ID 和 Secret Key。获取这些信息的方法如下：
1. 登录腾讯云对象存储，进入对象存储空间；
2. 如开发者未创建空间，可添加空间，空间名称（Bucket）由用户自行输入；

3. 单击【获取 API 密钥】，获取 App Id，Secret ID 和 Secret Key。

拼接明文字符串 Original

明文字符串 Original 按照签名的类型可划分为“多次”和“单次”有效签名。
拼接多次有效签名串 multi_effect_signature：
Original = "a=[appid]&b=[bucket]&k=[SecretID]&e=[expiredTime]&t=[currentTime]&r=[rand]&f="
拼接单次有效签名串 once_signature：
Original = "a=[appid]&b=[bucket]&k=[SecretID]&e=[expiredTime]&t=[currentTime]&r=[rand]&f=[fileid]"
签名串中各字段含义如下：

字段	解释
a	开发者的项目 ID，接入对象存储服务创建空间时系统生成的唯一标识项目的 ID，即 App ID
b	空间名称 Bucket
k	项目的 Secret ID
t	当前时间戳，是一个符合 Unix Epoch 时间戳规范的数值，单位为秒
e	多次有效签名时，e 为签名的失效时刻，是一个符合Unix Epoch时间戳规范的数值，单位为秒。e 的计算方式为 e = t + 签名有效时长。签名有效时长最大取值为7776000（90天）；**单次签名时，e 必须设置为0**
r	随机串，无符号10进制整数，用户需自行生成，最长 10 位
f	fileid ，唯一标识存储资源的相对路径。格式为 /appid/bucketname/dirname/[filename]，**并且需要对其中非 '/' 字符进行 UrlEncode 编码**。 当操作对象为 文件夹时，filename 缺省。filename 中要包含文件后缀名。

注意

- 当签名类型需要绑定 fileid 时，f 字段请按要求赋值。
- 拼接单次有效签名的签名串时，**过期时间 e 必须设置为0**，以保证此签名只能针对固定资源使用一次。
- 拼接多次有效签名的签名串时，**过期时间 e 的单位为秒**，不同编程语言获得的系统 Unix 时间戳单位可能有所差异（例如 Java 是毫秒），但都请转化为秒。
- 多次/单次有效签名的具体适用场景参见本文章节**哪些场景需要用到签名?**

将明文字符串转化为签名

拼接好签名的明文字符串 Original 后，用已经获取的 SecretKey 对明文串进行 HMAC-SHA1 加密，得到 SignTmp：

SignTmp = HMAC-SHA1(SecretKey, **Original**)

将密文串 SignTmp 放在明文串 Origin 前面，拼接后进行 Base64Encode 算法，得到最终的签名 Sign：

Sign = Base64 (append(SignTmp, **Original**))

> ⚠ 注意
>
> - 此处使用的是标准的 Base64 编码，不是 urlsafe 的 Base64 编码。
> - 由于使用了 HMAC 算法，计算 SignTmp 的结果为二进制字符串，因此建议将算法写在同一函数中实现。单独输出 SignTmp 可能导致拼接后的字符串有误。

示例

本节介绍生成签名的算法实例，实例中使用 PHP 语言（若开发者使用其他开发语言，请使用对应的函数）。

获取签名所需信息

获取得到的签名所需信息如下

项目 ID：200001

空间名称 Bucket：newbucket

Secret ID：AKID**********

Secret Key：bLcPnl88WU

Secret ID：AKID****RhSePfPdOfSruK

拼接签名串

拼接的多次有效签名串 multi_effect_signature 如下：

a=200001&b=newbucket&k=AKID**********&e=1437995704&t=1437995644&r=2081660421&f=

拼接的单次有效签名串 once_signature 如下：

a=200001&b=newbucket&k=AKID**********&e=0&t=1437995645&r=1166710792&f=/200001/newbucket/tencent_test.jpg

```
$appid = "200001";
$bucket = "newbucket";
$secret_id = "AKID**********";
$secret_key = "bLcPnl88WU****RhSePfPdOfSruK";
$expired = time() + 60;
$onceExpired = 0;
$current = time();
$rdm = rand();
$fileid = "/200001/newbucket/tencent_test.jpg";
```

```
$multi_effect_signature =
'a='.$appid.'&b='.$bucket.'&k='.$secret_id.'&e='.$expired.'&t='.$current.'&r='.$rdm.'
&f=';

$once_signature=
'a='.$appid.'&b='.$bucket.'&k='.$secret_id.'&e='.$onceExpired.'&t='.$current.'&r='.$
rdm.'&f='.$fileid;
```

生成签名

```
$multi_effect_signature = base64_encode(hash_hmac('SHA1', $multi_effect_signature,
$secret_key, true).$multi_effect_signature);

$once_signature = base64_encode(hash_hmac('SHA1',$once_signature,$secret_key,
true).$once_signature);

echo $multi_effect_signature."\n";

echo $once_signature."\n";
```

- 最终得到的多次有效签名为：
  v6+um3VE3lxGz97PmnSg6+/V9PZhPTIwMDAwMSZiPW5ld2J1Y2tldCZrPUFLSURVZkxVRVVpZ1FpWHFtN0NWU3NwS0pudWFpSUt0eHFBdiZlPTE0NzA3MzcwMDAmdD0xNDcwNzM2OTQwJnI9NDkwMjU4OTQzJmY9
- 单次有效签名为：
  CkZ0/gWkHy3f76ER7k6yXgzq7w1hPTIwMDAwMSZiPW5ld2J1Y2tldCZrPUFLSURVZkxVRVVpZ1FpWHFtN0NWU3NwS0pudWFpSUt0eHFBdiZlPTAmdD0xNDcwNzM2OTQwJnI9NDkwMjU4OTQzJmY9LzIwMDAwMS9uZXdidWNrZXQvdGVuY2VudF90ZXN0LmpwZw==

使用示例

最近更新时间：2020-02-12 11:51:10

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

本篇示例将实现三个简单的对象存储接口：创建目录，简单上传文件，删除文件。

准备工作

在任何文件上传至 COS 之前，都需要创建 Bucket 来存放文件。
进入 COS 管理控制台，单击 **创建 Bucket**，弹出如下对话框：

单击 **创建**，即可在页面 Bucket 列表中看到创建的 Bucket：

使用 RESTful API

您至少需要一个支持发起 HTTP RESTful 请求的客户端，例如 curl 或其他相关函数来调用 API。

示例

说明：以华东地区的请求为例

创建目录

在 buckettest 创建一个 foldertest 目录。

请求

```
POST /files/v2/10000202/buckettest/foldertest/  HTTP/1.1
Host: sh.file.myqcloud.com
Content-Type: application/json
Content-Length: 15
Authorization:
1k2i3/EXIiTDirFg9DoKgWNHc4JhPTEwMDAwMjAyJms9QUtJRFBOUHVyNUIyN3FjdVJhakN
FbXpLVjkzVTdrOFZjZXFXJmU9MTQ2NTg3NTU0OSZ0PTE0NjU4NzUzNjkmcj03MTI5NDYy
MzQmZj0mYj1qb25ueHU1

{
   "op": "create"
}
```

返回

```
HTTP/1.1 200 OK
```

```
Server: nginx
Date: Tue, 14 Jun 2016 03:36:12 GMT
Content-Length: 109

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "ctime": "1465875372",
      "resource_path": "/10000202/buckettest/foldertest/"
   }
}
```

简单上传文件

在 buckettest 的 foldertest 目录下上传一个 test_upload.pptx 文件。

请求

```
POST /files/v2/10000202/buckettest/foldertest/test_upload.pptx HTTP/1.1
Host: sh.file.myqcloud.com
Accept: /
Authorization:
5iMIxdTSSxBLIkBTnUr+clUZcTZhPTEwMDAwMjAyJms9QUtJRFBOUHVyNUIyN3FjdVJhakN
FbXpLVjkzVTdrOFZjZXFXJmU9MTQ2NTg3NTU1MyZ0PTE0NjU4NzUzNzMmcj0yMDExOTA
xNjkwJmY9JmI9am9ubnh1NQ==
Content-Type: multipart/form-data; boundary=----------8d3944816ef2585
Content-Length: 78963

----------8d3944816ef2585

Content-Disposition: form-data; name="op"

upload

----------8d3944816ef2585

Content-Disposition: form-data; name="sha"

D80DFA67880831C3691AA1458589C6BED4423736

----------8d3944816ef2585

Content-Disposition: form-data; name="insertOnly"
```

```
0

----------------8d3944816ef2585

Content-Disposition: form-data; name="fileContent"; filename="test_upload.pptx"

Content-Type: application/octet-stream
```

返回

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 14 Jun 2016 03:36:13 GMT
Content-Type: /
Content-Length: 330

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "access_url": "http://buckettest-
10000202.file.myqcloud.com/foldertest/test_upload.pptx",
      "resource_path": "/foldertest/test_upload.pptx",
      "source_url": "http://buckettest-
10000202.cos.myqcloud.com/foldertest/test_upload.pptx",
      "url": "http://region.file.myqcloud.com/files/v2/foldertest/test_upload.pptx"
   }
}
```

删除文件

请求

```
POST /files/v2/10000202/buckettest/foldertest/test_upload.pptx  HTTP/1.1
Host: sh.file.myqcloud.com
Content-Type: application/json
Content-Length: 15
Authorization:
g4THlNbztAH/beX6Pmf00q8T++thPTEwMDAwMjAyJms9QUtJRFBOUHVyNUIyN3FjdVJhak
NFbXpLVjkzVTdrOFZjZXFXJmU9MCZ0PTE0NjU4NzU0MzEmcj0yNTk2NTAyNDkmZj0vMTA
wMDAyMDIvam9ubnh1NS9mb2xkZXJ0ZXN0L3Rlc3RfdXBsb2FkLnBwdHgmYj1qb25ueH
U1
```

```
{
   "op": "delete"
}
```

返回

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 14 Jun 2016 03:37:11 GMT
Content-Type: /
Content-Length: 30

{
   "code": 0,
   "message": "SUCCESS"
}
```

调用方式
请求结构

最近更新时间：2020-02-12 11:53:41

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

腾讯云对象存储的 RESTful API 请求结构如下表：

参数名称	必选	类型	描述
Version	是	String	HTTP 版本
URL	是	String	HTTP 请求地址，总长度不得超过 1024 字节
Header	是	数据集合	HTTP 请求头部
HTTP Method	是	String	HTTP 请求方法，常用 GET 或 POST
Request Body	否	json	HTTP 请求正文

URL 请求地址格式举例：

```
<Region>.file.myqcloud.com/files/v2/<appid>/<bucket_name>
[/dir_name/<file_name>]
```

说明：
Region：所在地域，枚举值为 可用地域 中适用于 JSON API 的地域简称，例如 sh、gz、sgp 等。
appid：数字，在腾讯云对象存储 Web 控制台中，创建项目后得到 appid，用于唯一标识某个项目。
bucket_name：字母和数字的组合串。对应某个项目下的 bucket 名称。
<箭头括号>表示必须 替换为有效值的变量，[英文方括号]表示可选的命令或参数，具体要求以各 API 文档为准。
Header 结构：

参数名称	必选	类型	描述
Host	是	Stri	文件云服务器域名，形式为 .file.myqcloud.com ，其中 Region 为 可用地域 中适用于 JSON API 的地域简称，例如：sh.file.myqcloud.com ，sgp.file.myqcloud.com 等

		ng	
Authorization	是	String	用户的有效签名，用于鉴权。对应签名鉴权中得到的 sign 字符串
Content-Type	是	String	按要求填 application/json 或 multipart/form-data
Content-Length	是	Int	请求长度，单位：字节

示例

请求

```
POST /files/v2/10000202/buckettest/foldertest/  HTTP/1.1
Host: sh.file.myqcloud.com
Content-Type: application/json
Content-Length: 15
Authorization:
1k2i3/EXIiTDirFg9DoKgWNHc4JhPTEwMDAwMjAyJms9QUtJRFBOUHVyNUIyN3FjdVJhakN
FbXpLVjkzVTdrOFZjZXFXJmU9MTQ2NTg3NTU0OSZ0PTE0NjU4NzUzNjkmcj03MTI5NDYy
MzQmZj0mYj1qb25ueHU1

{
  "op": "create"
}
```

返回结构

最近更新时间：2020-02-12 11:54:18

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

腾讯云对象存储的 RESTful API 统一返回 HTTP 标准的状态码，HTTP 标准的头部信息以及 JSON 格式的返回包内容。
常见的返回格式如下：

参数名称	类型	描述
HTTP Status Code	Int	HTTP 状态码
HTTP Headers	String	HTTP 头部信息
Response Body	数据集合	HTTP 响应正文

Response Body 数据集合

参数名称	类型	描述
code	Int	服务端返回码
message	String	服务端提示内容
data	数据集合	服务端返回的应答数据

示例

返回

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 14 Jun 2016 03:36:12 GMT
Content-Length: 109

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
```

```
    }
}
```

错误码

最近更新时间：2020-02-20 14:59:57

> **注意**
> 目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

服务端 SDK 错误码

-1 指定的上传文件不存在
-2 网络请求失败（一般是用户主机和我们的服务无法连通）
-3 参数错误
-4 分片大小非法

下载状态码

-200 正常
-206（部分内容）服务器成功处理了部分 GET 请求
-403 鉴权错误或者命中了 refer 黑名单
-404 文件不存在
-405 请求方法不能被用于请求相应的资源
-500 内部服务器错误
-6101 文件不存在
-45062 文件未上传完成

后端 HTTP 返回码

400 HTTP_BAD_REQUEST 请求不合法，包体格式错误
401 HTTP_UNAUTHORIZED 权限验证失败
403 HTTP_FORBIDDEN 鉴权信息不合法，禁止访问
413 HTTP_REQUEST_ENTITY_TOO_LARGE 文件大小超长
500 HTTP_INTERNAL_SERVER_ERROR 服务内部错误
503 HTTP_SERVICE_UNAVAILABLE 服务不可用
504 HTTP_GATEWAY_TIME_OUT 后端服务超时
599 HTTP_UNKOWN_INTERNAL_SERVER_ERROR 未知服务器内部错误
612 HTTP_FILE_NOTEXIST 查无此文件
631 HTTP_INVALID_APPID 非法的业务 ID

业务返回码

_SUCCESS 0 成功
-9999 UNKOWN 未知错误

PROXY 错误码

-99 ERROR_PROXY_GET_CONFIG proxy读取配置失败
-98 ERROR_PROXY_AUTH_INVOKE 调用签名服务失败
-97 ERROR_PROXY_AUTH_FAILED 非法签名
-96 ERROR_PROXY_AUTH_EXPIRED 签名过期
-95 ERROR_PROXY_NO_SESSION 消息缺少session信息
-94 ERROR_PROXY_WRONG_SESSION 携带错误session信息
-89 ERROR_PROXY_ROUTE_CMD proxy转发cmd服务失败
-88 ERROR_PROXY_ENCODE_CMD 编码cmd服务消息失败
-87 ERROR_PROXY_DECODE_CMD 解析 cmd 服务消息失败
-86 ERROR_PROXY_ROUTE_PROCESS proxy 转发 process服务失败
-85 ERROR_PROXY_DECODE_PROCESS_RSP 解析 process 服务应答失败
-84 ERROR_PROXY_GET_PROCESS_L5 获取 processL5 失败
-83 ERROR_PROXY_AUTH_UNPACK 签名服务解包失败
-82 ERROR_PROXY_AUTH_APPID_NOEXIST 不存在此 APPID
-81 ERROR_PROXY_AUTH_SIGN_EMPTY 签名为空
-80 ERROR_PROXY_INVALID_APPID 非法的业务 ID
-79 ERROR_PROXY_AUTH_SECRETID_NOEXIST secretid 不存在
-78 ERROR_PROXY_INVALID_PROTOCOL SDK协议不匹配，请升级
-77 ERROR_PROXY_AUTH_REPLAY_ATTACH 单次性签名已不可用
-76 ERROR_PROXY_AUTH_ONCESIGN_NOFILEID 单次签名没有url
-75 ERROR_PROXY_AUTH_UNSUPPORTED_OP 不支持此操作
-74 ERROR_PROXY_AUTH_MULTISIGN_EXPIREDEMPTY 多次签名-过期时间为0
-73 ERROR_PROXY_AUTH_ONCESIGN_EXPIREDNOTEMPTY 单次签名-过期时间不为0
-72 ERROR_PROXY_APP_SIGN_FAILED 签名失败
-71 ERROR_PROXY_FREQ_CONTROL 操作太频繁,请稍后再试
-70 ERROR_PROXY_APPID_USERID_NOTMATCH 输入的appid与签名不匹配
-69 ERROR_PROXY_INVALID_PARAM_DOWNLOAD_URL_EMPTY 输入参数错误:download_urlempty
-68 ERROR_PROXY_ENCODE ip直通车打包失败
-67 ERROR_PROXY_DECODE ip直通车解包失败
-66 ERROR_PROXY_NETWORK 网络请求失败
-65 ERROR_PROXY_CONTINUE_TRANSFER_WITH_DATA 断点续传不支持携带数据包
-64 ERROR_PROXY_FORBID_APPID 该业务已经被屏蔽
-63 ERROR_PROXY_FILEID_NOTMATCH fileid与签名不匹配
-62 ERROR_PROXY_AUTH_APPID appid与secretid不匹配
-61 ERROR_PROXY_SIGN_BUCKET_NOTMATCH bucket与签名中的bucket不匹配
-61 ERROR_PROXY_INVALID_CMD_REQ 非法的CMD请求
-60 ERROR_PROXY_AUTH_INVALID_TOKEN 非法token值

CMD 错误码

-199 ERROR_CMD_FILE_MOVE_ERROR 文件移动失败
-198 ERROR_CMD_REDIRECT_ERROR 重定向错误
-197 ERROR_CMD_FILE_NOTEXIST 查无此文件
-196 ERROR_CMD_BACKEND_NETWORK 网络请求失败
-195 ERROR_CMD_BACKEND_RSP_DECODE 后端解包失败
-194 ERROR_CMD_BACKEND_REQ_ENCODE 后端打包失败
-193 ERROR_CMD_RSP_ENCODE 返回包打包失败
-192 ERROR_CMD_REQ_DECODE 请求包解析失败
-191 ERROR_CMD_URL_PARAMS_NOTMATCH url参数解析不匹配
-190 ERROR_CMD_FILE_DEL_ERROR 文件删除失败
-189 ERROR_CMD_INVALID_PARAM_DOWNLOAD_URL_EMPTY 输入参数错误:download_urlempty
-187 ERROR_CMD_URL_PARSE_ERROR 从url中解析参数失败
-186 ERROR_CMD_VIDEO_MOVE 暂不支持视频文件复制
-185 ERROR_CMD_PRE_PROCESS 业务预处理失败
-184 ERROR_CMD_POST_PROCESS 业务后处理失败
-183 ERROR_CMD_GET_ROUTE 获取路由失败
-182 ERROR_CMD_PARAM_INVALID 参数检验失败
-181 ERROR_CMD_COS_ERROR 存储后端错误
-180 ERROR_CMD_COS_INVALID_PATH 非法路径
-179 ERROR_CMD_COS_INVALID_MOD_FLAG 修改标志非法
-178 ERROR_CMD_COS_PATH_CONFLICT 路径冲突
-177 ERROR_CMD_COS_FILE_EXIST 文件已存在
-176 ERROR_CMD_COS_BUCKET_EXIST Bucket已存在
-175 ERROR_CMD_COS_FILE_SIZE_NOT_EQU 文件大小不一致
-174 ERROR_CMD_COS_SHA_NOT_EQU 文件SHA不一致
-173 ERROR_CMD_COS_DIR_NOT_EMPTY 目录非空
-172 ERROR_CMD_COS_BUCKET_NUM_LIMIT Bucket数量限制
-171 ERROR_CMD_COS_INVALID_APP_TYPE AppType非法
-170 ERROR_CMD_COS_INVALID_LIST_NUM LIST数量非法
-169 ERROR_CMD_COS_INVALID_QUERY_FLAG 查询标记非法
-168 ERROR_CMD_COS_INVALID_SHA 非法SHA
-167 ERROR_CMD_COS_INVALID_CNAME 非法的CNAME
-166 ERROR_CMD_COS_INDEX_ERROR 索引不存在
-165 ERROR_CMD_COS_MAX_NUM 单次拉取目录最大支持199
-164 ERROR_CMD_PARAM_PATH_EMPTY path为空
-163 ERROR_CMD_PARAM_PATH_ELEMENT_MISSING path字段缺少元素
-162 ERROR_CMD_PARAM_PATH_ELEMENT_NOT_MATCH path字段元素不匹配

-161 ERROR_CMD_PARAM_PATH_DIR_MISSING_SLASH 目录path字段缺少slash结尾
-160 ERROR_CMD_PARAM_PAHT_FILE_SURPLUS_SLASH 文件path多余slash结尾
-159 ERROR_CMD_PARAM_CNAME_ERROR CNAME 数量必须小于等于10
-158 ERROR_CMD_PARAM_AUTH_TYPE 鉴权类别错误//鉴权类别错误
-157 ERROR_CMD_PARAM_MIN_NUM 单次拉取条目最小为1//DirListReq中拉取数据条数
-156 ERROR_CMD_INVALID_VIDEO_NAME 非法的视频名
-155 ERROR_CMD_INVALID_FORMAT 非法视频FORMAT
-154 ERR_CMD_APP_VIDEO_NOMATCH 文件BUCKET中设置视频参数
-153 ERR_CMD_INVALID_VIDEO_APP 非法的视频APP
-152 ERR_CMD_COS_TIMEOUT 存储后端超时
-151 ERROR_CMD_PARAM_REFER_ERROR refer数量必须小于等于10
-150 ERROR_CMD_PARAM_SOURCE_DOMAIN_EMPTY 创建bucket时设置的回源路径不能为空
-149 ERROR_CMD_INVALID_REFER refer中含有非法字符
-148 ERROR_CMD_INVALID_CNAME CNAME中含有非法字符
-147 ERROR_CMD_NOT_UTF8_ENCODING 参数含非utf8编码
-146 ERROR_CMD_INVALID_HOST_NAME 回源地址不合法
-145 ERROR_CMD_INVALID_COVER 是否需要封面参数不合法
-144 ERROR_CMD_INDEX_NO_EXIST 索引不存在
-143 ERROR_CMD_INDEX_ERROR 索引错误

文件缓存的错误码

-300 ERROR_PROCESS_OVERLOAD 服务过载
-299 ERROR_PROCESS_UNKNOWN_CMD 命令字未知
-298 ERROR_PROCESS_REQ2MSG 解包失败
-297 ERROR_PROCESS_FRAME 框架handleprocess错误
-296 2RSP 打包失败
-295 ERROR_PROCESS_WRITEFILE 文件数据异常
-294 ERROR_PROCESS_READFILE 文件数据异常
-293 ERROR_PROCESS_NOTIFY 通知插件失败
-292 ERROR_PROCESS_CREATEFILE 文件缓存服务器错误
-291 ERROR_PROCESS_ENCODE_SESSION 编码session失败
-290 ERROR_PROCESS_OFFSET 无效的session
-289 ERROR_PROCESS_DENY 插件拒绝上传
-288 ERROR_PROCESS_ENCODE process打包失败
-287 ERROR_PROCESS_DECODE process解包失败
-286 ERROR_PROCESS_DECODE_SESSION 解码session失败
-285 ERROR_PROCESS_FILE_TOO_LARGE 文件过大
-284 ERROR_PROCESS_WRONG_SLICE_SIZE 分片大小不一致
-283 ERROR_PROCESS_SLICE_TOO_SMALL 分片过小

-282 ERROR_PROCESS_ANALYZE 图片分析失败

文件插件错误码

-4000 ErrDecodeReq 请求体解包失败
-4001 ErrEncodeRsp 回包打包失败
-4002 ErrParamInvalid 请求参数错误
-4003 ErrNoSetErrCode 插件处理错误
-4004 ErrCosRspHeadCode 入库出错
-4005 ErrCosRspInvalidOffset 入库返回非法偏移
-4006 ErrCosRspInvalidHeadFlag 入库返回非法的头部标记
-4007 ErrTmemRspSsdError 获取TMEM配置错误
-4008 ErrParserShmError 解析共享内存TMEM配置错误
-4010 ErrCreateControlFile 创建控制文件错误
-4011 ErrReadControlFile 读取控制文件错误
-4012 ErrGetInputData 获取入库数据失败
-4013 ErrDecodeSession 解session失败
-4014 ErrQueryOffset 查询分片错误
-4015 ErrTheSameOffset 同一个offset多次返回
-4016 ErrNotSameSHA sha校验失败
-4017 ErrHttpSvr 文件入库错误
-4018 ErrSameFileUpload 相同文件已上传过
-4019 ErrOffsetToZero 入库失败，请重新上传
-4020 ErrParamInvalidCheckTypeShouldBeSHA 校验类型需要SHA
-4021 ErrParamInvalidSessionDecodeError 参数session解码失败
-4022 ErrParamInvalidFilePathInvalid 获取bucket路径失败
-4023 ErrParamInvalidCheckSumSizeInvalid 校验和大小错误
-4024 ErrOffGoBack 偏移量回退，请重试
-4025 ErrHttpTimeout 文件入库超时
-4026 ErrOffsetBackTooMuch 索引被删除或SHA不一致，请重新上传
-4027 ErrOffsetRedo 索引错误，请重新上传

CGI 错误码

-5999 ERROR_CGI_PARAM 参数错误
-5998 ERROR_CGI_AUTHORIZATION 签名格式错误
-5997 ERROR_CGI_NETWORK 后端网络错误
-5996 ERROR_CGI_REQ_METHOD HTTP请求方法错误
-5995 ERROR_CGI_FILESIZE 文件大小错误
-5994 ERROR_CGI_URL_NOTMATCH url参数解析不匹配'
-5993 ERROR_CGI_FORM_ERR multipart/formdata参数错误

-5992 ERROR_CGI_REQ_PARAM 请求参数错误
-5991 ERROR_CGI_SLICE 分片过大
-5990 ERROR_CGI_MISSING_FILECONTENT 找不到 filecontent
-5989 ERROR_CGI_UPLOAD_FAIL 上传失败
-5988 ERROR_CGI_INIT cgi 初始化错误
-5987 ERROR_CGI_WUP_ENCODE wup 编码失败
-5986 ERROR_CGI_WUP_DECODE wup 解码失败
-5985 ERROR_CGI_NETWORK_L5 获取路由失败
-5984 ERROR_CGI_SHA_NOT_MATCH sha1 不匹配
-5983 ERROR_CGI_WRONG_SESSION 错误的 session
-5982 ERROR_CGI_SOCK_CREATE 建立连接错误
-5981 ERROR_CGI_SOCK_CONNECT 建立连接错误
-5980 ERROR_CGI_PARAM_MISSING_OP 缺少参数:op
-5979 ERROR_CGI_PARAM_MISSING_SHA 缺少参数:sha
-5978 ERROR_CGI_PDU_ENCODE pdu编码失败
-5977 ERROR_CGI_PDU_DECODE pdu解码失败
-5976 ERROR_CGI_REQ_PARAM_FLAG_NOTMATCH bucketflag与实际参数不一致
-5975 ERROR_CGI_JSON_PARSE_ERROR json解析失败
-5974 ERROR_CGI_VIDEO_FORMAT 视频码率错误
-5973 ERROR_CGI_CROSS_BUCKET_COPY 不允许跨存储空间复制
-5972 ERROR_CGI_UNSUPPORTED_METHOD 此方法不支持于当前版本
-5971 ERROR_CGI_PARAM_MISSING_FILESIZE 缺少参数：filesize

OSS 错误码

-6999 ERROR_OSS_UNKNOWN 未知错误
-6998 ERROR_OSS_INVALID_PROTOCOL 协议错误
-6997 ERROR_OSS_INVALID_PARAM 参数错误
-6996 ERROR_OSS_FILESIZE 文件过大
-6995 ERROR_OSS_MKDIR 创建目录失败
-6994 ERROR_OSS_GZIP gzip 压缩失败
-6993 ERROR_OSS_WRITE_FILE 写文件失败
-6992 ERROR_OSS_FILE_NOTEXIST 文件不存在
-6991 ERROR_OSS_READ_FILE 读文件失败

目录接口
创建目录

最近更新时间：2020-02-12 11:55:00

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

功能描述

创建目录 API 用于在 COS 的 Bucket 中创建一个新目录。成功创建新目录的前提条件是已经在控制台创建了 Bucket 。如果该 COS 中没有 Bucket 或新建目录名已存在，则创建新目录不成功。

请求

语法示例：

```
POST /files/v2/<APPID>/<BucketName>/<DirName>/ HTTP/1.1
Host: <Region>.file.myqcloud.com
Authorization: <MultiEffectSignature>
Content-Type: application/json
Content-Length: <ContentLength>
```

> **说明**
> - Authorization: <MultiEffectSignature> 多次有效签名（详细参见 签名算法 章节）。
> - Content-Length: <ContentLength> RFC 2616 中定义的 HTTP 请求内容长度（字节）。

请求体

该 API 接口请求的请求体具体节点内容为：

```
{
  "op":"create",
  "biz_attr":""
}
```

具体内容描述如下：

参数名称	描述	类型	必选

op	操作类型，填”create”	String	是
biz_attr	COS 服务调用方自定义属性，可通过 查询目录属性 获取该属性值	String	否

响应

响应体

该响应体返回为 application/json 数据，包含完整节点数据的内容展示如下：

```
{
  "code":0,
  "message":"SUCCESS",
  "request_id":"NTk5YWM5MDZfNjIyNWI2NF8xOWQxXzE0NWY4",
  "data":{
  "ctime":1503316230
  }
}
```

具体的参数描述如下：

参数名称	描述	类型
code	服务端返回码，如果没有发生任何错误取值为0；如果发生错误该参数指称具体的错误码。COS 服务相关的错误码请参见 错误码	Number
message	服务端提示内容，如果发生错误该字段将详细描述发生错误的情况。	String
data	服务端返回的应答数据，该内容代表了接口返回的具体的业务数据。	Object
request_id	该请求的唯一标识 ID	String

data 数据集参数描述：

参数名称	描述	类型
ctime	创建时间，10 位 Unix 时间戳（UNIX 时间是从协调世界时 1970 年 1 月 1 日 0 时 0 分 0 秒起的总秒数）	Number

实际案例

请求

```
POST /files/v2/1252448703/lewzylu02/testfolder/ HTTP/1.1
Host: gz.file.myqcloud.com
Authorization:
2jDzkKTIf0Dsk1OzTYIIK+ImIcNhPTEyNTI0NDg3MDMmaz1BS0lEMTVJc3NraUJRS1RaYkFv
NIdoZ2NCcVZsczlTbXVHMDAmZT0xNTAzMzE2MzY0JnQ9MTUwMzMxNjE4NCZyPTE3OD
gwJmY9JmI9bGV3enlsdTAy
Content-Type: application/json
Content-Length: 29

{"op":"create","biz_attr":""}
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 110
Connection: keep-alive
Date: Mon, 21 Aug 2017 11:50:30 GMT
Server: tencent-cos
x-cos-request-id: NTk5YWM5MDZfNjIyNWI2NF8xOWQxXzE0NWY4

{"code":0,"message":"SUCCESS","request_id":"NTk5YWM5MDZfNjIyNWI2NF8xOWQxX
zE0NWY4","data":{"ctime":1503316230}}
```

查询目录属性

最近更新时间：2020-02-12 11:55:33

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 查询目录的属性信息，包括：目录名、目录属性、创建时间、修改时间。
前置条件：该目录已存在。

请求

请求语法

```
GET /files/v2/<appid>/<bucket_name>/<dir_name>/?op=stat HTTP/1.1
Host: <Region>.file.myqcloud.com
Authorization: <multi_effect_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

data 数据集合

参数名称	必选	类型	描述
biz_attr	是	String	目录属性，业务端维护
ctime	是	String	创建时间，10 位 Unix 时间戳

mtime	是	String	修改时间，10 位 Unix 时间戳

示例

请求

```
GET /files/v2/10055004/accesslog/testfolder/?op=stat HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
Lb6JYSxu9GXfRGQybwu2ofs1kOlhPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFB
a1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjYzOTI1MSZ0PTE0NzI2MzkwNzEmcj0xMzc2N
zQ0MzAmZj0mYj1hY2Nlc3Nsb2c=
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 91
Date: Wed Aug 31 18:24:31 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "biz_attr": "",
      "ctime": 1472639071,
      "mtime": 1472639071
   }
}
```

列出目录

最近更新时间：2020-02-12 11:56:18

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 列出指定目录下的所有文件目录，可带前缀查询，只支持向后翻页。
前置条件：目录已存在。

> **说明**
> 在返回内容中不包含 biz_attr 字段，如需查询，请调用查询目录属性接口。

请求

请求语法

```
GET <Request_URL>?op=list&num=<Int64>  HTTP/1.1
Host: <Region>.file.myqcloud.com
Authorization: <multi_effect_signature>
```

请求地址 Request_URL 示例
直接查询 bucket 目录下所有文件和目录：
`sh.file.myqcloud.com/files/v2/<appid>/<bucket_name>/`
查询指定 bucket 中文件夹 dir_name 下的文件和目录：
`sh.file.myqcloud.com/files/v2/<appid>/<bucket_name>/<dir_name>/`
带前缀的查询：
`sh.file.myqcloud.com/files/v2/<appid>/<bucket_name>/<dir_name>/[prefix]`

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

对于Request_URL，带前缀查询时，[prefix] 末尾无 ‘ / ’ 。

请求参数

参数名称	必选	类型	描述

op	是	String	操作类型，填 ”list”
num	是	Int	本次查询所要列出的目录总数,最大支持1000
context	否	String	透传字段，从响应的返回内容中得到。若查看第一页，则将空字符串作为 context 传入。若需要翻页，需要将前一页返回内容中的 context 透传到参数中。

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

其中的 data 数据集合：

参数名称	必选	类型	描述
context	是	String	透传字段，用于翻页，业务端不需理解，需要往后翻页则透传给腾讯云
listover	是	Bool	是否有内容可以继续往后翻页
infos	是	数据集合	文件和文件夹列表，若当前目录下不存在文件或文件夹，则该返回值可能为空

其中的 infos 数据集合：

参数名称	必选	类型	描述
name	是	String	文件名
filesize	否	Int	文件大小，当类型为文件时返回
filelen	否	Int	文件已传输大小，当类型为文件时返回

sha	否	String	文件 sha，当类型为文件时返回
ctime	是	Number	创建时间，10位 Unix 时间戳
mtime	是	Number	修改时间，10位 Unix 时间戳
access_url	否	String	生成的资源可访问的cdn url，当类型为文件时返回
authority	否	String	如果没有对文件单独设置该属性，则可能不会返回该字段。返回值：eInvalid（表示继承 bucket 的读写权限）；eWRPrivate（私有读写）；eWPrivateRPublic（公有读私有写）。说明：文件可以和 bucket 拥有不同的权限类型，已经设置过权限的文件如果想要撤销，将会直接被赋值为 eInvalid，即继承 bucket 的权限
source_url	否	String	生成的资源可访问的源站 url，当类型为文件时返回

示例

请求

```
GET /files/v2/10055004/accesslog/aaa/bbb/?op=list&order=0&num=5 HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
F/mImHxxGlI5A7lLYnGa8aBtK6thPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFBa
1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0MDIwNiZ0PTE0NzI2NDAwMjYmcj03OTUxNz
kyMCZmPSZiPWFjY2Vzc2xvZw==
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 2502
Date: Wed Aug 31 18:40:26 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "context": "/accesslog/aaa/bbb/14709104920.89088300.txt",
```

```
    "infos": [
        {
            "access_url": "http://accesslog-
10055004.file.myqcloud.com/aaa/bbb/14709104920.63503400.txt",
            "authority": "eInvalid",
            "ctime": 1470910492,
            "filelen": 10264,
            "filesize": 10264,
            "mtime": 1470910492,
            "name": "14709104920.63503400.txt",
            "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/aaa/bbb/14709104920.63503400.txt?
cmd=txt_preview",
            "sha": "cad7d66231fa548b2e5b5f431fcf8f1bf028d0b0",
            "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/aaa/bbb/14709104920.63503400.txt"
        },
        {
            "access_url": "http://accesslog-
10055004.file.myqcloud.com/aaa/bbb/14709104920.71280700.txt",
            "authority": "eInvalid",
            "ctime": 1470910492,
            "filelen": 10264,
            "filesize": 10264,
            "mtime": 1470910492,
            "name": "14709104920.71280700.txt",
            "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/aaa/bbb/14709104920.71280700.txt?
cmd=txt_preview",
            "sha": "4ba80f5c809954e93b97de3a3bb0950f6374bdf7",
            "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/aaa/bbb/14709104920.71280700.txt"
        },
        {
            "access_url": "http://accesslog-
10055004.file.myqcloud.com/aaa/bbb/14709104920.77835100.txt",
            "authority": "eInvalid",
            "ctime": 1470910492,
            "filelen": 10264,
            "filesize": 10264,
            "mtime": 1470910492,
            "name": "14709104920.77835100.txt",
            "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/aaa/bbb/14709104920.77835100.txt?
cmd=txt_preview",
```

```
            "sha": "2130bd5f70e0da1a50e9ad70907f5d694ad88bb4",
            "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/aaa/bbb/14709104920.77835100.txt"
        },
        {
            "access_url": "http://accesslog-
10055004.file.myqcloud.com/aaa/bbb/14709104920.83866400.txt",
            "authority": "eInvalid",
            "ctime": 1470910492,
            "filelen": 10264,
            "filesize": 10264,
            "mtime": 1470910492,
            "name": "14709104920.83866400.txt",
            "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/aaa/bbb/14709104920.83866400.txt?
cmd=txt_preview",
            "sha": "1f47ec790b23c950869449309797737e39f1cb92",
            "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/aaa/bbb/14709104920.83866400.txt"
        },
        {
            "access_url": "http://accesslog-
10055004.file.myqcloud.com/aaa/bbb/14709104920.89088300.txt",
            "authority": "eInvalid",
            "ctime": 1470910492,
            "filelen": 10264,
            "filesize": 10264,
            "mtime": 1470910493,
            "name": "14709104920.89088300.txt",
            "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/aaa/bbb/14709104920.89088300.txt?
cmd=txt_preview",
            "sha": "afe17bb32c0835e8b9ce69450f95e3e6f2af4606",
            "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/aaa/bbb/14709104920.89088300.txt"
        }
    ],
    "listover": false
  }
}
```

删除目录

最近更新时间：2023-09-05 17:23:02

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 删除指定目录。
前置条件：必须已清空该目录下的所有文件或文件夹。
限制条件：目标目录不可填写 `/` ，无法删除根目录，即无法删除 Bucket。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>/<dir_name>/ HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: application/json
Authorization: <once_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填 ” delete”

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容

示例

请求

```
POST /files/v2/10055004/accesslog/testfolder/ HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
JYv1bYoo9SdiJSBMxZYveGsB001hPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFBa
1JCQzJEUlhCdFBkN0NybEpRJmU9MCZ0PTE0NzI2MzkwNzEmcj03ODgwMTAxNTQmZj0v
MTAwNTUwMDQvYWNjZXNzbG9nL3Rlc3Rmb2xkZXIvJmI9YWNjZXNzbG9n
Content-Type: application/json
Content-Length: 15

{
   "op": "delete"
}
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 30
Date: Wed Aug 31 18:24:31 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS"
}
```

文件接口
简单上传文件

最近更新时间：2020-02-12 11:24:04

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 上传 ≤ 20MB 的文件（超过20M 的文件，请使用“分片上传文件” API）。
请求地址中的参数即指明，将名称为 file_name 的文件存储于路径bucket_name/[dir_name] 下，其中，bucket_name 必填，dir_name 可缺省。
前置条件：指定的文件目录已存在。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>[/dir_name]/<file_name> HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <multi_effect_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填 “upload”
filecontent	是	Binary	文件内容
sha	否	String	文件的 SHA-1 校验码
biz_attr	否	String	文件属性，业务端维护

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

data 数据集合：

参数名称	必选	类型	描述
access_url	是	String	通过 CDN 访问该文件的资源链接（访问速度更快）
resource_path	是	String	该文件在 COS 中的相对路径名，可作为其 ID 标识。格式 /appid/bucket/filename。推荐业务端存储 resource_path，然后根据业务需求灵活拼接资源 url（通过 CDN 访问 COS 资源的 url 和直接访问 COS 资源的 url 不同）。
source_url	是	String	（不通过 CDN）直接访问 COS 的资源链接
url	是	String	操作文件的 url 。业务端可以将该 url 作为请求地址来进一步操作文件，对应 API：文件属性、更新文件、删除文件、移动文件中的请求地址。

说明

腾讯云 COS 会默认为每个资源生成经 CDN 的访问链接 access_url，当业务端尚未开通 CDN 时，仍然可以获得该链接，但是无法访问。

示例

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
```

```
Authorization:
VKQg5H9I/QMj/tzkk2M3JOljFr9hPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFBa1J
CQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0NjAzOCZ0PTE0NzI2NDU4NTgmcj0xMjUyMDk
1MjM5JmY9JmI9YWNjZXNzbG9n
Content-Length: 577
Content-Type: multipart/form-data; boundary=--------------------------aa502a40917c

----------------------------aa502a40917c
Content-Disposition: form-data; name="op"

upload
----------------------------aa502a40917c
Content-Disposition: form-data; name="sha"

1fe86826617c3a32da4f06bccc71a2a718b274c6
----------------------------aa502a40917c
Content-Disposition: form-data; name="biz_attr"



----------------------------aa502a40917c
Content-Disposition: form-data; name="filecontent"; filename="/data/liudg/cos-php-
sdk/111.txt"
Content-Type: application/octet-stream

hello, I am 111.txt.

----------------------------aa502a40917c--
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 434
Date: Wed Aug 31 20:17:38 2016
Server: tencent-cos

{
    "code": 0,
    "message": "SUCCESS",
    "data": {
        "access_url": "http://accesslog-10055004.file.myqcloud.com/testfolder/111.txt",
        "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/testfolder/111.txt?cmd=txt_preview",
```

```
        "resource_path": "/10055004/accesslog/testfolder/111.txt",
        "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/testfolder/111.txt",
        "url":
"http://sh.file.myqcloud.com/files/v2/10055004/accesslog/testfolder/111.txt"
    }
}
```

初始化分片上传

最近更新时间：2020-02-13 16:47:44

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

功能描述

使用 API 对大于20MB 的文件进行分片上传之前，需要先使用本 API 创建「分片上传会话」，即获取此次上传的 session。使用该接口前请确认对应上传目录已存在，如果 Bucket 中没有目录则请求不成功。

操作流程

第一步：发送「初始化分片上传」请求。
第二步：腾讯云会判断文件的上传状态，若未完成上传，会返回 -4019 错误，用户需调用查询上传分片接口查询已经上传完的分片，并进行断点续传的操作；若文件从未传输过，则返回用户上传的 session。
第三步：设置 session 和 offset 参数，构造「逐个分片上传」请求，逐片上传后续文件。
第四步：循环执行第三步直到文件的分片数据上传完成。
第五步： 调用 finish 接口结束分片上传。

请求

语法示例：

```
POST /files/v2/124566667/xy2/uploader1500000000 HTTP/1.1
Host: <Region>.file.myqcloud.com
Authorization: <MultiEffectSignature>
Content-Type: multipart/form-data
Content-Length: <ContentLength>
```

> Authorization：多次有效签名（详细请参阅 签名算法 章节）。
> Content-Length：RFC 2616 中定义的 HTTP 请求内容长度（字节）。

请求参数

该请求的请求体为空。

请求体

该请求的请求体如下：

```
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="op"
Content-Length: 17

upload_slice_init
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="sha"
Content-Length: 40

1
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="slice_size"
Content-Length: 7

1048576
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="filesize"
Content-Length: 5

26585
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="uploadparts"
Content-Length: 83

[{"offset":0,"datalen":26585,"datasha":"e98c925a97a8ae4ffec264ae674e28b8f975eba3"}]
--9fd48fba-1ea2-4789-8e6e-64c40c243480--
```

节点参数具体描述如下：

参数名称	描述	类型	必选
op	操作类型，填 “upload_slice_init”	String	是
sha	文件的 SHA-1 校验码， 如果需要做上传校验，则带上该参数。（您需要另外计算文件的 SHA-1校验码 ）	String	否
slice_size	分片大小，单位为 Byte，有效值：1048576 (1MB)	Number	是
filesize	文件总大小，单位为 Byte	Number	是

biz_attr	COS 服务调用方自定义属性，可通过 查询目录属性 获取该属性值	String	否
uploadparts	内容为 UploadPart 结构的 JSON 数据的数组。如果指定了 sha 参数，该参数也必须携带。UploadPart 结构： offset 为该数据块的距离文件头部的偏移量， datalen 为该数据块的长度， datasha 为**上一个分片数据的中间 sha 与本分片数据结合，计算的未经过 final 的中间 sha 摘要值**校验码	String	否

响应

响应体

该响应体返回为 **application/json** 数据，包含完整节点数据的内容展示如下：

```
{
  "code":0,
   "message":"SUCCESS",
   "request_id":"NTliNzhlOGRfMTliMjk0MGFDg0N1diY2E=",
   "data":{
   "session":"8ad1d23f1b1f3452a1bda9bfbf5f810a3dac6a73371c632e47baf12d",
   "slice_size":1048576}
}
```

具体的参数描述如下：

参数名称	描述	类型
code	服务端返回码，如果没有发生任何错误取值为0；如果发生错误该参数指称具体的错误码，COS服务相关的错误码可以查看 COS 错误码	Number
message	服务端提示内容，如果发生错误该字段将详细描述发生错误的情况。	String
request_id	该请求的唯一标识 ID	String
data	服务端返回的应答数据，该内容代表了接口返回的具体的业务数据	Object

data 数据集参数描述：

参数名	描述	类型

称		
access_url	通过 CDN 访问该文件的资源链接（访问速度更快）	String
resource_path	该文件在 COS 中的相对路径名，可作为其 ID 标识。格式'/appid/bucket/filename'。推荐业务端存储 resource_path，然后根据业务需求灵活拼接资源 url（通过 CDN 访问 COS 资源的 url 和直接访问 COS 资源的 url 不同）。	String
source_url	（不通过 CDN ）直接访问 COS 的资源链接	String
vid	当业务上传成功并触发了其他异步操作，该字段表示该异步操作的唯一标记。可调用其他接口来查询该异步任务的状态	String

说明

腾讯云对象存储会默认为每个资源生成经 CDN 的访问链接 access_url，当业务端尚未开通 CDN 时，仍然可以获得该链接，但是无法访问。

实际案例

请求

```
POST /files/v2/124566667/xy2/uploader1500000000  HTTP/1.1
Host: gz.file.myqcloud.com
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: cos-python-sdk-v4
Authorization:
irCtRo2BYVMOxzSooigDrmtJ4G9hPTEyNTE2Njg1Nzcmaz1BS0lEem9RbXNrVG9oUHVJVV
JUYW5uWDBEQXNLWlIlcWxDelomZT0xNTA1MjAyMTA1JnQ9MTUwNTIwMTgwNSZy8xMj
UxNjY4NTc3L2lhaW55dS9zYW1wbGVfYmlnZmlsZS50eHQmYj1pYWlueXU=^M
Content-Length: 618
Content-Type:  multipart/form-data; boundary=9fd48fba-1ea2-4789-8e6e-
64c40c243480

--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="op"
Content-Length: 17

upload_slice_init
--9fd48fba-1ea2-4789-8e6e-64c40c243480
```

```
Content-Disposition: form-data; name="sha"
Content-Length: 40

1
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="slice_size"
Content-Length: 7

1048576
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="filesize"
Content-Length: 5

26585
--9fd48fba-1ea2-4789-8e6e-64c40c243480
Content-Disposition: form-data; name="uploadparts"
Content-Length: 83

[{"offset":0,"datalen":26585,"datasha":"e98c925a97a8ae4ffec264ae674e28b8f975eba3"}]
--9fd48fba-1ea2-4789-8e6e-64c40c243480--
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 321
Connection: keep-alive
Date: Tue, 12 Sep 2017 07:36:45 GMT
Server: tencent-cos
x-cos-request-id: NTliNzhlOGRfMTliMjk0MGFfNDg0N18xMTdiY2E=

{
"code":0,
"message":"SUCCESS",
"request_id":"NTllOWQ2YjVfYzlhMzNiMGFfMTY0OV9jMzYwZTU=",
"data":
    {
        "access_url":"http://xy2-124566667.file.myqcloud.com/uploader1500000000",
    "resource_path":"/124566667/xy2/uploader1508497108630",
    "source_url":"http://xy2-124566667.cosgz.myqcloud.com/uploader1500000000",
    "url":"http://gz.file.myqcloud.com/files/v2/124566667/xy2/uploader1500000000",
    "vid":"4396314caa204d61f5d070d45248d9981508497077"
    }
```

```
}
```

逐个上传分片

最近更新时间：2020-02-12 11:22:48

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 对大于 20 MB 的文件进行逐片上传之前，需要先使用「初始化分片上传」API，从腾讯云获取到分片上传所需的参数（session 和 slice_size）。
前置条件：指定目录已存在，且已获得分片上传所需参数（session 和 slice_size）。
分片上传使用流程说明

1. 发送「初始化分片上传」请求。
2. 腾讯云会判断文件的上传状态，若未完成上传，会返回-4019错误，用户需调用查询上传分片接口查询已经上传完的分片，并进行断点续传的操作；若文件从未传输过，则返回用户上传的 session。
3. 设置 session 和 offset 参数，构造「逐个分片上传」请求，逐片上传后续文件。
4. 循环执行第 3 步直到文件的分片数据上传完成。
5. 调用finish接口结束分片上传。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>[/dir_name]/<file_name> HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <multi_effect_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填 “upload_slice_data”
filecontent	是	Binary	文件内容

session	是	String	唯一标识此文件传输过程的 ID，由后台下发， 调用方透传
offset	是	Int 64	本次分片位移

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

data 数据集合：

参数名称	必选	类型	描述
session	否	String	唯一标识此文件传输过程的 ID
offset	否	Int 64	请求包体里的传输的位移，调用方如果用多线程等方式传输,可以用来唯一确定本次分片结果

示例

逐个分片上传

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
uusZlBVdpHf800YuKlmvK5zvw75hPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFB
a1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0Nzc4MCZ0PTE0NzI2NDc2MDAmcj0xMDY
wMzQ1OTM4JmY9JmI9YWNjZXNzbG9n
Content-Length: 37406
Content-Type: multipart/form-data; boundary=--------------------------adb78d5a54eb

----------------------------adb78d5a54eb
Content-Disposition: form-data; name="op"

upload_slice_data
```

```
----------------------------adb78d5a54eb
Content-Disposition: form-data; name="session"

58ccd1b3120e85c4ebb7b786154408df4fe22230cb8f328d41599b237661855d624279
3ba2481c1ac0b1277051003cab
----------------------------adb78d5a54eb
Content-Disposition: form-data; name="offset"

0
----------------------------adb78d5a54eb
Content-Disposition: form-data; name="filecontent"

......G.E2...Q..l..E.i..U....w.o..v.....U2
(此处忽略文件内容)
----------------------------adb78d5a54eb--
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 175
Date: Wed Aug 31 20:46:40 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "datalen": 36864,
      "offset": 0,
      "session":
"58ccd1b3120e85c4ebb7b786154408df4fe22230cb8f328d41599b237661855d62427
93ba2481c1ac0b1277051003cab"
   }
}
```

结束上传分片

最近更新时间：2020-02-12 11:25:10

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 对大于 20 MB 的文件进行逐片上传之前，需要先使用「初始化分片上传」API，从腾讯云获取到分片上传所需的参数（session 和 slice_size）。
前置条件：指定目录已存在，且已获得分片上传所需参数（session 和 slice_size）。
当分片数据上传完成后，需调用此接口结束文件上传流程，腾讯云会返回文件的信息。
分片上传使用流程说明

1. 发送「初始化分片上传」请求。
2. 腾讯云会判断文件的上传状态，若未完成上传，会返回-4019错误，用户需调用查询上传分片接口查询已经上传完的分片，并进行断点续传的操作；若文件从未传输过，则返回用户上传的session。
3. 设置 session 和 offset 参数，构造「逐个分片上传」请求，逐片上传后续文件。
4. 循环执行第 3 步直到文件的分片数据上传完成。
5. 调用finish接口结束分片上传。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>[/dir_name]/<file_name>  HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <multi_effect_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填 “upload_slice_finish”

session	是	String	唯一标识此文件传输过程的 ID，由后台下发， 调用方透传
filesize	是	Int 64	文件大小
sha	否	String	文件的 SHA-1 校验码， **如果初始化上传携带了该参数，finish操作的时候也必须携带上，否则会返回-29094的错误**

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

data 数据集合：

参数名称	必选	类型	描述
access_url	否	String	通过 CDN 访问该文件的资源链接（访问速度更快）
resource_path	否	String	该文件在COS中的相对路径名，可作为其 ID 标识。 格式 /appid/bucket/filename。推荐业务端存储 resource_path ，然后根据业务需求灵活拼接资源 url（通过 CDN 访问 COS 资源的 url 和直接访问 COS 资源的 url 不同）。
source_url	否	String	（不通过 CDN ）直接访问 COS 的资源链接
url	否	String	操作文件的 url。业务端可以将该 url 作为请求地址来进一步操作文件，对应 API：查询文件属性、更新文件、删除文件、移动文件中的请求地址。

说明

腾讯云对象存储会默认为每个资源生成经 CDN 的访问链接 access_url，当业务端尚未开通 CDN 时，仍然可以获得该链接，但是无法访问。

示例

逐个分片上传

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
uusZlBVdpHf800YuKImvK5zvw75hPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFB
a1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0Nzc4MCZ0PTE0NzI2NDc2MDAmcj0xMDY
wMzQ1OTM4JmY9JmI9YWNjZXNzbG9n
Content-Length: 450
Content-Type: multipart/form-data; boundary=--------------------------2335c30f02ab

----------------------------2335c30f02ab
Content-Disposition: form-data; name="op"

upload_slice_finish
----------------------------2335c30f02ab
Content-Disposition: form-data; name="session"

58ccd1b3120e85c4ebb7b786154408df4fe22230cb8f328d41599b237661855d624279
3ba2481c1ac0b1277051003cab
----------------------------2335c30f02ab
Content-Disposition: form-data; name="filesize"

36864
----------------------------2335c30f02ab--
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 434
Date: Wed Aug 31 20:46:41 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS",
   "data": {
      "access_url": "http://accesslog-10055004.file.myqcloud.com/testfolder/111.txt",
```

```
        "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/testfolder/111.txt?cmd=txt_preview",
        "resource_path": "/10055004/accesslog/testfolder/111.txt",
        "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/testfolder/111.txt",
        "url":
"http://sh.file.myqcloud.com/files/v2/10055004/accesslog/testfolder/111.txt"
    }
}
```

查询上传分片

最近更新时间：2020-02-12 11:25:50

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 对大于 20 MB 的文件进行逐片上传之前，需要先使用「初始化分片上传」API，从腾讯云获取到分片上传所需的参数（session 和 slice_size）。
前置条件：指定目录已存在，且已获得分片上传所需参数（session 和 slice_size）。
当初始化分片上传返回-4019错误的时候，用户需调用此接口查询分片的断点和session，进行续传操作。
分片上传使用流程说明

1. 发送「初始化分片上传」请求。
2. 腾讯云会判断文件的上传状态，若未完成上传，会返回-4019错误，用户需调用查询上传分片接口查询已经上传完的分片，并进行断点续传的操作；若文件从未传输过，则返回用户上传的session。
3. 设置 session 和 offset 参数，构造「逐个分片上传」请求，逐片上传后续文件。
4. 循环执行第 3 步直到文件的分片数据上传完成。
5. 调用 finish 接口结束分片上传。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>[/dir_name]/<file_name>  HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <multi_effect_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填 “upload_slice_list”

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容
data	是	数据集合	服务器返回的应答数据

data 数据集合：

参数名称	必选	类型	描述
session	否	String	唯一标识此文件传输过程的 ID，命中秒传则不携带
slice_size	否	Int	分片大小，单位为 Byte。 有效值：524288 (512 KB)，1048576 (1 MB)，2097152 (2 MB)，3145728 (3 MB)
listparts	否	String	完成上传的文件的分片信息
access_url	否	String	文件已上传完成返回，通过 CDN 访问该文件的资源链接（访问速度更快）
resource_path	否	String	文件已上传完成返回，该文件在COS中的相对路径名，可作为其 ID 标识。 格式 /appid/bucket/filename。推荐业务端存储 resource_path ，然后根据业务需求灵活拼接资源 url（通过 CDN 访问 COS 资源的 url 和直接访问 COS 资源的 url 不同）
source_url	否	String	文件已上传完成返回，（不通过 CDN ）直接访问 COS 的资源链接
url	否	String	文件已上传完成返回，操作文件的 url。业务端可以将该 url 作为请求地址来进一步操作文件，对应 API：查询文件属性、更新文件、删除文件、移动文件中的请求地址。

说明

腾讯云对象存储会默认为每个资源生成经 CDN 的访问链接 access_url，当业务端尚未开通 CDN 时，仍然可以获得该链接，但是无法访问。

listparts 数据集合：

参数名称	必选	类型	描述
datalen	是	Int	分片大小
offset	是	Int64	分片偏移

示例

逐个分片上传

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
E+w/b7r3hVCOX9Dkc0sD0gU6ycFhPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFF
Ba1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0OTYwOCZ0PTE0NzI2NDk0Mjgmcj01NTg
zMDA2MzgmZj0mYj1hY2Nlc3Nsb2c=
Content-Length: 154
Content-Type: multipart/form-data; boundary=----------------------------2b804029035c

----------------------------2b804029035c
Content-Disposition: form-data; name="op"

upload_slice_list
----------------------------2b804029035c--
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 304
Date: Wed Aug 31 21:19:40 2016
Server: tencent-cos

{
    "code": 0,
    "message": "SUCCESS",
    "data": {
        "filesize": 104857600,
        "listparts": [
```

```
        {
            "datalen": 524288,
            "offset": 0
        },
        {
            "datalen": 524288,
            "offset": 524288
        },
        {
            "datalen": 524288,
            "offset": 1048576
        }
    ],
    "session":
"4331155f7e26e5fc7f032e7d1dac56644b68681d6c4260e39db441604773c883ca2b5e
d5f44ab5bd44f81f9ee383165f",
    "slice_size": 524288
  }
}
```

文件已经上传完成的情况

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
5/6eYiDGOxpd/YRDTscWG4Y265hhPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeFFB
a1JCQzJEUlhCdFBkN0NybEpRJmU9MTQ3MjY0OTQ4OCZ0PTE0NzI2NDkzMDgmcj0xNDA
1ODgwNTk0JmY9JmI9YWNjZXNzbG9n
Content-Length: 154
Content-Type: multipart/form-data; boundary=--------------------------72e1396a1d85

----------------------------72e1396a1d85
Content-Disposition: form-data; name="op"

upload_slice_list
----------------------------72e1396a1d85--
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 434
```

```
Date: Wed Aug 31 21:15:08 2016
Server: tencent-cos

{
    "code": 0,
    "message": "SUCCESS",
    "data": {
        "access_url": "http://accesslog-10055004.file.myqcloud.com/testfolder/111.txt",
        "preview_url": "http://accesslog-
10055004.preview.myqcloud.com/testfolder/111.txt?cmd=txt_preview",
        "resource_path": "/10055004/accesslog/testfolder/111.txt",
        "source_url": "http://accesslog-
10055004.cossh.myqcloud.com/testfolder/111.txt",
        "url":
"http://sh.file.myqcloud.com/files/v2/10055004/accesslog/testfolder/111.txt"
    }
}
```

查询文件属性

最近更新时间：2020-06-01 09:28:59

> **注意**
> 目前查阅的是历史版本 API 文档，后续不再更新和维护，我们建议您查阅新版 API 文档 。

功能描述

使用该 API 查询文件的属性信息，包括：包括文件大小、文件 SHA-1 校验码、文件创建时间、修改时间、访问权限、自定义头部信息等。使用该接口前请确认该文件已存在，如果 Bucket 中没有文件则请求不成功。

请求

语法示例：

```
GET /files/v2/1252448703/test02/sample_file.txt?op=stat
Host: <Region>.file.myqcloud.com
Authorization: <multi_effect_signature>
```

> **说明**
> Authorization：<MultiEffectSignature> 多次有效签名（详细参见 签名算法 章节）。

请求参数

该请求不带请求参数。

请求体

该请求的请求体为空。

响应

响应体

该响应体返回为 **application/json** 数据，包含完整节点数据的内容展示如下：

```
{
   "message": "SUCCESS",
   "code": 0,
   "data": {
      "slicesize": 0,
```

```
        "ctime": 1503385091,
        "biz_attr": "",
        "filelen": 0,
        "authority": "eInvalid",
        "source_url": "http://test01-1252448703.costj.myqcloud.com/sample_file.txt",
        "forbid": 0,
        "sha": "0000000000000000000000000000000000000000",
        "custom_headers": {
            "Cache-Control": "no",
            "Content-Language": "ch",
            "Content-Type": "application/pdf"
        },
        "filesize": 0,
        "mtime": 1503385091,
        "access_url": "http://test01-1252448703.file.myqcloud.com/sample_file.txt"
    },
    "request_id": "NTk5YmQ2MDNfYWRhYjM1MGFfMmVmZTdfZWFlODM="
}
```


具体的参数描述如下：

参数名称	描述	类型
code	服务端返回码，如果没有发生任何错误取值为0；如果发生错误该参数指称具体的错误码。COS 服务相关的错误码可以查看 COS 错误码汇总	Number
request_id	该请求的唯一标识 ID	String
message	服务端提示内容，如果发生错误该字段将详细描述发生错误的情况。	String
data	服务端返回的应答数据，该内容代表了接口返回的具体的业务数据。	Object

data 数据集参数描述：

参数名称	描述	类型
biz_attr	COS 服务调用方自定义属性，可通过 查询目录属性 获取该属性值	String
filesize	文件大小	Number

sha	文件 SHA-1 校验码	String
ctime	创建时间，10 位 Unix 时间戳（UNIX 时间是从协调世界时 1970 年 1 月 1 日 0 时 0 分 0 秒起的总秒数）	String
mtime	修改时间，10 位 Unix 时间戳（UNIX 时间是从协调世界时 1970 年 1 月 1 日 0 时 0 分 0 秒起的总秒数）	String
access_url	通过 CDN 访问该文件的资源链接	String
source_url	（不通过 CDN ）直接访问 COS 的资源链接	String
authority	Object 的权限，默认与 Bucket 权限一致，此时不会返回该字段。如果设置了独立权限，则会返回该字段。有效值：eInvalid 空权限，此时系统会默认调取 Bucket 权限 eWRPrivate 私有读写 eWPrivateRPublic 公有读私有写	String
custom_headers	用户自定义头部	Object

custom_headers 数据集参数描述：

参数名称	描述	类型
Cache-Control	文件的缓存机制	String
Content-Type	文件的 MIME 信息	String
Content-Disposition	MIME 协议的扩展	String
Content-Language	文件的语言	String
Content-Encoding	文件的加密格式	String
x-cos-meta-自定义内容	自定义内容	String

实际案例

请求

```
GET /files/v2/1252448703/lewzylu02/testfolder/?
op=list&pattern=eListBoth&order=0&num=20 HTTP/1.1
Host: gz.file.myqcloud.com
Date: Mon, 21 Aug 2017 11:54:31 GMT
Accept: */*
Authorization:
DUbxmGI7DuyqWQxB+LaUYDII+WdhPTEyNTI0NDg3MDMmaz1BS0lEMTVJc3NraUJRS1
RaYkFvNldoZ2NCcVZsczlTbXVHMDAmZT0xNTAzMzE2NjA1JnQ9MTUwMzMxNjQyNSZyP
TExMjMwJmY9JmI9bGV3enlsdTAy
Method:GET
User-Agent:cos-php-sdk-v4.3.2
Connection: keep-alive
```

响应

```
HTTP/1.1 200 OK
Content-Length: 427,
Content-Type: application/json; charset=utf-8
Server: tencent-cos
Connection: 'keep-alive
Date: Tue, 22 Aug 2017 07:35:36 GMT
x-cos-request-id': NTk5YmRlYzhfMmFhYzM1MGFfNzcxMV9mMjg1ZQ==

{
   "message": "SUCCESS",
   "code": 0,
   "data": {
      "slicesize": 0,
      "ctime": 1503385091,
      "biz_attr": "",
      "filelen": 0,
      "authority": "eInvalid",
      "source_url": "http://test01-1252448703.costj.myqcloud.com/sample_file.txt",
      "forbid": 0,
      "sha": "0000000000000000000000000000000000",
      "custom_headers": {},
      "filesize": 0,
      "mtime": 1503385091,
      "access_url": "http://test01-1252448703.file.myqcloud.com/sample_file.txt"
   },
   "request_id": "NTk5YmQ2MDNfYWRhYjM1MGFfMmVmZTdfZWFlODM="
}
```

更新文件属性

最近更新时间：2020-06-01 09:32:03

> **注意**
> 您目前查阅的是历史版本 API 文档，后续不再更新和维护，我们建议您查阅新版 API 文档。

功能描述

使用该 API 对腾讯云对象存储中某个文件的属性进行设置或修改。成功更新文件属性的前提条件是 Bucket 中已存在该文件。如果该 Bucket 中没有该文件请求不成功。

> **注意**
> 1. 目前仅提供对 authority 和 custom_headers 的更新操作，在更新时需要用 flag 来决定此次操作是更新 authority 还是 custom_headers 或者二者都更新。
> 2. 对于自定义头部 custom_headers 中的各属性值，均由用户来定义和维护，腾讯云对象存储不负责维护其属性值是否正确有效。

请求

语法示例：

```
POST /files/v2/100088888/test/sample_file.txt HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Length: <ContentLength>
Content-Type: application/json
Authorization: <OnceSignature>
```

> **说明**
> - Authorization: <OnceSignature> 单次有效签名（详细参见 签名算法 章节）。
> - Content-Length: <ContentLength> RFC 2616 中定义的 HTTP 请求内容长度（字节）。

请求参数

该请求不带请求参数。

请求体

该 API 接口请求的请求体具体节点内容为：

```
{
   "biz_attr": "demo",
   "custom_headers": {
      "x-cos-meta-yyy": "yyy",
      "Content-Disposition": "ccccxxx.txt",
      "Content-Language": "english",
      "Cache-Control": "cache_xxx",
      "Content-Type": "application/text",
      "x-cos-meta-xxx": "xxx"
   },
   "authority": "eWRPrivate",
   "op": "update"
}
```

具体内容描述如下：

参数名称	描述	类型	必选
biz_attr	COS 服务调用方自定义属性，可通过 查询目录属性 获取该属性值	String	否
custom_headers	用户自定义 header，在执行更新操作时，只需填写需要增加或修改的项	Object	否
authority	Object 的权限，默认与 Bucket 权限一致，此时不会返回该字段。如果设置了独立权限，则会返回该字段。 有效值：eInvalid（空权限），此时系统会默认调取 Bucket 权限，eWRPrivate（私有读写），eWPrivateRPublic（公有读私有写）	String	否
op	操作类型，填“update”	String	是

custom_headers 数据集参数描述：

参数名称	描述	类型	必选
x-cos-meta-yyy	用户自定义内容 yyy	String	否
Content-Disposition	MIME 协议的扩展	String	否
Content-	文件的语言	Strin	否

Language		g	
Cache-Control	文件的缓存机制	String	否
Content-Type	文件的 MIME 信息	String	否
x-cos-meta-xxx	用户自定义内容 xxx	String	否

响应

响应体

该响应体返回为 **application/json** 数据，包含完整节点数据的内容展示如下：

```
{
  "message": "SUCCESS",
  "code": 0,
  "request_id": "NTk5YmRjNjZfNmNhZDM1MGFfOGE0MF9lZDA1MQ=="
}
```

具体的参数描述如下：

参数名称	描述	类型
code	服务端返回码，如果没有发生任何错误取值为**0**；如果发生错误该参数指称具体的错误码。COS 服务相关的错误码可以查看 COS 错误码	Number
request_id	该请求的唯一标识 ID	String
message	服务端提示内容，如果发生错误该字段将详细描述发生错误的情况。	String

实际案例

请求

```
POST /files/v2/1252448703/test02/sample_file.txt HTTP/1.1
Content-Length: 268
Accept-Encoding: gzip, deflate
Accept': */*
Connection: keep-alive
```

```
Content-Type: application/json
Authorization:
W0o/dAf5RjYPqBOkFx+TVAw2PwhhPTEyNTI0NDg3MDMmaz1BS0lEMTVJc3NraUJRS1Ra
YkFvNIdoZ2NCcVZsczlTbXVHMDAmZT0wJnQ9MTUwMzM4OTUwMyZyPTc1ODUyMTM3O
SZmPS8xMjUyNDQ4NzAzL3Rlc3QwMi9zYW1wbGVfZmlsZS50eHQmYj10ZXN0MDI=

{
   "biz_attr": "demo",
   "custom_headers": {
      "x-cos-meta-yyy": "yyy",
      "Content-Disposition": "ccccxxx.txt",
      "Content-Language": "english",
      "Cache-Control": "cache_xxx",
      "Content-Type": "application/text",
      "x-cos-meta-xxx": "xxx"
   },
   "authority": "eWRPrivate",
   "op": "update"
}
```

响应

```
HTTP/1.1 200 OK
Content-Length': 616
Content-Type: application/json; charset=utf-8
Server: tencent-cos
Connection: keep-alive
Date: Tue, 22 Aug 2017 07:35:36 GMT
x-cos-request-id: NTk5YmRlYzhfMmFhYzM1MGFfNzcwNF9mNTU0OQ==

{
   "message": "SUCCESS",
   "code": 0,
   "request_id": "NTk5YmRjNjZfNmNhZDM1MGFfOGE0MF9lZDA1MQ == "
}
```

复制文件

最近更新时间：2020-02-12 11:28:29

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 进行文件的复制操作。“请求地址”中的参数为源文件的位置，“请求内容”中的参数 dest_fileid 指明复制至的目标路径（以及新的文件名）。
说明：当复制文件 A 至目标目录时，若目标目录下存在与 A 同名的文件，则请求中的 to_over_write 将决定是否覆盖该同名文件。若选择覆盖，则复制操作成功，若选择不覆盖，则复制操作失败。
前置条件：该文件已存在，且目标目录也已存在。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>/[dir_name]/<file_name> HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <once_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填”copy"
dest_fileid	是	String	目标路径（不带路径则为当前路径下，带路径则会复制到携带指定的路径下）
to_over_write	否	Int	覆盖写入目标文件选项，有效值： 0：不覆盖（若已存在重名文件，则不做覆盖，返回“复制失败”） 1：覆盖 默认值为0不覆盖。

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容

示例

请求

```
POST /files/v2/251668577/rabbitliutest4x/rabbit.txt HTTP/1.1
Authorization:ksYXsEbCG1G9ELdP15bp3LH9WQthPTEyNTE2Njg1Nzcmaz1BS0lEV3RUQ
0JZak01T3dMQjlDQXdBMVFiMlRoVFNVamZHRk8mZT0xNDc5Mzg4Njg3JnQ9MTQ3Njc5N
jY4NyZyPTk1OTM4ODQwNSZmPSZiPXJhYmJpdGxpdXRlc3Q0eA==
Host: gz.file.myqcloud.com
Content-Length: 255
Content-Type: multipart/form-data; boundary=--------------------------e492f6322ce5

HTTP/1.1 100 Continue

----------------------------e492f6322ce5
Content-Disposition: form-data; name="op"

copy
----------------------------e492f6322ce5
Content-Disposition: form-data; name="dest_fileid"

rabbit.txt.bak
----------------------------e492f6322ce5--
```

返回

```
HTTP/1.1 200 OK
Server: tencent-cos
Date: Thu, 15 Jun 2016 06:46:48 GMT
Content-Type: */*
Content-Length: 41
```

```
{
  "code": 0,
  "message": "SUCCESS"
}
```

下载文件

最近更新时间：2020-10-14 10:17:42

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

功能描述

使用该 API 下载 Bucket 中对应文件。公有文件无需拼接签名，私有文件需要拼接多次有限签名。

请求

语法示例：

```
GET /abcde.flv HTTP/1.1
Host: <bucketname-APPID>.cos<Region>.myqcloud.com
Authorization: <MultiEffectSignature>
```

> **说明**
> Authorization: <MultiEffectSignature> 多次有效签名（详细请参见 签名算法 文档）。

请求参数

该请求不带请求参数。

请求体

该请求的请求体为空。

响应

响应体

该请求的响应内容为 Object 文件。

实际案例

下面的请求和响应案例是 `abcde.flv` 文件在根目录下进行的 GET 操作。如需下载某目录（如 doc）下的文件 `abcde.flv` ，则发起 GET 请求时，将 `/abcde.flv` 改为 `/doc/abcde.flv` 即可。

请求

```
GET /abcde.flv HTTP/1.1
Host: examplebucket-1250000000.costj.myqcloud.com
Authorization:
lG59otAcouKclTGFddctiwVYqRNhPTEyNTE2Njg1Nzcmaz1BS0lEcDhSYVhIa0xoVDl5aEZ3
bG9mSzFmSWkxOWpkT3dLVVYmZT0xNTAzNTE4MzM0JnQ9MTUwMzkwOTQ5NiZyPTQz
Nzg5NDY5ODU5NDM1NTE3MzgmZj0mYj1zZXZlbnlvdXRlc3Q=
```

响应

```
HTTP/1.1 200 OK
Content-Type: video/x-flv
x-cos-storage-class: STANDARD
x-cos-version-id: null
ETag: a193f1f55f601956a47ea3c2283d1a1d
x-cos-object-type: normal
Last-Modified: Tue, 01 Aug 2017 15:39:43 GMT
Content-Length: 67

Object file
```

删除文件

最近更新时间：2020-02-12 11:31:38

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 进行文件的删除操作。
前置条件：该文件已存在。

> **注意**
> 对象一旦删除，不可恢复。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>[/dir_name]/<file_name>  HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: application/json
Authorization: <once_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填”delete”

返回

返回内容

参数名称	必	类	描述

	选	型	
code	是	Int	服务端返回码
message	是	String	服务端提示内容

示例

请求

```
POST /files/v2/10055004/accesslog/testfolder/111.txt HTTP/1.1
Host: sh.file.myqcloud.com
Authorization:
NvUbNZcimJ20OGI4tTVMX6GwGD5hPTEwMDU1MDA0Jms9QUtJRHpuOHd3S3VYanhpeF
FBa1JCQzJEUlhCdFBkN0NybEpRJmU9MCZ0PTE0NzI2NDU4NTgmcj0yMzY2NjE2NDkmZj
0vMTAwNTUwMDQvYWNjZXNzbG9nL3Rlc3Rmb2xkZXIvMTExLnR4dCZiPWFjY2Vzc2xvZ
w==
Content-Type: application/json
Content-Length: 15

{
   "op": "delete"
}
```

返回

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 30
Date: Wed Aug 31 20:17:38 2016
Server: tencent-cos

{
   "code": 0,
   "message": "SUCCESS"
}
```

移动文件

最近更新时间：2020-02-12 11:31:00

> **注意**
> 您目前查阅的是历史版本 API 文档，已不再更新和维护，我们建议您查阅新版 API 文档 。

描述

使用 API 进行文件的移动或重命名操作。“请求地址”中的参数为源文件的位置，“请求内容”中的参数 dest_fileid 指明移动至的目标路径（以及新的文件名）。
说明：当移动文件 A 至目标目录时，若目标目录下存在与 A 同名的文件，则请求中的 to_over_write 将决定是否覆盖该同名文件。若选择覆盖，则移动操作成功，若选择不覆盖，则移动操作失败。
前置条件：该文件已存在，且目标目录也已存在。

请求

请求语法

```
POST /files/v2/<appid>/<bucket_name>/[dir_name]/<file_name> HTTP/1.1
Host: <Region>.file.myqcloud.com
Content-Type: multipart/form-data
Authorization: <once_signature>
```

> **说明**
> <箭头括号>表示必须替换为有效值的变量，[英文方括号]表示可选的命令或参数。

请求内容

参数名称	必选	类型	描述
op	是	String	操作类型，填”move"
dest_fileid	是	String	目标路径（不带路径则为当前路径下，带路径则会移动到携带指定的路径下）
to_over_write	否	Int	覆盖写入目标文件选项，有效值： 0：不覆盖（若已存在重名文件，则不做覆盖，返回“移动失败”） 1：覆盖 默认值为0不覆盖

返回

返回内容

参数名称	必选	类型	描述
code	是	Int	服务端返回码
message	是	String	服务端提示内容

示例

请求

```
POST /files/v2/251668577/rabbitliutest4x/rabbit.txt HTTP/1.1
Authorization:ksYXsEbCG1G9ELdP15bp3LH9WQthPTEyNTE2Njg1Nzcmaz1BS0lEV3RUQ
0JZak01T3dMQjlDQXdBMVFiMlRoVFNVamZHRk8mZT0xNDc5Mzg4Njg3JnQ9MTQ3Njc5N
jY4NyZyPTk1OTM4ODQwNSZmPSZiPXJhYmJpdGxpdXRlc3Q0eA==
Host: gz.file.myqcloud.com
Content-Length: 255
Content-Type: multipart/form-data; boundary=----------------------------e492f6322ce5

HTTP/1.1 100 Continue

----------------------------e492f6322ce5
Content-Disposition: form-data; name="op"

move
----------------------------e492f6322ce5
Content-Disposition: form-data; name="dest_fileid"

rabbit.txt.bak
----------------------------e492f6322ce5--
```

返回

```
HTTP/1.1 200 OK
Server: tencent-cos
Date: Thu, 15 Jun 2016 06:46:48 GMT
Content-Type: */*
```

```
Content-Length: 41

{
   "code": 0,
   "message": "SUCCESS"
}
```

错误码

最近更新时间：2023-11-03 15:13:31

> **注意**
> 您目前查阅的是历史版本 API 错误码文档，已不再更新和维护，我们建议您使用新版 API，关于新版 API 错误码，请参见新版 API 错误码 文档。

常见错误码

错误码	描述	解决措施	HTTP STATUS
-47	签名中的 SECRETID 字段不存在	请检查签名中的 SECRETID 字段是否填写正确	403
-48	签名计算错误	请检查签名生成算法是否正确，生成的签名与预期不一致	403
-59	签名过期	请检查签名中的开始时间和结束时间是否有效，同时校验客户端的机器时间是否正确	403
-61	签名中的 bucket 与实际访问的 bucket不一致	请检查签名中的 bucket 与实际访问的 bucket 是否一致	403
-63	签名中的 fileid 与实际访问的 fileid 不一致	请检查签名中的 fileid 与实际访问的 fileid 是否一致	403
-70	签名错误,签名中的 APPID 字段与实际访问 APPID 字段不一致	请检查签名中的 APPID 与实际访问的 APPID 是否一致	403
-71	请求太频繁	请求触发了频控，QPS 规格请参见 规格与限制 文档	503
-97	签名比对错误	请检查签名是否计算正确，签名中的字段是否与实际发出请求中的字段一致	403
-121	匹配跨域规则失败	请检查 Origin，request method Access-Control-Request-Method 或 Access-Control-Requet-Headers 是否符合跨域规则	403
-1	指定的 bucket 不存在	请检查 bucket 是否拼写正确，或地域信息是否	404

33		填写正确	
-175	分片上传的预设文件长度与实际 finish 的长度不一致	请检查实际上传数据的长度是否和初始化分片上传时设置的文件长度相等	400
-177	文件已经存在	待上传的文件已经存在，如需覆盖请带上 inser_only 参数重新上传	409
-178	待创建的目录已经存在	请检查待创建的目录是否存在	409
-195	后端存储错误	服务器遇到了内部错误，请联系开发者	500
-196	服务暂时不可用	后端网络错误，请稍后重试	503
-197	指定的文件不存在	请检查路径名是否正确	404
-199	bucket 的名字不符合规则	请检查 bucket 是否只包含了数字、小写字母以及中划线-，是否超过了最大长度限制 850	400
-284	初始化分片上传的 slice_size 设置错误	请设置 slice_size 的大小为 1M，即 1048576	400
-4016	sha 校验失败	请检查 sha 值是否计算正确，或者文件上传过程中发生了未知改变	409
-4019	sha 值相同的文件未完成上传	请查询已经上传完的分片，并进行断点续传的操作	409
-4020	sha 值不同的文件未完成上传	请检查是否有 sha 值不同的文件未完成上传与现有操作发生冲突	409
-5965	初始化分片上传的 uploadparts 参数错误	请检查 uploadparts 是否符合 json 格式	400
-5997	后端网络错误	后端网络错误，请稍后重试	503
-599	参数错误	请检查对应请求的参数是否正确	400

9			
-200034	并发操作文件	请检查是否并发上传文件,或者并发修改某个文件的属性	409
-210095	存储后端错误	后端存储错误，请稍后重试	503
-290095	不带 sha 上传,上传分片的 offset 不是 1048576 的整数倍	请检查上传分片的 offset 是否是 1048576(1MB)的整数倍	400
-290096	sessionid 校验失败	请检查 sessionid 是否与初始化分片上传时返回的保持一致	400
-291005	文件上传超过系统最大限制	请检查上传文件的大小是否超过限制，文件大小最大支持 64G	400
-291033	带sha上传,初始化分片上传时，没有带 uploadparts 参数	带 sha 上传时，请检查是否带有 uploadparts 参数	400
-291034	初始化分片上传时，校验 uploadparts 中的参数失败	请检查初始化分片上传的 uploadparts 中的分片的 offset 是否与前面分片的长度相匹配, 或分片长度之和是否等于文件总长度	400
-291036	带 sha 上传,上传分片的 offset 与初始化分片上传时的 uploadparts 中的 offset 不匹配	请检查上传分片的 offset 是否和初始化的 offset 保持一致	400
-291038	上传数据的 SHA 校验失败	请检查头部中的 SHA 值是否与实际上传内容的 SHA 值一致	400
-291044	完成分片上传时，找不到对应的分片内容	请检查是否上传完每片数据	400
-293002	存储后端错误	后端存储错误，请稍后重试	503
-4	文件名非 utf8 编码	请对文件名进行 UTF8 编码	400

6157			
-46628	下载文件时，文件不存在	请确认请求下载的文件是否真实存在	404
-46646	文件禁止下载	文件已被封禁，请检查文件内容是否违规	451
-46659	bucket 打开版本控制禁止使用 JSON API	如您已开启了版本控制，请使用最新的 XML API	409
-50152	分片上传的预设文件长度与实际 finish 的长度不一致	请检查实际上传数据的长度是否和初始化分片上传时设置的文件长度相等	400

其它错误码

错误码	描述
-5996	HTTP 请求方法错误，或者没有发送 HTTP 请求方法
-5994	url 路径不符合规范
-5993	formdata body 格式不正确
-5991	append 接口，大于 5G 的追加上传
-5980	"没有 op 字段，用户没填写操作类型或格式错误解析不到"
-597	请求的 json 格式不正确

5	
-5971	分片上传 init 时没有填写文件大小
-5970	refer 白名单数量必须小于等于10
-5969	请 联系我们，数量必须小于等于10
-5968	refer 黑名单数量必须小于等于10
-5967	密钥数量必须小于等于 10
-5966	list 参数非数字或者大于 200
-5965	分片上传 init 时未指定 slice_size 参数
-5964	参数 op 填写错误，没有此操作
-5963	读取数据超时，检查网络是否过慢或上传并发数过大
-5962	XML 解析失败，请求携带的 XML 不符合规范
-5961	请求 body 体的 Content-MD5 不符合预期
-595	操作不存在的 bucket

7	
-5949	copy 文件时查询源文件信息失败
-5947	biz_attr 最大 4K
-5946	缺少头域 CONTENT_MD5，有些接口必须校验内容
-5945	内部错误，请 联系我们
-5939	内部错误，请 联系我们
-5938	内部错误，请 联系我们
-5937	内部错误，请 联系我们
-4018	文件已存在，并且已完成上传
-288	内部错误，请 联系我们
-287	内部错误，请 联系我们
-285	上传文件大小大于限制
-283	分片上传分片小于限制，或者 append 文件小于 4K
-281	分片上传分片大于限制

-194	内部错误，请 联系我们
-185	内部错误，请 联系我们
-184	内部错误，请 联系我们
-183	内部错误，请 联系我们
-181	内部错误，请 联系我们
-180	路径错误，可能是 url 超过长度限制，非法字符等原因
-176	创建的 bucket 已经存在
-173	删除非空的 bucket
-172	bucket 数量超过 200 个限制
-134	bucket 不支持跨域访问
-120	uploadid 不存在
-119	内部错误，请 联系我们
-118	上传块编号找不到
-117	分块上传 finish 时块编号不连续
-116	内部错误，请 联系我们
-101	"不需要拷贝,可能上次更新时间/ETag 规则匹配问题"

-100	copy 的前置条件冲突
-99	内部错误，请 联系我们
-98	内部错误，请 联系我们
-96	签名过期
-85	内部错误，请 联系我们
-83	内部错误，请 联系我们
-81	签名为空（需鉴权的操作提供签名为空）
-79	secret id 不存在
-77	单次性签名已被使用过
-76	单次签名没有指定f参数
-75	内部错误，请 联系我们
-74	多次签名-过期时间为 0
-73	单次签名-过期时间不为 0
-66	内部错误，请 联系我们
-64	secretid 中的 uid 和签名中的 APPID 不匹配
-62	secretid 可能填错或请 联系我们

-58	内部错误，请 联系我们
-57	签名缺失必填字段
-56	签名中的 header 未排序
-55	用户没有此操作权限
-53	签名有效期区间设置过长，最多设置一个月
-29214	频控过载
-29175	内部错误，请 联系我们
-29173	分块上传不应指定存储介质
-29172	内部错误，请 联系我们
-29171	内部错误，请 联系我们
-29170	内部错误，请 联系我们
-29169	内部错误，请 联系我们
-29168	内部错误，请 联系我们
-2	list 分块请求没有携带 UploadID

9166	
-29165	内部错误，请 联系我们
-29164	内部错误，请 联系我们
-29163	内部错误，请 联系我们
-29162	内部错误，请 联系我们
-29161	分片上传完成接口指定的 Etag 长度错误
-29159	结束上传分片请求 body 为空
-29156	内部错误，请 联系我们
-29155	内部错误，请 联系我们
-29150	内部错误，请 联系我们
-29148	内部错误，请 联系我们
-29147	JSON API 接口 md5 并发上传时，读取分片的内容和上传内容不一致
-2	内部错误，请 联系我们

9146	
-29145	内部错误，请 联系我们
-29143	上传文件 MD5 校验失败
-29141	分片上传 init 指定的 filesize 错误
-29140	内部错误，请 联系我们
-29137	内部错误，请 联系我们
-29132	内部错误，请 联系我们
-29131	内部错误，请 联系我们
-29130	内部错误，请 联系我们
-29129	下载文件时文件尚未完成上传
-29128	内部错误，请 联系我们
-29126	初始化分片上传未指定 filesize 参数
-2	内部错误，请 联系我们

9125	
-29124	内部错误，请 联系我们
-29123	内部错误，请 联系我们
-29119	内部错误，请 联系我们
-29118	内部错误，请 联系我们
-29117	内部错误，请 联系我们
-29115	内部错误，请 联系我们
-29114	内部错误，请 联系我们
-29112	内部错误，请 联系我们
-29111	内部错误，请 联系我们
-29110	请求 json 格式错误
-29109	内部错误，请 联系我们
-2	内部错误，请 联系我们

9108	
-2911007	内部错误，请 联系我们
-2911006	内部错误，请 联系我们
-2911004	结束上传分片未指定filesize参数
-2911003	上传时 sessionid 传错
-2911002	内部错误，请 联系我们
-2911001	内部错误，请 联系我们
-2911000	内部错误，请 联系我们
-2909099	内部错误，请 联系我们
-2909098	用户上传数据偏移量不是每块的整数倍错误
-2909094	传的 sessionid 不合法
-2909093	传的 sessionid 不合法
-2	内部错误，请 联系我们

9088	
-29087	内部错误，请 联系我们
-29086	内部错误，请 联系我们
-29085	内部错误，请 联系我们
-29084	内部错误，请 联系我们
-29083	内部错误，请 联系我们
-29082	内部错误，请 联系我们
-29081	内部错误，请 联系我们
-29080	内部错误，请 联系我们
-29070	内部错误，请 联系我们
-29069	内部错误，请 联系我们
-29066	内部错误，请 联系我们
-2	分片 md5 不一致

9063	
-2901044	内部错误，请 联系我们
-2901043	文件不存在
-2901037	If-Modified-Since 用户头部 If-Modified-Since 时间格式错误
-2901036	文件未上传完成，不允许下载
-2901034	下载的 offset 大于文件的 fsize
-2901033	下载请求没有有效的 range
-2901031	内部错误
-2901027	内部错误，请 联系我们
-2901026	内部错误，请 联系我们
-2901021	上传文件 sessionid 传输错误
-2901015	HEADER 字段中没有 host 或者 host 错误
-2	内部错误，请 联系我们

9012	
-290011	内部错误，请 联系我们
-290010	内部错误，请 联系我们
-290003	内部错误，请 联系我们
-290002	内部错误，请 联系我们
-290001	内部错误，请 联系我们
-290000	内部错误，请 联系我们
101	内部错误，请 联系我们
100	内部错误，请 联系我们
-299993	内部错误，请 联系我们
-299992	内部错误，请 联系我们
-299991	内部错误，请 联系我们
-466042	APPID 不存在

-4 66 41	多版本下的文件不存在
-4 66 40	内部错误，请 联系我们
-4 66 39	内部错误，请 联系我们
-4 66 38	内部错误，请 联系我们
-4 66 37	内部错误，请 联系我们
-4 66 36	内部错误，请 联系我们
-4 66 35	内部错误，请 联系我们
-4 66 34	内部错误，请 联系我们
-4 66 32	追加文件长度为 0
-4 66 27	用户在黑名单里面
-4 66 26	内部错误，请 联系我们
-4 66 22	非法分片大小

-4 66 21	文件被封禁，可能涉黄政治敏感信息被封禁
-4 66 20	bucket 被封禁，可能 qps 过高，恶意攻击被后台封禁
-4 66 19	签名过期，可能生成签名机器的时间跟北京时间不一致，或者用户拿着签名过期串访问
-4 66 18	签名校验失败，用户签名算法错误，密钥被禁用，APPID 和密钥对不上等错误
-4 66 17	bucket 的 refer 黑名单配置了访问的 refer 域名
-4 66 16	bucket 的 refer 白名单没有配置访问的 refer 域名
-4 66 15	bucket 禁止访问，可能您的 bucket 访问量过大或欠费或违反了政策法规导致的封禁
-4 66 14	文件没有完成上传，只传了一部分，检查上传时候有无报错
-4 66 13	自定义的 biz_attr 过大，目前最大 4K
-4 66 07	内部错误，请 联系我们
-4 66 01	内部错误，请 联系我们
-4 63 13	内部错误，请 联系我们

-463112	内部错误，请 联系我们
-463111	内部错误，请 联系我们
-463110	内部错误，请 联系我们
-463109	内部错误，请 联系我们
-463108	内部错误，请 联系我们
-463107	内部错误，请 联系我们
-463106	内部错误，请 联系我们
-463105	内部错误，请 联系我们
-463104	内部错误，请 联系我们
-463103	内部错误，请 联系我们
-463102	内部错误，请 联系我们
-463101	内部错误，请 联系我们

-462012	内部错误，请 联系我们
-462011	内部错误，请 联系我们
-462010	内部错误，请 联系我们
-462009	上传时 versionid 不正确
-462008	相同的文件名已存在，并且是目录
-462007	内部错误，请 联系我们
-462006	内部错误，请 联系我们
-462005	内部错误，请 联系我们
-462004	跨域设置时候规则超长，最大 100KB
-462003	跨域设置时候规则超长，最大 100KB
-462002	内部错误，请 联系我们
-462001	内部错误，请 联系我们

-4 62 00	批量删除时，数量过大，最多 1000 个
-4 61 99	分块上传 finish 时，传入的各个分块的 Etag 值与系统对不上
-4 61 98	在非追加类型的文件上追加数据
-4 61 97	追加上传时，offset 和 cos 上文件大小不一致
-4 61 96	内部错误，请 联系我们
-4 61 95	内部错误，请 联系我们
-4 61 94	内部错误，请 联系我们
-4 61 93	内部错误，请 联系我们
-4 61 92	内部错误，请 联系我们
-4 61 91	内部错误，请 联系我们
-4 61 90	内部错误，请 联系我们
-4 61 89	内部错误，请 联系我们

-4 61 88	跨域设置时候规则超长，最大 100KB
-4 61 87	跨域设置的规则参数有误
-4 61 86	设置跨域时标志有误
-4 61 85	自定义 header 超长，最大 4KB
-4 61 84	内部错误，请 联系我们
-4 61 83	内部错误，请 联系我们
-4 61 82	内部错误，请 联系我们
-4 61 81	refer 长度过长，最大 3K
-4 61 80	签名过长，最大 2K
-4 61 79	封禁文件时，标志传错
-4 61 78	查询文件列表时，delimiter 长度大于 1
-4 61 77	list 时填的 offset 长度超过 path 长度限制

-4 61 76	biz attr 长度非法，最大 4K
-4 61 75	内部错误，请 联系我们
-4 61 74	内部错误，请 联系我们
-4 61 73	内部错误，请 联系我们
-4 61 72	内部错误，请 联系我们
-4 61 71	内部错误，请 联系我们
-4 61 70	下载文件时，host 值传的不对
-4 61 69	内部错误，请 联系我们
-4 61 68	内部错误，请 联系我们
-4 61 58	内部错误，请 联系我们
-4 61 05	内部错误，请 联系我们
-4 61 04	内部错误，请 联系我们

-4 61 03	内部错误，请 联系我们
-4 61 02	内部错误，请 联系我们
-4 61 01	内部错误，请 联系我们
-4 60 31	内部错误，请 联系我们
-4 60 30	内部错误，请 联系我们
-4 60 26	内部错误，请 联系我们
-4 60 25	内部错误，请 联系我们
-4 60 24	内部错误，请 联系我们
-4 60 23	内部错误，请 联系我们
-4 60 22	内部错误，请 联系我们
-4 60 20	内部错误，请 联系我们
-4 60 07	内部错误，请 联系我们

-4 60 05	内部错误，请 联系我们
-4 60 04	内部错误，请 联系我们
-4 60 03	内部错误，请 联系我们
-4 60 02	内部错误，请 联系我们
-4 60 01	内部错误，请 联系我们
-5 01 22	内部错误，请 联系我们

历史版本 SDK

概览

最近更新时间：2020-02-12 13:40:50

> **注意**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

基于 JSON API 封装的 SDK

SDK	接入文档
Android SDK	Android SDK 接入说明
C# SDK	C# SDK 接入说明
C++ SDK	C++ SDK 接入说明
iOS SDK	iOS SDK 接入说明
Java SDK	Java SDK 接入说明
JavaScript SDK	JavaScript SDK 接入说明
PHP SDK	PHP SDK 接入说明
Python SDK	Python SDK 接入说明

Android SDK

最近更新时间：2020-02-12 13:43:55

> **注意**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

SDK 获取

GitHub 地址：Android SDK 。
更多示例可参考Demo： Android SDK Demo 。
（本版本 SDK 基于 JSON API 封装组成）

开发准备

1. SDK 支持 Android 2.2 及以上版本的手机系统。
2. 手机必须要有网络（GPRS、3G 或 WI-FI 网络等）。
3. 手机可以没有存储空间，但会使部分功能无法正常工作。
4. 从控制台获取 APPID、SecretId、SecretKey，详情参考权限控制。

SDK 配置

配置工程导入下列 jar 包：

- cos-android-sdk1.4.3.18.jar
- okhttp-3.2.0.jar
- okio-1.6.0.jar
- sha1utils.jar
- jniLibs（对应的 sha1 值计算的 .so 库）

SDK 需要网络访问相关的一些权限，需要在 AndroidManifest.xml 中增加如下权限声明：

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

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

初始化

进行操作之前需要实例化 COSClient 和 COSClientConfig。

实例化 COSClientConfig

调用 COSClientConfig() 方法实例化 COSClientConfig 对象。

```
COSClientConfig config = new COSClientConfig();
```

配置 COSClientConfig

方法	方法描述
setEndPoint(String endPoint)	设置园区：华南 "gz"， 华北 "tj"， 华东"sh"，新加坡"sgp"。SDK 中默认为华南地区
setConnectionTimeout(int connectionTimeout)	连接超时设置 （毫秒）
setSocketTimeout(int socketTimeout)	读取超时设置 （毫秒）
setMaxConnectionsCount(int maxConnectionsCount)	并发数大小设置
setMaxRetryCount(int maxRetryCount)	失败请求重试次数
setHttpProtocol(String httpProtocol)	设置请求协议类型：默认为http请求，即"http://"；若为https请求，则为 "https://"

实例化 COSClient

调用 COSClient（Context context, String appid, COSClientConfig config, String peristenceId) 方法实例化 COSClient 对象。

参数说明

参数名称	类型	是否必填	参数描述
context	Context	是	上下文

appid	String	是	腾讯云注册的 AppID
config	COSClientConfig	否	配置设置
persistenceId	String	否	持久化 ID，每个 COSClient 需设置一个唯一的 ID 用于持久化保存未完成任务 列表，以便应用退出重进后能够继续进行上传；传入为 Null，则不会进行持久化保存

示例

```
//创建COSClientConfig对象，根据需要修改默认的配置参数
COSClientConfig config = new COSClientConfig();
//设置园区
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);
```

快速入门

初始化 COSClient

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

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

COSClient cos = new COSClient(context,appid,config,peristenceId);
```

上传文件

```
String bucket = "cos空间名称";
String cosPath = "远端路径，即存储到cos上的路径";
String srcPath = "本地文件的绝对路径";
```

```
String sign = "签名，此处使用多次签名";

PutObjectRequest putObjectRequest = new PutObjectRequest();
putObjectRequest.setBucket(bucket);
putObjectRequest.setCosPath(cosPath);
putObjectRequest.setSrcPath(srcPath);
putObjectRequest.setSign(sign);
putObjectRequest.setListener(new  IUploadTaskListener(){
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {

        PutObjectResult result = (PutObjectResult) cosResult;
        if(result != null){
            StringBuilder stringBuilder = new StringBuilder();
            stringBuilder.append(" 上传结果：  ret=" + result.code + "; msg ="
+result.msg + "\n");
            stringBuilder.append(" access_url= " + result.access_url == null ? "null"
:result.access_url + "\n");
            stringBuilder.append(" resource_path= " + result.resource_path == null ?
"null" :result.resource_path + "\n");
            stringBuilder.append(" url= " + result.url == null ? "null" :result.url);
            Log.w("TEST",stringBuilder.toString());
         }
    }

    @Override
    public void onFailed(COSRequest COSRequest, final COSResult cosResult) {
        Log.w("TEST","上传出错：  ret =" +cosResult.code + "; msg =" + cosResult.msg);
    }

    @Override
    public void onProgress(COSRequest cosRequest, final long currentSize, final long
totalSize) {
        float progress = (float)currentSize/totalSize;
        progress = progress *100;
        Log.w("TEST","进度：  " + (int)progress + "%");
     }
});
PutObjectResult result = cos.PutObject(putObjectRequest);
```

下载文件

```
String downloadURl =  "下载文件的URL";
String savePath = "本地保存文件的路径";
```

```
String sign = "开启token防盗链了，则需要签名；否则，不需要";

GetObjectRequest getObjectRequest = new
GetObjectRequest(downloadURI,savePath);
getObjectRequest.setSign(null);
getObjectRequest.setListener(new IDownloadTaskListener() {
    @Override
    public void onProgress(COSRequest cosRequest, final long currentSize, final long
totalSize) {
        float progress =  currentSize / (float)totalSize;
        progress = progress * 100;
        progressText.setText("progress =" + (int) (progress) + "%");
        Log.w("TEST", "progress =" + (int) (progress) + "%");
    }

    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        Log.w("TEST","code =" + cosResult.code + "; msg =" + cosResult.msg);
    }

    @Override
    public void onFailed(COSRequest COSRequest, COSResult cosResult) {
        Log.w("TEST","code =" + cosResult.code + "; msg =" + cosResult.msg);
    }
});

GetObjectResult getObjectResult = cos.getObject(getObjectRequest);
```

生成签名

签名类型：

类型	含义
多次有效	有效时间内多次始终有效
单次有效	与资源URL绑定，一次有效

签名获取：

SDK 中用到的 SIGN，推荐使用 服务器端SDK，并由移动端向业务服务器请求。SIGN 的具体生成和使用请参照 访问权限 。

目录操作

创建目录

方法原型

调用此接口可在指定的 bucket 下创建目录，具体步骤如下：

1. 调用 CreateDirRequest() 实例化 CreateDirRequest 对象。
2. 调用 COSClient 的 createDir 方法，传入 CreateDirRequest，返回 CreateDirResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称
cosPath	String	是	需要创建目录的路径
biz_attr	String	否	目录绑定的属性信息，由用户维护
sign	String	是	签名信息，此处使用多次签名
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过CreateDirResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息
ctime	long(Unix时间戳)	创建时间

示例

```
CreateDirRequest createDirRequest = new CreateDirRequest();
createDirRequest.setBucket(bucket);
createDirRequest.setCosPath(cosPath);
createDirRequest.setBiz_attr(biz_attr);
createDirRequest.setSign(sign);
```

```
createDirRequest.setListener(new ICmdTaskListener() {
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        final CreateDirResult createDirResult = (CreateDirResult) cosResult;
        Log.w("TEST","目录创建成功： ret=" + createDirResult.code + "; msg=" +
createDirResult.msg
            +"ctime = " + createDirResult.ctime));
    }

    @Override
    public void onFailed(COSRequest COSRequest, final COSResult cosResult) {
        Log.w("TEST","目录创建失败： ret=" + cosResult.code + "; msg=" +
cosResult.msg);
    }
});

CreateDirResult result = cos.createDir(createDirRequest);
```

目录列表查询

方法原型

调用此接口可查询指定目录下的列表信息，具体步骤如下：

1. 调用 ListDirRequest() 实例化 ListDirRequest 对象。
2. 调用 COSClient 的 listDir 方法，传入 ListDirRequest，返回 ListDirResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称
cosPath	String	是	远程相对路径
num	int	否	返回的数目，默认为1000，最大1000
content	String	否	透传字段，首次拉取必须清空。拉取下一页，需要将前一页返回值中的context透传到参数中
prefix	String	否	前缀查询的字符串
sign	String	是	签名信息，此处使用多次签名

listener	ICmdTaskListener	否	结果回调

返回结果说明

通过ListDirResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息
content	String	透传字段
listover	boolean	标识是否还有数据， true：列举结束；false：还有数据
infos	List	列举目录列表中文件或文件夹的属性

示例

```
ListDirRequest listDirRequest = new ListDirRequest();
listDirRequest.setBucket(bucket);
listDirRequest.setCosPath(cosPath);
listDirRequest.setNum(100);
listDirRequest.setContent("");
listDirRequest.setSign(sign);
listDirRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        //查询成功
        ListDirResult listObjectResult = (ListDirResult) cosResult;
        if(listObjectResult.infos != null && listObjectResult.infos.size() > 0) {
        for(int i = 0; i < length; i++){
            String str = listObjectResult.infos.get(i);
            try {
                JSONObject jsonObject = new JSONObject(str);
                if(jsonObject.has("sha")){
                 //当前搜索的结果是文件
                }else{
                //当前搜索的结果是文件夹
                }
            } catch (JSONException e) {
              e.printStackTrace();
```

```
            }
        }
        }

        if (!listover) {
            // 存在下一页保存当前页的状态信息
            String content = result.content;
        }
    }

    @Override
    public void onFailed(COSRequest COSRequest, final COSResult cosResult) {
        Log.w("TEST",目录查询失败：ret=" + cosResult.code  + "; msg =" +
cosResult.msg);
    }
});


// 支持前缀查询
listDirRequest.setPrefix(prefix);
ListDirResult result=cos.listDir(listDirRequest);
```

目录更新

方法原型

调用此接口可更新指定目录的信息，具体步骤如下：

1. 调用 UpdateObjectRequest() 实例化 UpdateObjectRequest 对象。
2. 调用 COSClient 的 updateObject 方法，传入 UpdateObjectRequest，返回 UpdateObjectResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称
cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处使用单次签名
bizAttr	String	否	目录绑定的属性信息
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过UpdateObjectResult对象的成员变量成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息

示例

```
UpdateObjectRequest updateObjectRequest = new UpdateObjectRequest();
updateObjectRequest.setBucket(bucket);
updateObjectRequest.setCosPath(cosPath);
updateObjectRequest.setBizAttr(biz_attr);
updateObjectRequest.setSign(onceSign);
updateObjectRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        //更新成功
        Log.w("TEST", cosResult.code+" : "+cosResult.msg);
    }

     @Override
     public void onFailed(COSRequest COSRequest, COSResult cosResult) {
         //更新失败
         Log.w("TEST", cosResult.code+" : "+cosResult.msg);
     }
});

UpdateObjectResult result = cos.updateObject(updateObjectRequest);
```

目录查询

方法原型

调用此接口可查询指定目录的信息，具体步骤如下：

1. 调用 GetObjectMetadataRequest() 方法实例化 GetObjectMetadataRequest 对象。
2. 调用 COSClient 的 getObjectMetadata方法，传入 GetObjectMetadataRequest，返回 GetObjectMetadataResult对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称
cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处使用多次签名
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过GetObjectMetadataResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息
biz_attr	String	目录绑定的属性信息
ctime	long(Unix时间戳)	创建时间
mtime	long(Unix时间戳)	最后一次修改时间

示例

```
GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest();
getObjectMetadataRequest.setBucket(bucket);
getObjectMetadataRequest.setCosPath(cosPath);
getObjectMetadataRequest.setSign(sign);
getObjectMetadataRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        GetObjectMetadataResult result = (GetObjectMetadataResult) cosResult;
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("code=" + result.code + "; msg=" +result.msg + "\n");
        stringBuilder.append("ctime =" +result.ctime + "; mtime=" +result.mtime +
"\n" );
        stringBuilder.append("biz_attr=" + result.biz_attr == null ? "" : result.biz_attr );
        Log.w("TEST",stringBuilder.toString());
     }
```

```
    @Override
    public void onFailed(COSRequest cosRequest, final COSResult cosResult) {
       Log.w("TEST", cosResult.code+" : "+cosResult.msg);
     }

});

GetObjectMetadataRequest result =
cos.getObjectMetadata(getObjectMetadataRequest);
```

目录删除

方法原型

调用此接口可删除指定目录，具体步骤如下，注意只能删除空目录：

1. 调用 RemoveEmptyDirRequest() 方法实例化 RemoveEmptyDirRequest 对象。
2. 调用 COSClient 的 removeEmptyDir 方法，传入 RemoveEmptyDirRequest，返回 RemoveEmptyDirResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云 APPID
bucket	String	是	目录所属 bucket 名称
cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处使用单次签名
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过RemoveEmptyDirResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息

示例

```
RemoveEmptyDirRequest removeEmptyDirRequest = new RemoveEmptyDirRequest();
```

```
removeEmptyDirRequest.setBucket(bucket);
removeEmptyDirRequest.setCosPath(cosPath);
removeEmptyDirRequest.setSign(onceSign);
removeEmptyDirRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        final DeleteObjectResult removeEmptyDirResult = (DeleteObjectResult)
cosResult;
        Log.w("TEST","removeDir 结果： ret=" + removeEmptyDirResult.code + "; msg="
+ removeEmptyDirResult.msg );
    }

    @Override
    public void onFailed(COSRequest COSRequest, final COSResult cosResult) {
        Log.w("TEST","目录删除失败：ret=" + cosResult.code  + "; msg =" +
cosResult.msg);
    }
});

RemoveEmptyDirResult result = cos.removeEmptyDir(removeEmptyDirRequest);
```

文件操作

文件上传

方法原型

调用此接口上传指定的文件，具体步骤如下：

1. 调用 PutObjectRequest() 方法实例化 PutObjectRequest对象。
2. 调用 COSClient 的 putObject 方法，传入 PutObjectRequest，返回 PutObjectResult对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云 APPID
bucket	String	是	目录所属 bucket 名称
cosPath	String	是	远程相对路径
srcPath	String	是	本地绝对路径
slice_size	int	否	分片上传时，设置的分片大小，默认为：1M

sign	String	是	签名信息，此处使用多次签名
listener	IUploadTaskListener	否	结果回调

返回结果说明

通过PutObjectResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息
access_url	String	访问文件的URL
url	String	操作文件的url

示例

```
PutObjectRequest putObjectRequest = new PutObjectRequest();
putObjectRequest.setBucket(bucket);
putObjectRequest.setCosPath(cosPath);
putObjectRequest.setSrcPath(srcPath);
putObjectRequest.setSign(sign);

 /* 设置是否允许覆盖同名文件： "0"，允许覆盖；"1",不允许覆盖；
putObjectRequest.setInsertOnly("1");

//设置是否开启分片上传
putObjectRequest.setSliceFlag(true);//设置是否分片上传：true，分片上传;false,简单文件上
传
putObjectRequest.setSlice_size(1024*1024);//分片上传时，分片的大小
 */

putObjectRequest.setListener(new  IUploadTaskListener(){
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {

        PutObjectResult result = (PutObjectResult) cosResult;
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append(" 上传结果： ret=" + result.code + "; msg =" +result.msg +
"\n");
        stringBuilder.append(" access_url= " + result.access_url + "\n");
```

```
        stringBuilder.append(" resource_path= " + result.resource_path + "\n");
        stringBuilder.append(" url= " result.url);
        Log.w("TEST",stringBuilder.toString();
    }

    @Override
    public void onFailed(COSRequest COSRequest, final COSResult cosResult) {
        Log.w("TEST","上传出错： ret =" +cosResult.code + "; msg =" + cosResult.msg);
    }

    @Override
    public void onProgress(COSRequest cosRequest, final long currentSize, final long
totalSize) {
        float progress = (float)currentSize/totalSize;
        progress = progress *100;
        Log.w("TEST","进度：  " + (int)progress + "%");
    }
});

PutObjectResult result = cos.putObject(putObjectRequest);
```

文件更新

方法原型

调用此接口可更新指定文件信息，具体步骤如下：

1. 调用 UpdateObjectRequest() 方法实例化 UpdateObjectRequest 对象。
2. 调用 COSClient 的 updateObject 方法，传入 UpdateObjectRequest，返回 UpdateObjectResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云 APPID
bucket	String	是	目录所属 bucket 名称
cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处使用单次签名
bizAttr	String	否	目录绑定的属性信息

authority	String	否	文件操作权限，与 bucket 拥有相同的权限（COSAuthority.EINVALID），私有读写权限（COSAuthority.EWRPRIVATE），私有写公有读权限（COSAuthority.EWPRIVATERPUBLIC）
custom_headers	Map<String,String>	否	文件自定义的header: 如Cache-Control，Content-Disposition,Content-Language,x-cos-meta-。
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过UpdateObjectResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息

示例

```
UpdateObjectRequest updateObjectRequest = new UpdateObjectRequest();
updateObjectRequest.setBucket(bucket);
updateObjectRequest.setCosPath(cosPath);
updateObjectRequest.setBizAttr(biz_attr);
updateObjectRequest.setAuthority(authority);
updateObjectRequest.setCustomHeader(customer);
updateObjectRequest.setSign(onceSign);
updateObjectRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        Log.w("TEST", cosResult.code+" : "+cosResult.msg);
    }

    @Override
    public void onFailed(COSRequest COSRequest, COSResult cosResult) {
                    Log.w("TEST", cosResult.code+" : "+cosResult.msg);
    }
});

UpdateObjectResult result= cos.updateObject(updateObjectRequest);
```

文件查询

方法原型

调用此接口可查询指定文件信息，具体步骤如下：

1. 调用 GetObjectMetadataRequest() 方法实例化 GetObjectMetadataRequest 对象。
2. 调用 COSClien t的 getObjectMetadata 方法，传入 GetObjectMetadataRequest，返回 GetObjectMetadataResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称
cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处使用多次签名
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过GetObjectMetadataResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息
biz_attr	String	目录绑定的属性信息
ctime	long(Unix时间戳)	创建时间
mtime	long(Unix时间戳)	最后一次修改时间
sha	String	文件的sha值
customs_headers	Map<String,String>	文件的头部属性
filelen	int	文件的长度大小

示例

```
GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest();
getObjectMetadataRequest.setBucket(bucket);
getObjectMetadataRequest.setCosPath(cosPath);
getObjectMetadataRequest.setSign(sign);
getObjectMetadataRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        GetObjectMetadataResult result = (GetObjectMetadataResult) cosResult;
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("code=" + result.code + "; msg=" +result.msg + "\n");
        stringBuilder.append("ctime =" +result.ctime + "; mtime=" +result.mtime +
"\n" );
        stringBuilder.append("biz_attr=" + result.biz_attr == null ? "" : result.biz_attr );
        stringBuilder.append("sha=" + result.sha);
        Log.w("TEST",stringBuilder.toString());
    }

    @Override
    public void onFailed(COSRequest cosRequest, final COSResult cosResult) {
        Log.w("TEST", cosResult.code+" : "+cosResult.msg);
    }

});

GetObjectMetadataRequest
result=cos.getObjectMetadata(getObjectMetadataRequest);
```

文件删除

方法原型

调用此接口可删除指定文件，具体步骤如下：

1. 调用 DeleteObjectRequest() 方法实例化 DeleteObjectRequest 对象。
2. 调用 COSClient 的 deleteObject 方法，传入 DeleteObjectRequest，回 DeleteObjectResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
appid	String	是	腾讯云APPID
bucket	String	是	目录所属bucket 名称

cosPath	String	是	远程相对路径
sign	String	是	签名信息，此处单次签名
listener	ICmdTaskListener	否	结果回调

返回结果说明

通过DeleteObjectResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息

示例

```
DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest();
deleteObjectRequest.setBucket(bucket);
deleteObjectRequest.setCosPath(cosPath);
deleteObjectRequest.setSign(onceSign);
deleteObjectRequest.setListener(new ICmdTaskListener() {
    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        Log.w("TEST", cosResult.code+" : "+cosResult.msg);
    }

    @Override
    public void onFailed(COSRequest COSRequest, COSResult cosResult) {
       Log.w("TEST", cosResult.code+" : "+cosResult.msg);
   }
});

DeleteObjectResult result = cos.deleteObject(deleteObjectRequest);
```

文件下载

方法原型

调用此接口可下载指定文件，具体步骤如下：

1. 调用 GetObjectRequest(String downloadURl,String savePath) 方法实例化 GetObjectRequest 对象。
2. 调用 COSClient 的 getObject 方法，传入 GetObjectRequest,返回 GetObjectResult 对象。

参数说明

参数名称	类型	是否必填	参数描述
downloadUrl	String	是	下载文件的URL
localPath	String	是	本地保存文件的绝对地址
sign	String	否	签名信息：开启了防盗链则需要，否则不需要
listener	IDownloadTaskListener	否	结果回调

返回结果说明

通过GetObjectResult对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
code	String	结果码
msg	String	详细结果信息

示例

```
GetObjectRequest getObjectRequest = new
GetObjectRequest(downloadURI,savePath);
getObjectRequest.setSign(null);
getObjectRequest.setListener(new IDownloadTaskListener() {
    @Override
    public void onProgress(COSRequest cosRequest, final long currentSize, final long
totalSize) {
        float progress =  currentSize / (float)totalSize;
        progress = progress * 100;
        progressText.setText("progress =" + (int) (progress) + "%");
        Log.w("TEST", "progress =" + (int) (progress) + "%");

    }

    @Override
    public void onSuccess(COSRequest cosRequest, COSResult cosResult) {
        Log.w("TEST","code =" + cosResult.code + "; msg =" + cosResult.msg);
    }
```

```
    @Override
    public void onFailed(COSRequest COSRequest, COSResult cosResult) {
                    Log.w("TEST","code =" + cosResult.code + "; msg =" +
cosResult.msg);
    }
});


GetObjectResul result = cos.getObject(getObjectRequest);
```

C# SDK

最近更新时间：2020-02-12 13:46:11

> **注意**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

相关资源

C# SDK github项目下载地址

开发准备

1. SDK 依赖 C# 4.0版本及以上， 推荐使用相同的版本。
2. 从控制台获取 APP ID、SecretID、SecretKey。
3. 修改园区，CosCloud.cs文件内的URL定义，例如华东： `http://sh.file.myqcloud.com/files/v2/` 华北： `http://tj.file.myqcloud.com/files/v2/` 华南： `http://gz.file.myqcloud.com/files/v2/`
   （本版本SDK基于JSON API封装组成）

SDK 配置

直接下载 github 上提供的源代码，集成到开发环境。
若需 HTTPS 支持，则将 cos_dotnet_sdk/Api/CosCloud.cs 文件中变量 COSAPI_CGI_URL 中的 HTTP 修改为 HTTPS 即可。

生成签名

多次有效签名

方法原型

```
public static string Signature(int appId, string secretId, string secretKey, long expired,
string bucketName)
```

参数说明

参数名	类型	必填	参数描述
appId	int	是	AppId
secretId	string	是	Secret Id

secretKey	string	是	Secret Key，以上三项从 控制台 获取。
expired	long	是	过期时间，Unix 时间戳
bucketName	string	是	bucket 名称，bucket 创建参见 创建Bucket

示例

```
var sign = Sign.Signature(appId, secretId, secretKey, expired, bucketName);
```

单次有效签名

方法原型

```
public static string SignatureOnce(int appId, string secretId, string secretKey, string remotePath, string bucketName)
```

参数说明

参数名	类型	必填	参数描述
appId	int	是	AppId
secretId	string	是	Secret Id
secretKey	string	是	Secret Key，以上三项从 控制台 获取。
bucketName	string	是	bucket名称，bucket创建参见 创建Bucket
remotePath	string	是	文件唯一的标识，格式 /appid/bucketname/filepath/filename，其中 /filepath/filename为文件在此 bucketname 下的全路径，

示例

```
var sign = Sign.SignatureOnce(appId, secretId, secretKey,remotePath, bucketName);
```

更多签名详细说明，请参考 权限控制 。

目录操作

创建目录

接口说明：用于目录的创建，可以通过此接口在指定 bucket 下创建目录。

方法原型

```
public string CreateFolder(string bucketName, string remotePath, Dictionary<string, string> parameterDic = null)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	需要创建目录的全路径
parameterDic	string	否	属性字典

属性字典说明

参数名	类型	必填	参数描述
biz_attr	string	否	目录属性

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为 0
message	String	是	错误信息
data	Array	否	返回数据，请参考《Restful API 创建目录》

示例

```
var createFolderParasDic = new Dictionary<string, string>();
    createFolderParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");
var result = cos.CreateFolder(bucketName, folder, createFolderParasDic);
```

目录更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
public string UpdateFolder(string bucketName, string remotePath, Dictionary<string, string>  parameterDic = null)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	需要创建目录的全路径
parameterDic	string	是	目录更新参数字典

目录更新参数字典

参数名	类型	必填	参数描述
biz_attr	string	否	目录属性

返回结果说明

参数名	类型	必选	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var updateParasDic = new Dictionary<string, string>();
    updateParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");
var result = cos.UpdateFolder(bucketName, folder, updateParasDic);
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```
public string GetFolderStat(string bucketName, string remotePath)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	目录的全路径

返回结果说明

参数名	类型	必选	参数描述
code	Int	是	错误码，成功时为 0
message	String	是	错误信息
data	Array	否	目录属性数据，请参考《Restful API 目录查询》

示例

```
var result = cos.GetFolderStat(bucketName, folder);
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
public string DeleteFolder(string bucketName, string remotePath)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	目录的全路径

返回结果说明

参数名	类型	必选	参数描述
code	Int	是	错误码，成功时为0

message	String	是	错误信息

示例

```
var result = cos.DeleteFolder(bucketName, folder);
```

列举目录下文件&目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
public string GetFolderList(string bucketName, string remotePath, Dictionary<string, string>  parameterDic = null)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	目录的全路径
parameterDic	Dictionary<string, string>	是	目录列表参数字典

目录列表参数字典

参数名	类型	必填	参数描述
num	string	是	要查询的目录/文件数量，最大1000
listFlag	string	否	listFlag：大于0返回全部数据，否则返回部分数据
context	String	否	透传字段，用于翻页，前端不需理解，需要往后翻页则透传回来
prefix	string	否	搜索前缀

返回结果说明

参数名	类型	必带	参数描述

code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 目录列表》

示例

```
var folderlistParasDic = new Dictionary<string, string>();
    folderlistParasDic.Add(CosParameters.PARA_NUM,"100");
var result = cos.GetFolderList(bucketName, folder, folderlistParasDic);
```

文件操作

文件上传

接口说明：文件上传的统一接口，对于大于8m的文件，内部会通过多次分片的方式进行文件上传。没有断点续传功能，需要自己用分片上传接口实现。

方法原型

```
public string UploadFile(string bucketName, string remotePath, string localPath,
Dictionary<string, string>  parameterDic = null)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称，bucket创建参见 创建 Bucket
remotePath	string	是	文件在服务端的全路径
localPath	string	是	文件本地路径
parameterDic	Dictionary<string, string>	否	文件上传参数字典

文件上传参数字典

参数名	类型	必填	参数描述

biz_attr	string	否	文件属性
slice_size	string	否	可选取值为：64*1024* *512*1024，1*1024*1024（默认），*2*1024*1024，*3*1024*1024

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 创建文件》

示例

```
var uploadParasDic = new Dictionary<string, string>();
uploadParasDic.Add(CosParameters.PARA_BIZ_ATTR,"file attribute");
uploadParasDic.Add(CosParameters.PARA_INSERT_ONLY,"0");
uploadParasDic.Add(CosParameters.PARA_SLICE_SIZE,SLICE_SIZE.SLIZE_SIZE_3M.ToStr
ing());
result = cos.UploadFile(bucketName, remotePath, localPath, uploadParasDic);
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
public string UpdateFile(string bucketName, string remotePath, Dictionary<string,
string> parameterDic = null)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	文件在 COS 的全路径
parameterDic	string	否	文件更新参数字典

文件更新参数字典

参数名	类型	必填	参数描述
biz_attr	string	否	待更新的文件属性信息
authority	string	否	eInvalid（继承Bucket的读写权限）；eWRPrivate（私有读写）；eWPrivateRPublic（公有读私有写）
Cache-Control	string	否	指定请求和响应遵循的缓存机制，如：no-cache；max-age=200
Content-Type	string	否	返回内容的 MIME 类型，如：text/html
Content-Disposition	string	否	控制用户请求所得的内容存为一个文件的时候提供一个默认的文件名，如：attachment; filename="fname.ext"
Content-Language	string	否	使用的语言，如：zh-CN
x-cos-meta-自定义内容	string	否	表示以“x-cos-meta-”名字开头的参数，用户按照自身业务场景，设置需要在 Header 中传输什么参数

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var optionParasDic = new Dictionary<string, string>();
    optionParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");

optionParasDic.Add(CosParameters.PARA_AUTHORITY,AUTHORITY.AUTHORITY_PRIVATE
PUBLIC);
    optionParasDic.Add(CosParameters.PARA_CACHE_CONTROL,"no");
    optionParasDic.Add(CosParameters.PARA_CONTENT_TYPE,"application/text");

optionParasDic.Add(CosParameters.PARA_CONTENT_DISPOSITION,"filename=\"test.pdf
\"");
    optionParasDic.Add(CosParameters.PARA_CONTENT_LANGUAGE,"en");
```

```
    optionParasDic.Add("x-cos-meta-test","test");
result = cos.UpdateFile(bucketName, remotePath, optionParasDic);
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

```
public string GetFileStat(string bucketName, string remotePath)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称
remotePath	string	是	文件在Cos的全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件属性数据，请参考《Restful API 文件查询》

示例

```
var result = cos.GetFileStat(bucketName, remotePath);
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```
public string DeleteFile(string bucketName, string remotePath)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var result = cos.DeleteFile(bucketName, remotePath);
```

分片上传list

接口说明：查询分片上传的情况

方法原型

```
public string UploadSliceList(string bucketName, string remotePath)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

上传完成data的数据说明

参数名	类型	必带	参数描述
url	String	是	操作文件的 url
access_url	String	是	生成的文件下载 url
source_url	String	是	源地址
resource_path	String	是	资源路径，格式： /appid/bucket/xxx
preview_url	String	否	预览地址

上传未完成data的数据说明

参数名	类型	必带	参数描述
session	String	是	init返回的标识
filesize	Int	是	文件大小
slice_size	Int	是	分片大小（64K-3M）大于1M 必须为1M整数倍
sha	String	否	文件的全文sha值，init时若已带sha值，则返回该值
listparts	Json Array	是	已上传完成的分片，形如：[{“offset”:0,“datalen”:1024}, {}, {}].

示例

```
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadInit(bucketName, remotePath, localPath, fileSha, bizAttribute,
sliceSize, insertOnly);
```

分片上传init

接口说明：分片上传的第一次握手,如果上一次分片上传未完成，会返回{"code":-4019,"message":"_ERROR_FILE_NOT_FINISH_UPLOAD"}init
接口说明：分片上传的第一次握手

方法原型

```
public string SliceUploadInit(string bucketName, string remotePath, string localPath,
string fileSha,string bizAttribute = "", int sliceSize = SLICE_SIZE.SLIZE_SIZE_1M, int
insertOnly = 1)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket 名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	string	是	文件的 SHA 值

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	类型	必带	参数描述
session	string	否	(非秒传的大部分情况会有) 唯一标识此文件传输过程的 ID
slice_size	Int	否	(非秒传大部分情况下会有) 分片大小
serial_upload	int	否	(非秒传大部分情况下会有) 1：只支持串行分片上传其它：支持并行分片

示例

```
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadInit(bucketName, remotePath, localPath, fileSha, bizAttribute,
sliceSize, insertOnly);
```

分片上传

接口说明：分片上传数据

方法原型

```
public string SliceUploadData(string bucketName, string remotePath, string localPath,
string fileSha, string session, long offset,int sliceSise,string sign)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	string	是	文件的Sha值
session	string	是	唯一标识此文件传输过程的id
offset	int	是	分片的偏移量
sliceSize	int	是	分片的大小
sign	string	是	签名（多次）

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	类型	必带	参数描述
session	string	是	(非秒传的大部分情况会有)　唯一标识此文件传输过程的 ID
offset	Int	是	当前分片的 offset

datalen	int	是	分片长度 slice_size，返回的 datalen 就是当前分片的大小
serial_upload	int	否	(非秒传大部分情况下会有) 1：只支持串行分片上传其它：支持并行分片

示例

```
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadDataInit(bucketName, remotePath, localPath, fileSha, session,
offset, sliceSize,sign);
```

分片上传 finish

接口说明：告诉COS所有分片上传成功。

方法原型

```
public string SliceUploadFinish(string bucketName, string remotePath, string localPath,
string fileSha, string session)
```

参数说明

参数名	类型	必填	参数描述
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	string	是	文件的Sha值
session	string	是	唯一标识此文件传输过程的id

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	类型	必带	参数描述
session	string	是	(非秒传的大部分情况会有) 唯一标识此文件传输过程的id
offset	Int	是	当前分片的offset
datalen	int	是	分片文件长度

示例

```
result = SliceUploadFinish(bucketName,remotePath,localPath,fileSha,
sessionbizAttribute, sliceSize, insertOnly);
```

C++ SDK

最近更新时间：2020-02-12 13:52:48

> ⚠ 注意
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

相关资源

Github 项目下载地址

开发环境

1. 安装 openssl 的库和头文件。
2. 安装 curl 的库和头文件。
3. 安装 jsoncpp 的库和头文件。
4. 安装 boost 的库和头文件。
5. 安装 cmake 工具。
6. 登录 腾讯云控制台 获取 APPID、SecretID、SecretKey。

> ⚠ 注意
> 1. 本 SDK 仅适用于 Linux 系统和编译环境，暂不支持 Windows 环境。
> 2. sdk 中提供了curl 和 jsoncpp 的库以及头文件，以上库编译好后替换掉 sdk 中相应的库和头文件即可，如果以上库已经安装到系统里，也可删除 sdk 中相应的库和头文件。
> 3. curl 默认不支持多线程环境，如果项目使用多线程，在编译 curl 执行 configure 时需指定 --enable-ares 参数来开启异步 DNS 解析，依赖 c-ares 库，如果系统没有，可到 c-ares 下载安装。
> 4. jsoncpp 的 1.y.x 版本需要 c++11 的支持，如果编译器不支持，可以换成 0.y.x 版本。（本版本 SDK 基于 JSON API 封装组成）

SDK 配置

直接下载 github 上提供的源代码，集成到您的开发环境。
执行下面的命令：

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

```
cd build
cmake ..
make
```

cos_demo.cpp 里面有常见 API 的例子。生成的 cos_demo 可以直接运行，生成的静态库名称为：libcossdk.a。生成的 libcossdk.a 放到您自己的工程里 lib 路径下，include 目录拷贝到自己的工程的 include 路径下。

生成签名

多次有效签名

方法原型

```
static string AppSignMuti(const uint64_t appId,
                          const string &secretId,
                          const string &secretKey,
                          const uint64_t expired_time,
                          const string &bucketName);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
appId	uint64_t	是	无	项目 App ID
secretId	String	是	无	用户 Secret ID
secretKey	String	是	无	用户 SecretKey
expired_time	uint64_t	否	无	过期时间，Unix 时间戳
bucketName	String	否	无	bucket 名称

返回结果说明

返回值：签名字符串

示例

```
uint64_t expired = time(NULL) + 60;
```

```
string sign =
Auth::AppSignMuti(10000000,"SecretId","SecretKey",expired,"bucketName");
```

单次有效签名

方法原型

```
static string AppSignOnce(const uint64_t appId,
                          const string &secretId,
                          const string &secretKey,
                          const string &path,
                          const string &bucketName);
```

参数说明

参数名	类型	必须	默认值	参数描述
appId	uint64_t	是	无	项目 App ID
secretId	String	是	无	项目 SecretID
secretKey	String	是	无	项目 SecretKey
bucketName	String	否	无	bucket名称
path	String	是	无	文件路径，以斜杠开头，例如/filepath/filename，为文件在此 bucketname 下的全路径

返回值说明

返回值：签名字符串

示例

```
string path= "/myFloder/myFile.rar";
sign = Auth::AppSignOnce(10000000, "SecretId", "SecretKey", path, bucketName);
```

初始化操作

初始化

接口说明：在使用 COS 操作之前，需要首先进行 COS 系统参数的设置，然后分别创建 CosConfig 以及 CosAPI 对象，COS 的操作都是基于 CosAPI 对象进行的。

配置文件

```
"AppID":********,
"SecretID":"**********************************",
"SecretKey":"**********************************",
"Region":"sh",  //COS区域, 华东园区：sh ； 华南园区：gz ； 华北园区：tj ； 上传和下载域名均是跟此有关系，因此一定要保证正确
"SignExpiredTime":360, //签名超时时间
"CurlConnectTimeoutInms":180,  //http超时时间
"CurlGlobalConnectTimeoutInms":360, //
"UploadSliceSize":1048576, //文件分片大小，可选值有524288、1048576、2097152、3145728
"IsUploadTakeSha":0, //文件上传时是否携带文件sha值，0：不携带，1：携带
"DownloadDomainType":2, //下载域名类型,1:cdn,2:cos,3:innercos,4:自定义域名
"SelfDomain":"",//自定义域名
"UploadThreadPoolSize":5, //单文件分片上传线程池大小
"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
```

COS API 对象构造原型

```
CosConfig(const string& config_file);
CosAPI(CosConfig& config);
```

目录操作

创建目录

接口说明：用于目录的创建，调用者可以通过此接口在指定 bucket 下创建目录。

方法原型

```
string CosAPI::FolderCreate(FolderCreateReq& request);
```

参数说明

参数名	类型	默认值	参数描述

request	FolderCreateReq	无	目录创建请求类型

request成员	类型	默认值	设置方法	描述
bucket	string	无	构造函数	bucket 名称
cosPath	string	无	构造函数	上传的目标路径
bizAttr	string	空字符串	构造函数	文件夹属性

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	返回码，成功时为 0
message	String	描述信息
data	Array	返回数据，请参考《Restful API 创建目录》

示例

```
FolderCreateReq folderCreateReq(bucket,folder,folder_biz_attr);
string rsp = cos.FolderCreate(folderCreateReq);
```

目录更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
string FolderUpdate(FolderUpdateReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FolderUpd	无	目录更新请求类型

	ateReq		

参数名	类型	默认值	设置方法	参数描述
bucket	String	无	构造函数	bucket名称
cosPath	String	无	构造函数	目录的全路径
bizAttr	string	空	构造函数	文件夹属性

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	返回码，成功时为 0
message	String	描述信息
data	Array	返回数据，请参考《Restful API 目录更新》

示例

```
FolderUpdateReq folderUpdateReq(bucket,folder, folder_biz_attr);
string rsp = cos.FolderUpdate(folderUpdateReq);
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```
string CosAPI::FolderStat(FolderStatReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FolderStatReq	无	目录查询请求类型

参数名	类型	默认值	设置方法	参数描述
bucket	String	无	构造函数	bucket名称
cosPath	String	无	构造函数	目录的全路径

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	返回码，成功时为0
message	String	描述信息

示例

```
FolderStatReq folderStatReq(bucket,folder);
string rsp = cos.FolderStat(folderStatReq);
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
string CosAPI::FolderDelete(FolderDeleteReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FolderDeleteReq	无	目录删除请求类型

参数名	类型	默认值	设置方法	参数描述
bucket	String	无	构造函数	bucket名称
cosPath	String	无	构造函数	目录的全路径

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	返回码，成功时为 0
message	String	描述信息

示例

```
FolderDeleteReq folderDeleteReq(bucket,folder);
string rsp = cos.FolderDelete(folderDeleteReq);
```

目录列表

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
string CosAPI::FolderList(FolderListReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FolderListReq	无	目录列表请求类型

参数名	类型	默认值	设置方法	参数描述
bucket	String	无	构造函数	bucket 名称
cosPath	String	无	构造函数	目录的全路径
listNum	int	1000	构造函数	查询数目，最大为1000
context	string	空字符串	构造函数	查询上下文。查看第一页，则传空字符串；若需翻页，需要将前一页查询响应中的 context 设置到参数中

返回结果说明

通过函数返回值返回请求结果的 json 字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	返回码，成功时为 0
message	String	是	描述信息
data	Array	是	返回数据，请参考《Restful API 目录列表》

示例

```
FolderListReq folderListReq(bucket,folder);
string rsp = cos.FolderList(folderListReq);
```

文件操作

文件上传

接口说明：文件上传，包括单文件上传和分片上传（大于8M的文件需要通过将文件内容进行分片上传）。

方法原型

```
string CosAPI::FileUpload(FileUploadReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FileUploadReq	无	文件上传请求类型

参数名	类型	默认值	设置方法	参数描述
bucket	String	无	构造函数	bucket名字
srcPath	String	无	构造函数	待上传的本地文件路径
cosPath	String	无	构造函数	文件的全路径
bizAttr	string	无	setBizAttr()	文件属性

sliceSize	int	1048576	setSliceSize()	分片大小，可选值512K/1M/2M/3M；默认1M，均需转换为字节数值

返回结果说明

通过函数返回值返回请求结果的 json 字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	返回码，成功时为 0
message	String	是	描述信息
data	Array	否	返回数据，请参考《Restful API 文件上传》

示例

```
FileUploadReq fileUploadReq(bucket, srcPath, dstPath);
string rsp = cos.FileUpload(fileUploadReq);
```

文件更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
string CosAPI::FileUpdate(FileUpdateReq& request);
```

参数说明

参数名	类型	默认值	参数描述
request	FileUpdateReq	无	文件更新请求类型

参数名	类型	是否必填	设置方法	参数描述
bucket	String	是	构造函数	bucket 名称
cosPath	String	是	构造函数	文件在对象存储服务端的路径
bizAttr	string	否	setBizAttr(	文件属性

			)	
authority	string	否	setAuthority()	文件权限，默认是继承 bucket 的权限合法取值: eInvalid(继承bucket), eWRPrivate(私有读写), eWPrivateRPublic(私有写, 公有读)
forbid	int	否	setForbid()	文件封禁标志
custom_header	map	否	setCustomHeader()	用户自定义属性

custom_header说明

参数名	类型	是否必填	参数描述
Cache-Control	string	否	参见 HTTP 的 Cache-Control
Content-Type	string	否	参见 HTTP 的 Content-Type
Content-Disposition	string	否	参见 HTTP 的 Content-Language
Content-Language	string	否	参见 HTTP 的 Content-Disposition
Content-Encoding	string	否	参见 HTTP 的 Content-Encoding
x-cos-meta-	string	否	自定义 HTTP 头，参数必须以 x-cos-meta-开头，值由用户定义，可设置多个

tips: custom_header 是整体覆盖式更新，即更新属性可以选择其中的某几个，如果本次只更新其中的某几个，其他的都会被抹掉。

返回结果说明

通过函数返回值返回请求结果的json字符串：

参数名	类型	是否必然返回	参数描述
code	Int	是	返回码，成功时为 0
message	Strin	是	描述信息

	g	

示例

```
FileUpdateReq fileUpdateReq(bucket,dstpath);
fileUpdateReq.setBizAttr(file_biz_attr);
fileUpdateReq.setAuthority(auth);
fileUpdateReq.setForbid(0);
fileUpdateReq.setCustomHeader(custom_header);
string rsp = cos.FileUpdate(fileUpdateReq);
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

```
string CosAPI::FileStat(FileStatReq& request)
```

参数说明

参数名	类型	默认值	参数描述
request	FileStatReq	无	文件查询请求类型

FileStatReq

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket 名称
cosPath	String	是	无	文件在对象存储服务端的全路径

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为 0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 文件查询》

示例

```
FileStatReq fileStatReq(bucket,dstpath);
string rsp = cos.FileStat(fileStatReq);
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```
FileDeleteReq fileDeleteReq(bucket,dstpath);
string rsp = cos.FileDelete(fileDeleteReq);
```

参数说明

参数名	类型	默认值	参数描述
request	FileDeleteReq	无	文件删除请求类型

FolderDeleteReq

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket 名称
cosPath	String	是	无	文件在对象存储服务端的全路径

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	返回码，成功时为 0
message	String	是	描述信息

示例

```
FileDeleteReq fileDeleteReq(bucket,dstpath);
string rsp = cos.FileDelete(fileDeleteReq);
```

iOS SDK

最近更新时间：2019-11-27 09:16:15

温馨提示

请注意这是历史版本（V4，基于 JSON API 封装的 SDK），**已经不再推荐使用**。
对于新接入SDK的用户，我们推荐使用最新的V5版本基于 XML API 封装的 SDK 。如果因为种种原因确实仍然需要使用基于 JSON API 封装的 SDK ,那么推荐使用我们基于历史版本重构后的基于 JSON API 封装的 SDK 。

开发准备

SDK 获取

对象存储服务的 iOS SDK 的下载地址：iOS SDK
更多示例可参考Demo：iOS Demo
（本版本 SDK 基于 JSON API 封装组成）

开发准备

- iOS 8.0+;
- 手机必须要有网络（GPRS、3G或Wifi网络等）；
- 从控制台获取APP ID、SecretID、SecretKey。

SDK 配置

SDK 导入

使用Cocoapods导入

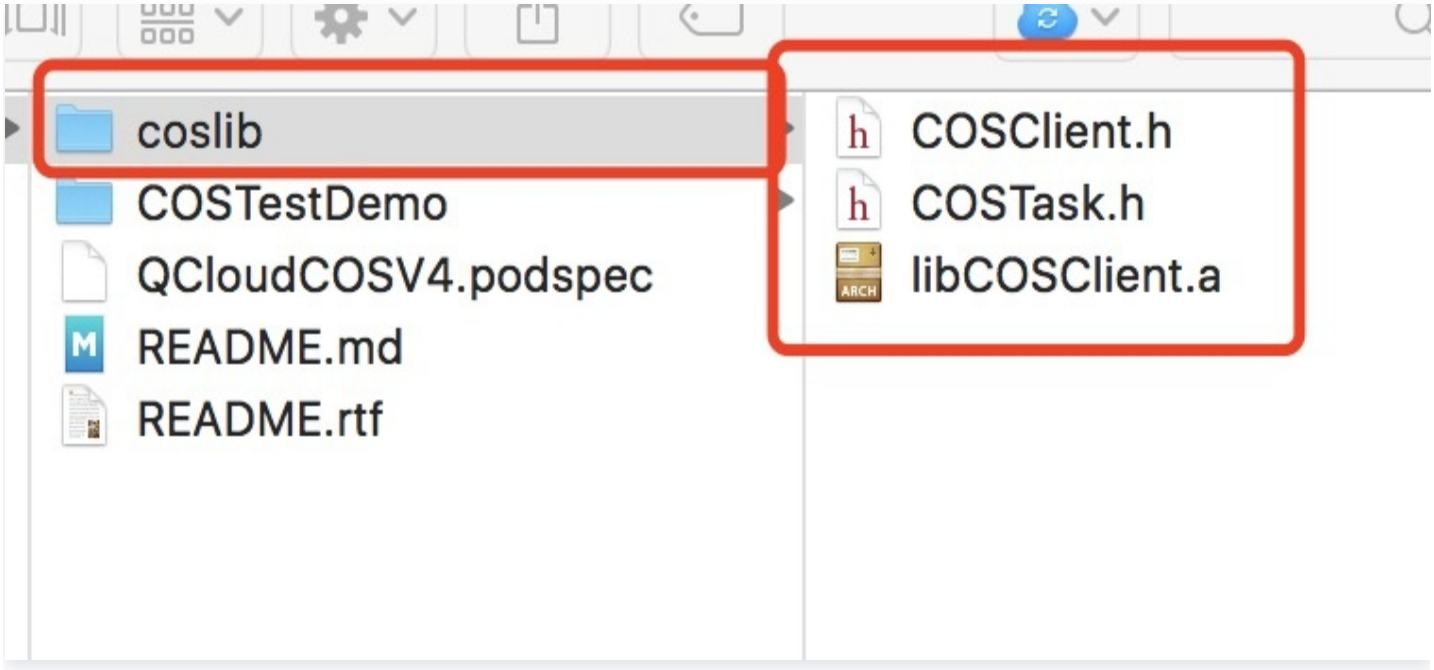
在Podfile文件中使用：

```
pod "QCloudCOSV4"
```

使用静态库导入

```
git clone https://github.com/tencentyun/COS_iOS_SDK.git
```

将目录 coslib 下面的文件拖入到工程中即可：

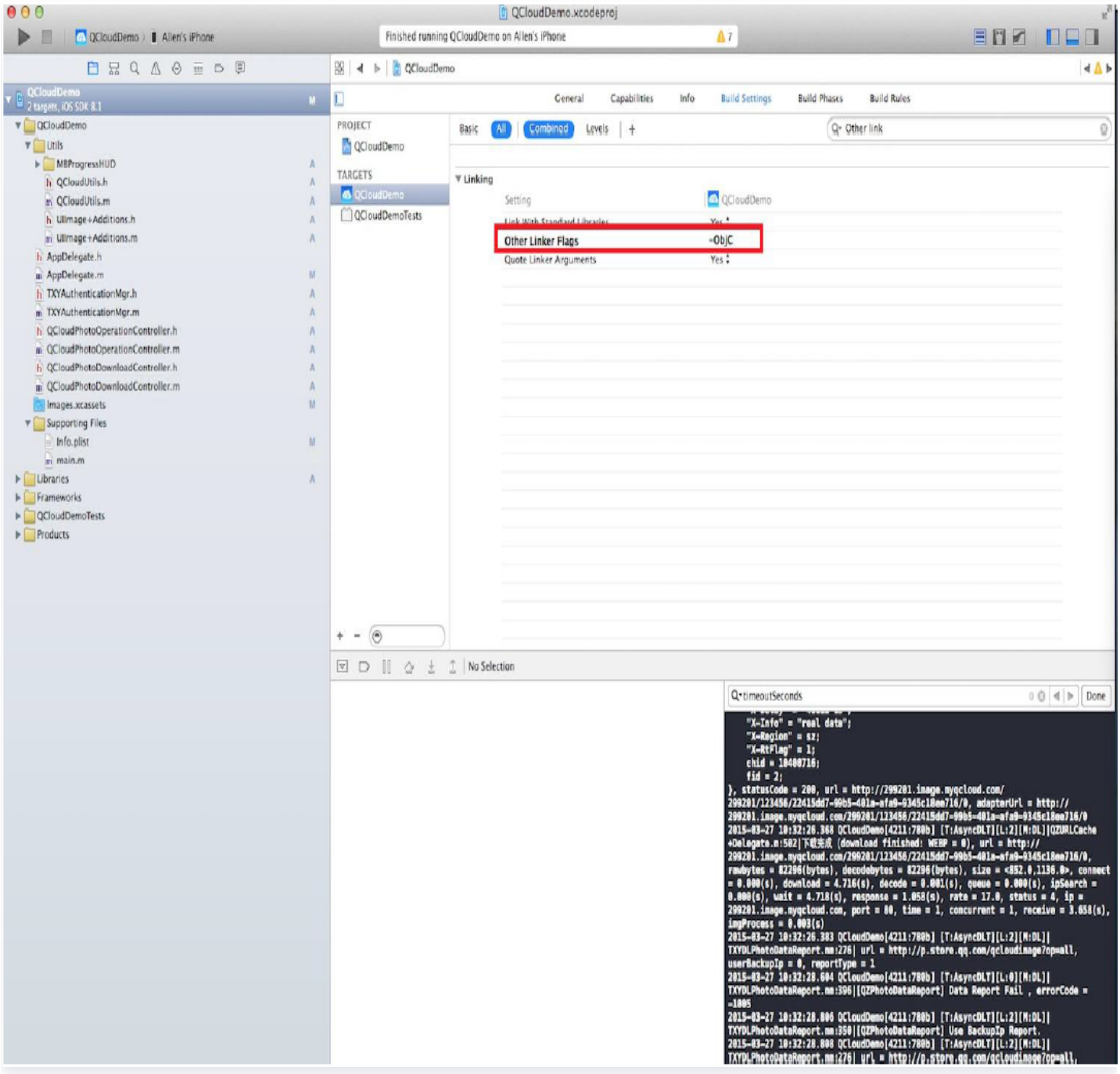


将目录 coslib 下面的文件拖入到工程拖入工程目录，Xcode 会自动将其加入链接库列表中。
并添加以下依赖库：

- CoreTelephony.framework
- Foundation.framework
- SystemConfiguration.framework

工程配置

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



在工程info.plist文件中添加App Transport Security Settings 类型，然后在App Transport Security Settings下添加Allow Arbitrary Loads 类型Boolean,值设为YES。

如果想要在程序中使用HTTPS协议，请做如下设置：

```
COSClient *client= [[COSClient alloc] initWithAppId:appId
withRegion:@"sh";//@“sh”bucket所在机房
[client openHttpsRequest:YES];
```

初始化

引入上传 SDK 的头文件 *COSClient .h*，使用目录操作时，需要先实例化 COSClient 对象。

方法原型

```
- (instancetype)initWithAppId:(NSString*)appId  withRegion:(NSString *)region;
```

参数说明

参数名称	类型	是否必填	说明
appId	NSString *	是	项目ID，即APP ID。
region	NSString *	是	bucket被创建的时候机房区域，例如华东园区：“sh”，华南园区："gz"，华北园区："tj"

示例

```
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:@“sh”];
```

快速入门

这里演示的上传和下载的基本流程，更多细节可以参考demo；在进行这一步之前必须在腾讯云控制台上申请COS业务的appid；

STEP－1 初始化COSClient

示例

```
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig instance].region];
```

STEP－2 上传文件

在这里我们假设您已经申请了自己业务bucket。SDK所有的任务对应了相应的task，只要把相应的task参数传递给client，就可以完成相应的动作；

示例

```
    COSObjectPutTask *task = [COSObjectPutTask new];
```

```
    task.filePath = path;
    task.fileName = fileName;
    task.bucket = bucket;
    task.attrs = @"customAttribute";
    task.directory = dir;
    task.insertOnly = YES;
    task.sign = _sign;
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        if (resp.retCode == 0) {
         //sucess
        }else{}
    };
    client.progressHandler = ^(NSInteger bytesWritten,NSInteger
totalBytesWritten,NSInteger totalBytesExpectedToWrite){
          //progress
    };
    [client putObject:task];
```

STEP – 3 下载文件

示例

```
     COSObjectGetTask *cm = [[COSObjectGetTask alloc] initWithUrl:imgUrl.text];
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        //
    };
    client.downloadProgressHandler = ^(int64_t receiveLength,int64_t contentLength){
    };
    [client getObjectRequest:cm];
```

生成签名

签名类型：

类型	含义
多次有效	有效时间内多次始终有效
单次有效	与资源URL绑定，一次有效

签名获取：

移动端 SDK 中用到的签名，推荐使用 服务器端SDK，并由移动端向业务服务器请求。

目录操作

目录创建

方法原型

通过此接口在指定的 bucket 下创建目录，具体步骤如下：

1. 实例化 COSCreateDirCommand 对象；
2. 调用 COSClient 的 createDirRequest 方法，将 COSCreateDirCommand 对象传入。
3. 通过COSCreatDirTaskRsp 的对象返回结果

参数说明

参数名称	类型	是否必填	说明
dir	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
attrs	NSString *	否	用户自定义属性

返回结果说明

通过COSCreatDirTaskRsp 的对象返回结果

属性名称	类型	说明
retCode	int	任务描述代码
descMsg	NSString *	任务描述信息

示例

```
    COSCreateDirCommand *cm = [COSCreateDirCommand new];
    cm.directory = dir;
    cm.bucket = bucket;
    cm.sign = _sign;
    cm.attrs = @"dirTest";
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
```

```
    if (resp.retCode == 0) {
        //sucess
    }else{
        //fail
    }
};
[client createDir:cm];
```

目录属性更新

方法原型

通过调用此接口更新目录的自定义属性，具体步骤如下：

1. 实例化 COSUpdateDirCommand 对象；
2. 调用 COSClient 的 updateDirRequest 方法，将 COSUpdateDirCommand 对象传入；
3. 通过COSUpdateDirTaskRsp 对象返回结果

参数说明

参数名称	类型	是否必填	说明
dir	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
attrs	NSString *	否	用户自定义属性

返回结果说明

通过COSUpdateDirTaskRsp 的对象返回结果

属性名称	类型	说明
retCode	int	任务描述代码
descMsg	NSString *	任务描述信息

示例

```
COSUpdateDirCommand *cm = [COSUpdateDirCommand new];
cm.directory = dir;
cm.bucket = bucket;
```

```
    cm.sign = _sign;//本业务使用一次性签名
    cm.attrs = @"dirTest";
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        if (resp.retCode == 0) {
            //sucess
        }else{}
    };
    [client updateDir:cm];
```

目录属性查询

方法原型

通过调用此接口来查询目录的详细属性，具体步骤如下：

1. 实例化 COSDirmMetaCommand 对象；
2. 调用 COSClient 的 getDirMetaData 方法，将 COSDirmMetaCommand 对象传入；
3. 通过COSDirMetaTaskRsp的对象返回结果信息；

参数说明

参数名称	类型	是否必填	说明
dir	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名

返回结果说明

通过COSDirmMetaCommand的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码
descMsg	NSString *	任务描述信息
data	NSDictionary *	任务描述信息

示例

```
    COSDirmMetaCommand *cm = [COSDirmMetaCommand new];
    cm.directory = dir;
    cm.bucket = bucket;
    cm.sign = sign;
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        if (resp.retCode == 0) {
            //sucess
        }else{}
    };
    [client getDirMetaData:cm];
```

目录删除

方法原型

调用此接口，进行指定 bucket 下目录的删除，如果目录中存在有效文件或目录，将不能删除。具体步骤如下：

1. 实例化 COSDeleteDirCommand 对象；
2. 调用 COSClient 的 deleteDirRequest 命令，传入 COSDeleteDirCommand 对象；
3. 通过COSdeleteDirTaskRsp 的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
dir	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名

返回结果说明

通过COSdeleteDirTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码
descMsg	NSString *	任务描述信息

示例

```
COSDeleteDirCommand *cm = [COSDeleteDirCommand new];
cm.directory = dir;
cm.bucket = bucket;
cm.sign = _oneSign;//删除使用一次性签名
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
      if (resp.retCode == 0) {
            //sucess;
      }else{}
};
[client deleteDir:cm];
```

目录列表

方法原型

调用此接口可以列出 bucket 中，指定目录下的文件、目录信息，具体步骤如下：

1. 实例化 COSListDirCommand 对象；
2. 调用 COSClient 对象的 listDirRequest 方法，将 COSListDirCommand 对象传入；
3. 通过COSDirListTaskRsp 的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
num	NSUIntegeger	是	一次拉取数目设定
pageContext	NSString *	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中

prefix	NSString *	是	前缀查询

返回结果说明

通过TXYListDirCommandRsp的对象返回结果信息

属性名称	类型	说明
context	NSString *	目录个数
listover	NSString *	文件个数
infos	NSArray *	文件目录属性列表
retCode	int	任务描述代码
descMsg	NSString *	任务描述信息

示例

```
COSListDirCommand *cm = [COSListDirCommand new];
cm.directory = dir;
cm.bucket = bucket;
cm.sign = _sign;
cm.number = 100;
cm.pageContext = @"";
cm.prefix = @"xx";
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
      if (resp.retCode == 0) {
         //sucess
      }else{}
};
[client listDir:cm];
```

文件操作

初始化

方法原型

与目录操作相同，在进行文件操作之前，需引入上传 SDK 的头文件 *COSClient .h*，实例化 COSClient 对象。

参数说明

参数名称	类型	是否必填	说明
appId	NSString *	是	项目ID，即APP ID。
region	NSString *	是	bucket被创建的时候机房区域，例如上海：“sh” 广州："gz"

示例

```
- (instancetype)initWithAppId:(NSString*)appId  withRegion:(NSString *)region;
```

文件上传

方法原型

调用此接口者可进行本地文件上传操作，具体步骤如下：

1. 实例化 COSObjectPutTask ；
2. 调用 COSClient 对象的 putObject 方法，将 COSObjectPutTask 对象传入；
3. 通过COSObjectUploadTaskRsp的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
filePath	NSString *	是	文件路径
sign	NSString *	是	签名
bucket	NSString *	是	目标 Bucket 名称
fileName	NSString *	是	目标 文件上传cos后显示的 名称

attrs	NSString *	否	文件自定义属性
directory	NSString *	是	文件上传目录，相对路径 ,举例:@"path", 注意directory的首尾不要加上多余的/，SDK内部在生成请求URL时会加上/拼成完整的路径。

返回结果说明

通过COSObjectUploadTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息
sourceURL	NSString *	成功后，后台返回文件的 CDN url
sourceURL	NSString *	成功后，后台返回文件的 源站 url

示例

```
COSObjectPutTask *task = [COSObjectPutTask new];
task.filePath = path;
task.fileName = fileName;
task.bucket = bucket;
task.attrs = @"customAttribute";
task.directory = dir;
task.insertOnly = YES;
task.sign = _sign;
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
      if (resp.retCode == 0) {
       //sucess
      }else{}
};
```

```
client.progressHandler = ^(NSInteger bytesWritten,NSInteger
totalBytesWritten,NSInteger totalBytesExpectedToWrite){
        //progress
};
[client putObject:task];
```

文件属性更新

方法原型

调用此接口更新文件的自定义属性，具体步骤如下：

1. 实例化 COSObjectUpdateCommand 对象；
2. 调用 COSClient 的 updateObject 命令，传入 COSObjectUpdateCommand 对象；
3. 通过COSObjectUpdateTaskRsp的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
fileName	NSString *	是	-
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
attrs	NSString *	否	用户自定义属性

返回结果说明

通过TXYUpdateCommandRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息

示例

```
COSObjectUpdateCommand *cm = [COSObjectUpdateCommand new]
cm.fileName = file;
```

```
    cm.bucket = bucket;
    cm.sign = _oneSign;//单次签名
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        if (resp.retCode == 0) {
            //sucess
        }else{
    };
    [client updateObject:cm];
```

文件属性查询

方法原型

调用此接口可查询文件的属性信息，具体步骤如下：

1. 实例化 COSObjectMetaCommand 对象；
2. 调用 COSClient 的 getObjectInfo 命令，传入 COSObjectMetaCommand 对象；
3. 通过COSObjectMetaTaskRsp 类返回结果信息

参数说明

参数名称	类型	是否必填	说明
filename	NSString *	是	-
bucket	NSString *	是	文件所属 bucket 名称
directory	NSString *	是	目录路径（相对于bucket的路径）
sign	NSString *	是	签名

返回结果说明

通过TXYStatCommandRsp类返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息

data	NSDictionary *	成功时，文件基本信息

示例

```
COSObjectMetaCommand *cm = [COSObjectMetaCommand new] ;
cm.fileName = file;
cm.bucket = bucket;
cm.directory = dir;
cm.sign = _oneSign;//单次签名
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
      if (resp.retCode == 0) {
         //sucess
      }else{}
};
[client getObjectMetaData:cm];
```

文件删除

方法原型

调用此接口进行文件的删除操作，具体步骤如下：

1. 实例化 COSObjectDeleteCommand 对象；
2. 调用 COSClient 的 deleteObject 命令，传入 COSObjectDeleteCommand 对象。
3. 通过COSObjectDeleteTaskRsp的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
filename	NSString *	是	-
bucket	NSString *	是	文件所属 Bucket 名称
directory	NSString *	是	目录路径（相对于bucket的路径）
sign	NSString *	是	签名
objectType	TXYObjectType	是	业务类型，文件删除时设置为：TXYObjectFile

返回结果说明

通过COSObjectDeleteTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息

示例

```
COSObjectDeleteCommand *cm = [COSObjectDeleteCommand new];
cm.fileName = file;
cm.bucket = bucket;
cm.directory = dir;
cm.sign = _oneSign;//单次签名
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];;
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
      if (resp.retCode == 0) {
            //sucess
      }else{       }
};
[client deleteObject:cm];
```

文件下载

方法原型

调用此接口进行文件的下载操作，具体步骤如下：

1. 实例化 COSObjectGetTask 对象；
2. 调用 COSClient 的 getObjectRequest 命令，传入 COSObjectGetTask 对象。
3. 通过COSGetObjectTaskRsp 的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
filePath	NSString *	是	文件下载地址

返回结果说明

通过 通过COSGetObjectTaskRsp 的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息
object	NSMutableData *	下载文件

示例

```
    COSObjectGetTask *cm = [[COSObjectGetTask alloc] initWithUrl:imgUrl.text];
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
    client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
        //
    };
    client.downloadProgressHandler = ^(int64_t receiveLength,int64_t contentLength){
    };
    [client getObject:cm];
```

文件分片上传

方法原型

调用此接口进行文件的下载操作，具体步骤如下：

1. 实例化 COSObjectPutTask 对象；
2. 调用 COSClient 的 putObject 命令，传入 COSObjectPutTask 对象。
3. 通过COSObjectUploadTaskRsp 的对象返回结果信息
4. 当multipartUpload 参数设置为YES 的时候上传文件上传的方式为分片上传，该参数默认为NO；

参数说明

参数名称	类型	是否必填	说明

filePath	NSString *	是	文件路径
multipartUpload	BOOL	否	文件上传是否使用分片上传
sign	NSString *	是	签名
bucket	NSString *	是	目标 Bucket 名称
fileName	NSString *	是	目标 文件上传cos后显示的 名称
attrs	NSString *	否	文件自定义属性
directory	NSString *	是	文件上传目录，相对路径 ,举例:@"path", 注意directory的首尾不要加上多余的/，SDK内部在生成请求URL时会加上/拼成完整的路径。

返回结果说明

通过COSObjectUploadTaskRsp 的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息

示例

```
COSObjectPutTask *task = [[COSObjectPutTask alloc] init];
task.multipartUpload = YES;//分片上传设置参数
task.filePath = path;
```

```
    task.fileName = fileName;
    task.bucket = bucket;
    task.attrs = @"customAttribute";
    task.directory = dir;
    task.insertOnly = YES;
    task.sign = _sign;
    COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];  client.completionHandler = ^(COSTaskRsp *resp, NSDictionary
*context){

        if (rsp.retCode == 0) {
          //sucess
        }else{ }
    };
    client.progressHandler = ^(NSInteger bytesWritten,NSInteger
totalBytesWritten,NSInteger totalBytesExpectedToWrite){
      //进度
    };
    [client putObject:task];
```

文件断点续传

方法原型

调用此接口进行文件的上传操作，具体步骤如下：

1. 实例化 COSObjectPutTask ；
2. 调用 COSClient 对象的 putObject 方法，将 之前上传过的 COSObjectPutTask 对象传入；
3. 通过 COSObjectUploadTaskRsp 的对象返回结果信息

参数说明

参数名称	类型	是否必填	说明
filePath	NSString *	是	文件路径
sign	NSString *	是	签名
bucket	NSString	是	目标 Bucket 名称

	*		
fileName	NSString *	是	目标 文件上传cos后显示的 名称
attrs	NSString *	否	文件自定义属性
directory	NSString *	是	文件上传目录，相对路径，举例：@"path"，注意 directory 的首尾不要加上多余的/，SDK 内部在生成请求 URL 时会加上/拼成完整的路径。

返回结果说明

通过COSObjectUploadTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode == 0时标示成功，为负数表示为失败，20000以上的返回码为 SDK 内部错误
descMsg	NSString *	任务描述信息
sourceURL	NSString *	成功后，后台返回文件的 CDN url
sourceURL	NSString *	成功后，后台返回文件的 源站 url

示例

```
COSObjectPutTask *task = [COSObjectPutTask new];
task.filePath = path;
task.fileName = fileName;
task.bucket = bucket;
task.attrs = @"customAttribute";
task.directory = dir;
task.insertOnly = YES;
task.sign = _sign;
COSClient *client= [[COSClient alloc] initWithAppId:appId withRegion:[Congfig
instance].region];
client.completionHandler = ^(COSTaskRsp *resp, NSDictionary *context){
```

```
if (resp.retCode == 0) {
//sucess
}else{}
};
client.progressHandler = ^(NSInteger bytesWritten,NSInteger
totalBytesWritten,NSInteger totalBytesExpectedToWrite){
//progress
};
[client putObject:task];
```

Java SDK

最近更新时间：2020-02-12 13:55:29

> **注意**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

相关资源

cos java sdk v4 github 项目。

环境依赖

JDK 1.7（本版本 SDK 基于 JSON API 封装组成）。

安装 SDK

- maven 安装。
  pom.xml 添加依赖

```
<dependency>
        <groupId>com.qcloud</groupId>
        <artifactId>cos_api</artifactId>
        <version>4.7</version>
</dependency>
```

- 源码安装。
  从 cos java sdk v4 github 下载源码。

卸载SDK

删除 pom 依赖或源码。

历史版本

4.2版本是针对 COS 4.X系统，接口与3.x的基本一致，如果需要使用历史版本，请参见 cos java sdk v3 github 。

生成客户端对象

初始化密钥信息

```
long appId = 1000000;
String secretId = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx";
String secretKey = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx";
// 设置要操作的bucket
String bucketName = "xxxxxxxxx";
// 初始化密钥信息
Credentials cred = new Credentials(appId, secretId, secretKey);
```

初始化客户端配置（如设置园区）

```
// 初始化客户端配置
ClientConfig clientConfig = new ClientConfig();
// 设置bucket所在的区域，例如华南园区：gz； 华北园区：tj；华东园区：sh ；
clientConfig.setRegion("gz");
```

生成客户端

```
// 初始化cosClient
COSClient cosClient = new COSClient(clientConfig, cred);
```

文件操作

上传文件

方法原型

```
String uploadFile(UploadFileRequest request);
```

参数说明

参数名	类型	默认值	参数描述
request	UploadFileRequest	无	上传文件类型请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或	bucket 名称

			set 方法	
cosPath	String	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根 / 开始，文件路径不能以 / 结尾，例如 /mytest/demo.txt
localPath	String	无	构造函数或 set 方法	通过磁盘文件上传的本地绝对路径
contentBuffer	byte[]	无	构造函数或 set 方法	通过内存上传的 buffer 内容
bizAttr	String	空	构造函数或 set 方法	文件的备注，主要用于对该文件用途的描述
enableShaDigest	boolean	false	set 方法	是否计算 sha 摘要，如果开启 sha，并且 bucket 下有相同内容文件，则会触发秒传。sha 计算会耗费一定的 CPU 和时间，建议大文件不开启
taskNum	int	16	set 方法	文件上传的并发数

返回值

返回值类型	返回值描述
String	{'code':$code, 'message':$mess, 'data':$data}，code 为0表示成功，message 为 SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
UploadFileRequest uploadFileRequest = new UploadFileRequest(bucketName, "/sample_file.txt", "local_file_1.txt");
String uploadFileRet = cosClient.uploadFile(uploadFileRequest);
```

下载文件

方法原型

```
String getFileLocal(GetFileLocalRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	GetFileLocalRequest	无	下载文件请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根 / 开始，文件路径不能以 / 结尾，例如 /mytest/demo.txt
localPath	String	无	构造函数或 set 方法	要下载到的本地路径
useCDN	boolean	true	set 方法	是否通过 CDN 进行下载
referer	String	空串	set 方法	设置 Referer（针对开启了 refer 防盗链的 bucket）
rangeStart	long	0	set 方法	要下载的字节起始，参见 HTTP Range
rangeEnd	long	Long.MAX_VALUE	set 方法	下载的字节结束，参见 HTTP Range

示例

```
String localPathDown = "src/test/resources/local_file_down.txt";
GetFileLocalRequest getFileLocalRequest =
  new GetFileLocalRequest(bucketName, cosFilePath, localPathDown);
getFileLocalRequest.setUseCDN(false);
getFileLocalRequest.setReferer("*.myweb.cn");
String getFileResult = cosClient.getFileLocal(getFileLocalRequest);
```

移动文件

方法原型

```
String moveFile(MoveFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	MoveFileRequest	无	移动文件请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket名称
cosPath	String	无	构造函数或 set 方法	cos路径，必须从bucket下的根 / 开始，文件路径不能以 / 结尾，例如 /mytest/demo.txt
dstCosPath	String	无	构造函数或 set 方法	移动文件的目标地址，必须从bucket下的根 / 开始，文件路径不能以/结尾，例如 /mytest/demo.txt.move
overWrite	枚举类型	NO_OVER_WRITE (不覆盖)	set 方法 setOv	在移动的目标文件存在时，选择不覆盖还是覆盖，默认不覆盖

			erWrite	

示例

```
String cosFilePath = "/sample_file.txt";
String dstCosFilePath = "/sample_file.txt.bak";
MoveFileRequest moveRequest =
  new MoveFileRequest(bucketName, cosFilePath, dstCosFilePath);
String moveFileResult = cosClient.moveFile(moveRequest);
```

获取文件属性

方法原型

```
String statFile(StatFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFileRequest	无	获取文件属性请求

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根 `/` 开始，文件路径不能以 `/` 结尾，例如 `/mytest/demo.txt`

返回值

返回值类型	返回值描述
Strin	{'code':$code, 'message':$mess, 'data':$data}，code 为0表示成功，message 为

g	SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
StatFileRequest statFileRequest = new StatFileRequest(bucketName,
"/sample_file.txt");
String statFileRet = cosClient.statFile(statFileRequest);
```

更新文件属性

方法原型

```
String updateFile(UpdateFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFileRequest	无	更新文件属性请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径, 必须从 bucket 下的根 / 开始，文件路径不能以 / 结尾，例如 /mytest/demo.txt
bizAttr	String	无	set 方法	文件的备注，主要用于对改文件用途的描述
authority	String (枚举)	无	set 方法	文件权限，默认是继承 bucket 的权限合法取值: eInvalid（继承bucket），eWRPrivate（私有读写），eWPrivateRPublic（私有写, 公有读）

cacheControl	String	无	set 方法	参见 HTTP 的 Cache-Control
contentType	String	无	set 方法	参见 HTTP 的 Content-Type
contentLanguage	String	无	set 方法	参见 HTTP 的 Content-Language
contentDisposition	String	无	set 方法	参见 HTTP 的 Content-Disposition
x-cos-meta-	String	无	set 方法	自定义 HTTP 头，参数必须以 x-cos-meta-开头，值由用户定义，可设置多个

tips: 更新属性可以选择其中的某几个，对于HTTP头部cache_control，content_type，content_disposition 和 x-cos-meta-，如果本次只更新其中的某几个，其他的都会被抹掉，即这4个属性是整体更新。

返回值

返回值类型	返回值描述
String	{'code':$code，'message':$mess}，code为0表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
UpdateFileRequest updateFileRequest = new UpdateFileRequest(bucketName,
"/sample_file.txt");

updateFileRequest.setBizAttr("测试目录");
updateFileRequest.setAuthority(FileAuthority.WPRIVATE);
updateFileRequest.setCacheControl("no cache");
updateFileRequest.setContentDisposition("cos_sample.txt");
updateFileRequest.setContentLanguage("english");
updateFileRequest.setContentType("application/json");
updateFileRequest.setXCosMeta("x-cos-meta-xxx", "xxx");
updateFileRequest.setXCosMeta("x-cos-meta-yyy", "yyy");

String updateFileRet = cosClient.updateFile(updateFileRequest);
```

删除文件

方法原型

```
String delFile(DelFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFileRequest	无	删除文件请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径, 必须从 bucket 下的根 / 开始，文件路径不能以 / 结尾，例如 /mytest/demo.txt

返回值

返回值类型	返回值描述
String	{'code':$code, 'message':$mess}, code 为0表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
DelFileRequest delFileRequest = new DelFileRequest(bucketName,
"/sample_file_move.txt");
String delFileRet = cosClient.delFile(delFileRequest);
```

目录操作

创建目录

方法原型

```
String createFolder(CreateFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	CreateFolderRequest	无	创建目录请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos路径，必须从 bucket 下的根 / 开始，目录路径必须以 / 结尾，例如 /mytest/dir/
bizAttr	String	空	set方法	目录的备注，主要用于对目录用途的描述

返回值

返回值类型	返回值描述
String	{'code':$code, 'message':$mess}，code 为0表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
CreateFolderRequest createFolderRequest = new CreateFolderRequest(bucketName, "/sample_folder/");
String createFolderRet = cosClient.createFolder(createFolderRequest);
```

获取目录属性

方法原型

```
String statFolder(StatFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFolderRequest	无	获取目录属性请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根 / 开始，目录路径必须以 / 结尾，例如 /mytest/dir/

返回值

返回值类型	返回值描述
String	{'code':$code, 'message':$mess, 'data':$data}，code 为0表示成功，message 为 SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
StatFolderRequest statFolderRequest = new StatFolderRequest(bucketName,
"/sample_folder/");
String statFolderRet = cosClient.statFolder(statFolderRequest);
```

更新目录属性

方法原型

```
String updateFolder(UpdateFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFolderRequest	无	更新目录属性请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos路径，必须从 bucket 下的根 / 开始，目录路径必须以 / 结尾, 例如 /mytest/dir/
bizAttr	String	空	set 方法	目录的备注，主要用于对目录用途的描述

返回值

返回值类型	返回值描述
String	{'code':$code, 'message':$mess}，code 为0表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
UpdateFolderRequest updateFolderRequest = new UpdateFolderRequest(bucketName, "/sample_folder/");
updateFolderRequest.setBizAttr("这是一个测试目录");
String updateFolderRet = cosClient.updateFolder(updateFolderRequest);
```

获取目录列表

方法原型

```
String listFolder(ListFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	ListFolderRequest	无	获取目录成员请求

request 成员

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径, 必须从 bucket 下的根 `/` 开始，目录路径必须以 `/` 结尾, 例如 `/mytest/dir/`
num	int	199	构造函数或 set 方法	获取列表成员的数量，最大为199
prefix	String	空	构造函数或 set 方法	搜索成员的前缀, 例如 prefix 为 test 表示只搜索以 test 开头的文件或目录
context	String	空	构造函数或 set 方法	搜索上下文, 由上一次 list 的结果返回，作为这一次搜索的起点，用于循环获取一个目录下的所有成员

返回值

返回值类	返回值描述

型	
String	{'code':$code，'message':$mess, 'data':$data}，code 为0表示成功，message 为 SUCCESS 或者失败原因，data 中包含成员列表，详情请参见返回值模块

示例

```
ListFolderRequest listFolderRequest = new ListFolderRequest(bucketName,
"/sample_folder/");
String listFolderRet = cosClient.listFolder(listFolderRequest);
```

删除目录

方法原型

```
String delFolder(DelFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFolderRequest	无	删除目录请求

request 成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或 set 方法	bucket 名称
cosPath	String	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根 `/` 开始，目录路径必须以 `/` 结尾，例如 `/mytest/dir/`

返回值

返回值类型	返回值描述
String	{'code':$code，'message':$mess}，code为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
DelFolderRequest delFolderRequest = new DelFolderRequest(bucketName,
"/sample_folder/");
String delFolderRet = cosClient.delFolder(delFolderRequest);
```

签名管理

签名模块提供了生成多次签名、单次签名和下载签名的接口，其中多次签名和单次签名在文件和目录操作的api内部使用，用户不用关心，下载签名用于方便用户生成下载私有 bucket 的文件签名。

多次签名

```
String getPeriodEffectiveSign(String bucketName, String cosPath, Credentials cred,
long expired)
```

使用场景

上传文件，重命名文件，创建目录，获取文件目录属性，拉取目录列表。

参数说明

参数名	参数类型	默认值	参数描述
bucket	String	无	bucket 名称
cos_path	String	无	要签名的 cos 路径
cred	Credentials	无	用户身份信息，包括 appid，secretId，secretkey
expired	long	无	签名过期时间，UNIX时间戳

返回值

base64 编码的字符串。

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
long expired = System.currentTimeMillis() / 1000 + 600;
String signStr = Sign.getPeriodEffectiveSign(bucketName, "/pic/test.jpg", cred,
expired);
```

单次签名

```
String getOneEffectiveSign(String bucketName, String cosPath, Credentials cred)
```

使用场景

删除和更新文件目录。

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket 名称
cos_path	unicode	无	要签名的 cos 路径
cred	Credentials	无	用户身份信息，包括 appid，secretId，secretkey

返回值

base64 编码的字符串。

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
String signStr = Sign.getOneEffectiveSign(bucketName, "/pic/test.jpg", cred);
```

下载签名

```
String getDownLoadSign(String bucketName, String cosPath, Credentials cred, long expired)
```

使用场景

生成文件的下载签名，用于下载私有 bucket 的文件。

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket 名称
cos_pat	unicode	无	要签名的 cos 路径

h			
cred	Credentials	无	用户身份信息，包括 APPID，SecretId，Secretkey
expired	long	无	签名过期时间， UNIX 时间戳

返回值

base64 编码的字符串。

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
long expired = System.currentTimeMillis() / 1000 + 600;
String signStr = Sign.getDownLoadSign(bucketName, "/pic/test.jpg", cred, expired);
```

返回值

code	含义
0	操作成功
-1	输入参数错误，例如输入的本地文件路径不存在，cos 文件路径不符合规范
-2	网络错误，例如404等
-3	连接 cos 时发生异常，如连接超时

JavaScript SDK

最近更新时间：2023-01-16 17:49:00

> **注意**
> 目前查阅的是历史版本 SDK 文档，后续不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

SDK 获取

COS 服务的 JS SDK V4 版本的 GitHub 地址 。

开发环境

1. 使用 SDK 需要浏览器支持 HTML 5。
2. 请您登录 腾讯云控制台 获取您的项目 ID（APPID），bucket，secret_id 和 secret_key。
3. 请您登录 对象存储控制台 针对您要操作的 bucket 进行跨域（CORS）设置。

> **说明**
> 本版本 SDK 基于 JSON API 封装组成。

SDK 配置

直接下载 github 上提供的源代码，使用 SDK 之前，加载 dist 目录里的 cos-js-sdk-v4.js 文件即可。

```
<script type="text/javascript" src="cos-js-sdk-v4.js"></script>
```

初始化

```
    //初始化逻辑
    //特别注意: JS-SDK 使用之前请先到 console.cloud.tencent.com/cos 对相应的 Bucket 进行跨域设置
    var cos = new CosCloud({
        appid: appid,// APPID 必填参数
        bucket: bucket,//bucketName 必填参数
        region: 'sh',//地域信息 必填参数 华南地区填 gz 华东填 sh 华北填 tj
        getAppSign: function (callback) {//获取签名 必填参数
```

```
        //下面介绍获取签名的几种办法

        //1.搭建一个鉴权服务器，自己构造请求参数获取签名，推荐实际线上业务使用，优点是安全性好，不会暴露自己的私钥
        //拿到签名之后记得调用 callback
        /**
         $.ajax('SIGN_URL').done(function (data) {
            var sig = data.sign;
            callback(sig);
        });
         **/

        //2.直接在浏览器前端计算签名，需要获取自己的 accessKey 和 secretKey，一般在调试阶段使用
        //拿到签名之后记得调用 callback
        //var res = getAuth(); //这个函数自己根据签名算法实现
        //callback(res);


        //3.直接复用别人算好的签名字符串，一般在调试阶段使用
        //拿到签名之后记得调用 callback
        //callback('YOUR_SIGN_STR')
        //

    },
    getAppSignOnce: function (callback) {//单次签名，必填参数，参考上面的注释即可
        //填上获取单次签名的逻辑
    }
});
```

初始化参数说明

参数名	类型	是否必填	默认值	参数描述
appid	int	是	无	APPID
bucket	String	是	无	bucket 名称
region	String	是	'gz'	地域信息，必填参数，华南地区填 gz 华东填 sh 华北填tj
getAppSign	Function	是	无	获取多次签名的函数，建议从服务器端获取签名字符串

getAppSignOnce	Function	是	无	获取单次签名的函数，建议从服务器端获取签名字符串

返回结果说明

返回值：cos object 对象，初始化之后可以用这个 cos object 对象进行内置接口调用例如 uploadFile，deleteFile 等。

文件操作

普通文件上传

接口说明：通常用于较小文件（一般小于20MB）的上传，可以通过此接口上传较小的文件并获得文件的 url，如果文件大于20M则本接口内部会去调用分片上传接口。

方法原型

```
cos.uploadFile(successCallBack, errorCallBack, progressCallBack, bucket, path, file, insertOnly);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	上传成功的回调
errorCallBack	Function	是	无	上传失败的回调
progressCallBack	Function	是	无	上传过程进度的回调，例如文件1M已经上传了100K则会回调进度0.1
bucket	String	是	无	bucket 名称
path	String	是	无	文件在 COS 服务端的路径
file	File	是	无	本地要上传文件的文件对象（二进制数据）

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	提示信息
data	Object	是	返回数据
data.access_url	String	是	生成的文件 CDN 下载 url
data.source_url	String	是	生成的文件 COS 源站 url
data.url	String	是	操作文件的 url
data.resource_path	String	是	资源路径，格式：/appid/bucket/xxx

示例

```
var myFolder = '/111/';//需要操作的目录
var successCallBack = function (result) {
    $("#result").val(JSON.stringify(result));
};

var errorCallBack = function (result) {
    result = result || {};
    $("#result").val(result.responseText || 'error');
};

var progressCallBack = function(curr){
    $("#result").val('uploading... curr progress is '+curr);
};

$('#js-file').off('change').on('change', function (e) {
    var file = e.target.files[0];
    cos.uploadFile(successCallBack, errorCallBack, progressCallBack, bucket,
myFolder+file.name, file, 0);
    return false;
});
```

大文件分片上传

接口说明：通常用于较大文件（一般大于20MB）的上传，可以通过此接口上传较大的文件并获得文件的 url。

方法原型

```
cos.sliceUploadFile(successCallBack, errorCallBack, progressCallBack, bucket, path,
file, insertOnly);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	上传成功的回调
errorCallBack	Function	是	无	上传失败的回调
progressCallBack	Function	是	无	上传过程进度的回调，例如文件1M已经上传了100K则会回调进度0.1
bucket	String	是	无	bucket 名称
path	String	是	无	文件在 COS 服务端的路径
file	File	是	无	本地要上传文件的文件对象（二进制数据）

返回结果说明(json字符串)

需要注意的是大文件上传会经多个接口处理，以下是最后一片上传成功才会触发的回调

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	提示信息
data	Object	是	返回数据
data.access_url	String	是	生成的文件 CDN 下载 url
data.source_url	String	是	生成的文件 COS 源站 url
data.url	String	是	操作文件的url
data.resource_pat	String	是	资源路径. 格式:/appid/bucket/xxx

h

示例

```
var myFolder = '/111/';//需要操作的目录
var successCallBack = function (result) {
    $("#result").val(JSON.stringify(result));
};

var errorCallBack = function (result) {
    result = result || {};
    $("#result").val(result.responseText || 'error');
};

var progressCallBack = function(curr){
    $("#result").val('uploading... curr progress is '+curr);
};

$('#js-file').off('change').on('change', function (e) {
    var file = e.target.files[0];
    //大文件也可以直接调用 uploadFile
    cos.uploadFile(successCallBack, errorCallBack, progressCallBack, bucket,
myFolder+file.name, file, 0);
    //也可以用 sliceUploadFile，选一个即可
    //cos.sliceUploadFile(successCallBack, errorCallBack, progressCallBack, bucket,
myFolder+file.name, file, 0);
    return false;
});
```

删除文件

接口说明：删除文件

方法原型

```
cos.deleteFile(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述

successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	文件在 COS 服务端的路径

返回结果说明(json字符串)

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	提示信息

示例

```
//删除文件
$('#deleteFile').on('click', function () {
    var myFile = myFolder+'2.txt';//填您自己实际存在的文件
    cos.deleteFile(successCallBack, errorCallBack, bucket, myFile);
});
```

获取文件属性

接口说明：通过此接口查询文件的各项属性信息。

方法原型

```
cos.getFileStat(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	文件在 COS 服务端的路径

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Object	是	文件属性数据
data.name	String	是	文件或目录名
data.biz_attr	String	是	文件属性，业务端维护
data.ctime	String	是	文件的创建时间，unix 时间戳
data.mtime	String	是	文件的修改时间，unix 时间戳
data.filesize	Int	是	文件大小
data.filelen	Int	是	文件已传输大小
data.sha	String	是	文件 sha
data.access_url	String	是	生成的文件下载 url
data.authority	String	否	eInvalid，eWRPrivate，eWPrivateRPublic，文件可以与 bucket 拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为eInvalid，则会采用 bucket 的权限
data.custom_headers	String	否	自定义 header 对象

示例

```
//获取文件属性
$('#getFileStat').on('click', function () {
    var myFile = myFolder+'2.txt';//填您自己实际存在的文件
    cos.getFileStat(successCallBack, errorCallBack, bucket, myFile);
});
```

更新文件属性

接口说明：通过此接口更新文件的属性信息。

方法原型

```
cos.updateFile(successCallBack, errorCallBack, bucket, path, bizAttr);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	文件在 COS 服务端的路径
bizAttr	String	是	无	文件的自定义属性

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
//更新文件属性
$('#updateFile').on('click', function () {
    var myFile = myFolder+'2.txt';//填您自己实际存在的文件
    cos.updateFile(successCallBack, errorCallBack, bucket, myFile, 'my new file attr');
});
```

拷贝文件

接口说明：通过此接口把文件拷贝（即复制）到另一个路径。

方法原型

```
cos.copyFile(successCallBack, errorCallBack, bucket, path, destPath, overWrite);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要复制的文件在 COS 服务端的路径
destPath	String	是	无	复制的目标的路径
overWrite	Int	是	无	是否覆盖同名文件，0表示不覆盖，1表示覆盖

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
//拷贝文件，从源文件地址复制一份到新地址
$('#copyFile').on('click', function () {

    var myFile = '111/2.txt';//填您自己实际存在的文件
```

```
    //注意目标的路径，这里如果填333/2.txt 则表示文件复制到111/333/2.txt
    //如果填/333/2.txt 则表示文件复制到bucket根目录下的333/2.txt
    var newFile = '/333/2.txt';
    var overWrite = 1;//0 表示不覆盖 1表示覆盖
    cos.copyFile(successCallBack, errorCallBack, bucket, myFile, newFile, overWrite);
});
```

移动文件

接口说明：通过此接口把文件移动（剪切）到另一个路径，如果是移动到相同路径的话，可以达到修改文件名的效果。

方法原型

```
cos.moveFile(successCallBack, errorCallBack, bucket, path, destPath, overWrite);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要移动的文件在 COS 服务端的路径
destPath	String	是	无	移动的目标的路径
overWrite	Int	是	无	是否覆盖同名文件，0表示不覆盖，1表示覆盖

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
    //移动文件，把源文件移动到新地址，如果是同一个目录移动且文件名不同的话，相当于改了一个文件名
    //如果是移动到新目录，相当于剪切当前的文件，粘贴到了新目录
    $('#moveFile').on('click', function () {

        var myFile = '/111/2.txt';//填您自己实际存在的文件

        //注意目标的路径，这里如果填333/2.txt 则表示文件移动到111/333/2.txt
        //如果填/333/2.txt 则表示文件移动到 bucket 根目录下的333/2.txt
        //如果填/111/3.txt 则相当于把2.txt改名成3.txt
        var newFile = '/333/2.txt';
        var overWrite = 1;//0 表示不覆盖 1表示覆盖
        cos.moveFile(successCallBack, errorCallBack, bucket, myFile, newFile, overWrite);
    });
```

文件夹（目录）操作

新增文件夹

接口说明：通过此接口增加一个指定的文件夹。

方法原型

```
cos.createFolder(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要操作的文件夹在 COS 服务端的路径

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
$('#createFolder').on('click', function () {
    var newFolder = '/333/';//填您需要创建的文件夹
    cos.createFolder(successCallBack, errorCallBack, bucket, newFolder);
});
```

删除文件夹

接口说明：通过此接口删除指定的文件夹。

方法原型

```
cos.deleteFolder(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要操作的文件夹在 COS 服务端的路径

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
//删除文件夹
$('#deleteFolder').on('click', function () {
    var newFolder = '/333/';//填您需要删除的文件夹
    cos.deleteFolder(successCallBack, errorCallBack, bucket, newFolder);
});
```

获取文件夹属性

接口说明：通过此接口获取指定的文件夹属性。

方法原型

```
cos.getFolderStat(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要操作的文件夹在 COS 服务端的路径

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Object	是	文件夹属性信息对象
data.biz_attr	String	是	文件夹属性信息字符串

示例

```
//获取文件夹属性
$('#getFolderStat').on('click', function () {
    cos.getFolderStat(successCallBack, errorCallBack, bucket, '/333/');
});
```

更新文件夹属性

接口说明：通过此接口更新指定的文件夹属性。

方法原型

```
cos.updateFolder(successCallBack, errorCallBack, bucket, path, bizAttr);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要操作的文件夹在 COS 服务端的路径
bizAttr	String	是	无	新的属性信息

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```

```

```
//更新文件夹属性
$('#updateFolder').on('click', function () {
    cos.updateFolder(successCallBack, errorCallBack, bucket, '/333/', 'new attr');
});
```

获取文件夹内列表

接口说明：通过此接口获取指定的文件夹内的文件列表。

方法原型

```
cos.getFolderList(successCallBack, errorCallBack, bucket, path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
successCallBack	Function	是	无	操作成功的回调
errorCallBack	Function	是	无	操作失败的回调
bucket	String	是	无	bucket 名称
path	String	是	无	需要操作的文件夹在 COS 服务端的路径

返回结果说明（json字符串）

参数名	类型	是否必然返回	参数描述
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据

data.listover	Bool	是	是否有内容可以继续往前/往后翻页
data.context	String	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的 context 透传到参数中。order 用于指定翻页顺序。若 order 填0，则从当前页正序/往下翻页；若 order 填1，则从当前页倒序/往上翻页
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix 时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix 时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小（通过与 filesize 对比可知文件传输进度）
data.infos.sha	Stri	否(当类型为文	文件 sha

	ng	件时返回)	
data.infos.access_url	String	否(当类型为文件时返回)	生成的文件下载url
data.infos.authority	String	否	eInvalid，eWRPrivate，eWPrivateRPublic，文件可以与 bucket 拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为 eInvalid，则会采用 bucket 的权限

示例

```
//获取指定文件夹内的列表,默认每次返回20条
$('#getFolderList').on('click', function () {
    cos.getFolderList(successCallBack, errorCallBack, bucket, myFolder);
});
```

PHP SDK

最近更新时间：2024-04-25 18:04:01

> **注意**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

相关资源

COS PHP SDK V4 GitHub 项目（本版本 SDK 基于 JSON API 封装组成）。

开发环境

依赖环境：PHP 5.3.0 版本及以上。

SDK 配置

下载 SDK 后，在使用 SDK 时，加载 cos-php-sdk-v4/include.php 并设置全局的超时时间及 COS 所在的区域即可。

```
require('cos-php-sdk-v4/include.php');
use Qcloud\Cos\Api;

$config = array(
    'app_id' => '',
    'secret_id' => '',
    'secret_key' => '',
    'region' => 'gz',
    'timeout' => 60
);

$cosApi = new Api($config);
```

生成签名

多次有效签名

方法原型

```
public function createReusableSignature($expiration, $bucket, $filepath);
```

参数说明

参数名	参数描述	类型	必填
expiration	过期时间，Unix 时间戳	long	是
bucket	Bucket 名称	String	是
filepath	文件路径	String	否

示例

```
$auth = new Auth($appId = '',$secretId ='',$secretKey = '');
$expiration = time() + 60;
$bucket = 'testbucket';
$filepath = "/myFloder/myFile.rar";
$sign = $auth->createReusableSignature($expiration, $bucket, $filepath);
```

单次有效签名

方法原型

```
public function createNonreusableSignature($bucket, $filepath);
```

参数说明

参数名	参数描述	类型	必填
bucket	Bucket 名称	String	是
filepath	文件路径，以斜杠开头。 例如 `/filepath/filename` 为文件在此 bucketname 下的全路径	String	是

示例

```
$auth = new Auth($appId = '',$secretId ='',$secretKey = '');
$bucket = 'testbucket';
$filepath = "/myFloder/myFile.rar";
$sign = $auth->createNonreusableSignature($bucket, $filepath);
```

目录操作

创建目录

接口说明：用于目录的创建，可通过此接口在指定 Bucket 下创建目录。

方法原型

```
public function createFolder($bucketName, $path, $bizAttr = null);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	目录全路径	String	是
bizAttr	目录属性信息，业务自行维护	String	否

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是
data	返回数据，请参考《Restful API 创建目录》	Array	否

示例

```
$bizAttr = "attr_folder";
$result  = $cosApi->createFolder($bucketName, $path,$bizAttr)
```

目录更新

接口说明：用于目录业务自定义属性的更新，调用者可以通过此接口更新业务的自定义属性字段。

方法原型

```
public function updateFolder($bucketName, $path, $bizAttr = null);
```

参数说明

参数名	参数描述	类型	必填

bucketName	Bucket 名称	String	是
path	目录路径	String	是
bizAttr	目录属性信息	String	否

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是

示例

```
$bizAttr = "folder new attribute";
$result  = $cosApi->updateFolder($bucketName, $path, $bizAttr)
```

目录查询

接口说明：用于目录属性的查询，调用者可以通过此接口查询目录的属性。

原型方法

```
public function statFolder($bucketName, $path);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	目录路径	String	是

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是
data	目录属性数据，请参考《Restful API 目录查询》	Array	否

示例

```
$result = $cosApi->statFolder($bucketName, $path);
```

删除目录

接口说明：用于目录的删除，调用者可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
public function delFolder($bucketName, $path);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	目录全路径	String	是

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是

示例

```
$result = $cosApi->delFolder($bucketName, $path);
```

列举目录中的文件和目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
public function listFolder($bucketName, $path, $num = 20, $context = null);
```

注意： v4.3.7及之后的sdk版本不再包含pattern和order这两个参数。

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket名称	String	是
path	目录的全路径	String	是
num	要查询的目录/文件数量	int	否
context	透传字段，查看第一页，则传空字符串。 若需要翻页，需要将前一页返回值中的 context 透传到参数中	String	否

返回值说明(json)

参数名	参数描述	类型	必带
code	API 错误码，成功时为 0	Int	是
message	错误信息	String	是
data	返回数据，请参考 《Restful API 目录列表》	Array	是

示例

```
$result = $cosApi->listFolder($bucketName, $path, 20, 'eListBoth',0);
```

列举目录下指定前缀文件/目录

接口说明：用于列举目录下指定前缀的文件和目录，可以通过此接口查询目录下的指定前缀的文件和目录信息。

原型方法

```
public function prefixSearch($bucketName, $prefix, $num = 20,
$context = null);
```

注意： v4.3.7及之后的sdk版本不再包含pattern和order这两个参数。

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称，Bucket 创建参见 创建Bucket	String	是
prefix	列出含此前缀的所有文件(带全路径)	String	是
num	要查询的目录/文件数量	int	否
context	透传字段，查看第一页，则传空字符串。 若需要翻页，需要将前一页返回值中的 context 透传到参数中	String	否

返回值说明（JSON）

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	API 错误信息	String	是
data	返回数据，请参考 《Restful API 目录列表》	Array	是

示例

```
$prefix= "/myFolder/2015-";
$result = $cosApi->prefixSearch($bucketName, $prefix, 20, 'eListBoth',0);
```

文件操作

文件上传

接口说明：文件上传的统一接口，对于大于 20M 的文件，内部会通过多次分片的方式进行文件上传。

原型方法

```
public function upload($bucketName, $srcPath, $dstPath,
        $bizAttr = null, $slicesize = null);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称，Bucket 创建参见 创建 Bucket	String	是
srcPath	本地要上传文件的全路径	String	是
dstPath	文件在 COS 服务端的全路径，不包括 /appid/bucketname	String	是
bizAttr	文件属性，业务端维护	String	否
slicesize	文件分片大小，当文件大于 20M 时，SDK 内部会通过多次分片的方式进行上传。 默认分片大小为 1M，支持的最大分片大小为 3M	int	否

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是
data	返回数据，请参考 《Restful API 创建文件》	Array	是

示例

```
$dstPath = "/myFolder/test.mp4";
$bizAttr = "";
$sliceSize = 3 * 1024 * 1024;
$result = $cosApi->upload($bucketName, $srcPath, $dstPath ,"biz_attr");
```

文件下载

接口说明：文件下载的统一接口。

原型方法

```
public function download($bucket, $srcPath, $dstPath);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket名称，bucket 创建参见 创建Bucket
srcPath	String	是	文件在 COS 服务端的全路径，不包括/appid/bucketname
dstPath	String	是	本地要保存文件的全路径

返回值说明(json)

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 创建文件》

示例

```
$result = $cosApi->download($bucketName,“a.txt”, "E:/a.txt");
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

原型方法

```
public function update($bucketName, $path,
            $bizAttr = null, $authority=null,$customer_headers_array=null);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	文件在文件服务端的全路径，不包括 /appid/bucketname	String	是
bizAttr	待更新的文件属性信息	String	否

authority	eInvalid(继承Bucket的读写权限)；eWRPrivate(私有读写)；eWPrivateRPublic(公有读私有写)	String	否
customer_headers_array	用户自定义头域。可携带参数名分别为：Cache-Control 、Content-Type 、Content-Disposition 、Content-Language 、以及以 x-cos-meta- 为前缀的参数名称	String	否

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是

示例

```
$bizAttr = "";
$authority = "eWPrivateRPublic";
$customer_headers_array = array(
    'Cache-Control' => "no",
    'Content-Language' => "ch",
);
$result = $cosApi->update($bucketName, $dstPath, $bizAttr,$authority,
$customer_headers_array);
```

文件查询

接口说明：用于文件的查询，调用者可以通过此接口查询文件的各项属性信息。

原型方法

```
public function stat($bucketName, $path);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	文件在文件服务端的全路径	String	是

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是
data	文件属性数据，请参考《Restful API 文件查询》	Array	是

示例

```
$result = $cosApi->stat($bucketName, $path);
```

文件删除

接口说明：用于文件的删除，调用者可以通过此接口删除已经上传的文件。

原型方法

```
public function delFile($bucketName, $path);
```

参数说明

参数名	参数描述	类型	必填
bucketName	Bucket 名称	String	是
path	文件的全路径	String	是

返回值说明(json)

参数名	参数描述	类型	必带
code	错误码，成功时为 0	Int	是
message	错误信息	String	是

示例

```
$result = $cosApi->delFile($bucketName, $path);
```

Python SDK

最近更新时间：2023-07-11 11:56:32

> ⚠ **注意：**
> 您目前查阅的是历史版本 SDK 文档，已不再更新和维护，我们建议您查阅新版 SDK 文档 。

开发准备

相关资源

- Python-SDK 项目 GitHub 地址，欢迎贡献代码以及反馈问题。
- PyPi 项目 GitHub 地址。
  （本版本 SDK 基于 JSON API 封装组成）

环境依赖

Python 2.7
获取 Python 版本的方法：

- Linux Shell

```
$ python -V
Python 2.7.11
```

- Windows cmd

```
D:\>python -V
Python 2.7.11
```

如果提示不是内部或外部命令，请先在 Windows 环境变量 PATH 里添加上 Python 的绝对路径。

安装 SDK

- pip 安装

```
pip install qcloud_cos_v4
```

- 源码安装

github 上下载 SDK，解压后如下执行（如果提示 permission deny，需要有管理员权限）。

```
python setup.py install
```

卸载 SDK

```
pip uninstall qcloud_cos_v4
```

生成客户端对象

初始化客户端

```
appid = 100000                  # 替换为用户的 appid
secret_id = u'xxxxxxxx'         # 替换为用户的 secret_id
secret_key = u'xxxxxxx'         # 替换为用户的 secret_key
region_info = "sh"              # 替换为用户的 region，例如 sh 表示华东园区, gz 表示华南园区,
tj 表示华北园区
cos_client = CosClient(appid, secret_id, secret_key, region=region_info)
```

自定义接入点

如果需要使用自定义的接入域名，可以通过下面的方式设置：

```
conf = CosConfig(hostname='', download_hostname='')
cos_client.set_config(conf)
```

文件操作

上传文件

方法原型

```
def upload_file(self, request)
```

参数说明

参数名	类型	默认值	参数描述
request	UploadFileRequest	无	上传文件类型请求

request 成员	类型	默认值	设置方法	描述

bucket_name	unicode	无	构造函数或set 方法	bucket 名称
cos_path	unicode	无	构造函数或set 方法	cos 路径，必须从 bucket 下的根/开始，文件路径不能以/结尾，例如 /mytest/demo.txt
local_path	unicode	无	构造函数或set 方法	要上传的本地文件的绝对路径
biz_attr	unicode	空	构造函数或set 方法	文件的备注，主要用于对该文件用途的描述

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess, 'data':$data}， code 为0表示成功，message 为SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
request = UploadFileRequest(bucket, u'/sample_file.txt', u'local_file_1.txt')
upload_file_ret = cos_client.upload_file(request)
```

获取文件属性

方法原型

```
def stat_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFileRequest	无	获取文件属性请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set 方法	bucket 名称

cos_path	unicode	无	构造函数或set 方法	cos 路径，必须从 bucket 下的根/开始，文件路径不能以/结尾，例如 /mytest/demo.txt

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess, 'data':$data}，code 为0表示成功，message 为SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
request = StatFileRequest(bucket, u'/sample_file.txt')
stat_file_ret = cos_client.stat_file(request)
```

更新文件属性

方法原型

```
def update_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFileRequest	无	更新文件属性请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set 方法	bucket 名称
cos_path	unicode	无	构造函数或set 方法	cos 路径，必须从 bucket 下的根/开始，文件路径不能以/结尾，例如 /mytest/demo.txt

biz_attr	unicode	无	set 方法	文件的备注，主要用于对改文件用途的描述
authority	unicode (枚举)	无	set 方法	文件权限，默认是继承 bucket 的权限合法取值：eInvalid（继承bucket），eWRPrivate（私有读写），eWPrivateRPublic（私有写，公有读）
cache_control	unicode	无	set 方法	参见 HTTP 的 Cache-Control
content_type	unicode	无	set 方法	参见 HTTP 的 Content-Type
content_language	unicode	无	set 方法	参见 HTTP 的 Content-Language
content_disposition	unicode	无	set 方法	参见 HTTP 的 Content-Disposition
x-cos-meta-	unicode	无	set 方法	自定义 HTTP 头，参数必须以 x-cos-meta-开头，值由用户定义，可设置多个

tips: 用户可以在以上这些属性中选择几个进行更新。如果本次只更新 HTTP 头部 cache_control，content_type，content_disposition 和 x-cos-meta- 这四个中的某几个，其他的几个没有更新和设置，那么其他没有被设置的头部会被清除删除掉，不会出现，即这4个属性会出现整体一起更新变动。

返回值

返回值类型	返回值描述
dict	{'code':$code，'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = UpdateFileRequest(bucket, u'/sample_file.txt')
request.set_biz_attr(u'这是个demo文件')          # 设置文件 biz_attr 属性
request.set_authority(u'eWRPrivate')            # 设置文件的权限
request.set_cache_control(u'cache_xxx')         # 设置 Cache-Control
request.set_content_type(u'application/text')   # 设置 Content-Type
request.set_content_disposition(u'ccccxxx.txt')  # 设置 Content-Disposition
```

```
request.set_content_language(u'english')        # 设置 Content-Language
request.set_x_cos_meta(u'x-cos-meta-xxx', u'xxx') # 设置自定义的 x-cos-meta- 属性
request.set_x_cos_meta(u'x-cos-meta-yyy', u'yyy') # 设置自定义的 x-cos-meta- 属性
update_file_ret = cos_client.update_file(request)
```

下载文件

方法原型

```
def download_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	DownloadFileRequest	无	下载文件请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数	bucket 名称
cos_path	unicode	无	构造函数	cos 路径，必须从 bucket 下的根/开始，文件路径不能以/结尾，例如 /mytest/demo.txt
local_filename	unicode	无	构造函数	本地文件路径

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = DownloadFileRequest(bucket, u'/sample_file_move.txt', u'/tmp/a.txt')
```

```
download_ret = cos_client.download_file(request)
```

删除文件

方法原型

```
def del_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFileRequest	无	删除文件请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，文件路径不能以/结尾，例如 /mytest/demo.txt

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = DelFileRequest(bucket, u'/sample_file_move.txt')
del_ret = cos_client.del_file(request)
```

目录操作

创建目录

方法原型

```
def create_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	CreateFolderRequest	无	创建目录请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/
biz_attr	unicode	空	set 方法	目录的备注，主要用于对目录用途的描述

返回值

返回值类型	返回值描述
dict	{'code':$code，'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = CreateFolderRequest(bucket, u'/sample_folder/')
create_folder_ret = cos_client.create_folder(request)
```

获取目录属性

方法原型

```
def stat_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFolderRequest	无	获取目录属性请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess, 'data':$data}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，data 中包含相关的属性，详情请参见返回值模块

示例

```
request = StatFolderRequest(bucket, u'/sample_folder/')
stat_folder_ret = cos_client.stat_folder(request)
```

更新目录属性

方法原型

```
def update_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFolderRequest	无	更新目录属性请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/
biz_attr	unicode	空	set 方法	目录的备注，主要用于对目录用途的描述

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = UpdateFolderRequest(bucket, u'/sample_folder/')
request.set_biz_attr(u'这是一个测试目录')
update_folder_ret = cos_client.update_folder(request)
```

获取目录列表

方法原型

```
def list_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述

request	ListFolderRequest	无	获取目录成员请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/
num	int	199	构造函数或 set 方法	获取列表成员的数量，最大为 199
prefix	unicode	空	构造函数或 set 方法	搜索成员的前缀，例如 prefix 为 test 表示只搜索以 test 开头的文件或目录
context	unicode	空	构造函数或 set 方法	透传字段，从响应的返回内容中得到。若查看第一页，则将空字符串作为 context 传入。若需要翻页，需要将前一页返回内容中的 context 透传到参数中。

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess, 'data':$data}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，data 中 包含成员列表，详情请参见返回值模块

示例

```
request = ListFolderRequest(bucket, u'/sample_folder/')
list_folder_ret = cos_client.list_folder(request)
```

删除目录

方法原型

```
def del_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFolderRequest	无	删除目录请求

request 成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或 set 方法	bucket 名称
cos_path	unicode	无	构造函数或 set 方法	cos 路径，必须从 bucket 下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/

返回值

返回值类型	返回值描述
dict	{'code':$code, 'message':$mess}，code 为 0 表示成功，message 为 SUCCESS 或者失败原因，详情请参见返回值模块

示例

```
request = DelFolderRequest(bucket, u'/sample_folder/')
delete_folder_ret = cos_client.del_folder(request)
```

签名管理

签名模块提供了生成多次签名、单次签名和下载签名的接口，其中多次签名和单次签名在文件和目录操作的api内部使用，用户不用关心，下载签名用于方便用户生成下载私有 bucket 的文件签名。

多次签名

```
def sign_more(self, bucket, cos_path, expired)
```

使用场景

上传文件, 重命名文件, 创建目录, 获取文件目录属性, 拉取目录列表

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket 名称
cos_path	unicode	无	要签名的 cos 路径
expired	int	无	签名过期时间，UNIX 时间戳

返回值

base64 编码的字符串

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_more(u'mybucket', u'/pic/1.jpg',  int(time.time()) + 600)
```

单次签名

```
def sign_once(self, bucket, cos_path)
```

使用场景

删除和更新文件目录

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket 名称
cos_path	unicode	无	要签名的 cos 路径

返回值

base64 编码的字符串。

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_once(u'mybucket', u'/pic/1.jpg')
```

下载签名

```
def sign_download(self, bucket, cos_path, expired)
```

使用场景

生成文件的下载签名, 用于下载私有 bucket 的文件。

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket 名称
cos_path	unicode	无	要签名的 cos 路径
expired	int	无	签名过期时间，UNIX 时间戳

返回值

base64 编码的字符串。

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_download(u'mybucket', u'/pic/1.jpg',  int(time.time()) + 600)
```

返回码

code	含义
0	操作成功
-1	输入参数错误，例如输入的本地文件路径不存在，COS 文件路径不符合规范
-2	网络错误，如 404 等

-3	连接 COS 时发生异常，如连接超时
-71	操作频率过快，触发 COS 的攻击