阿里云
对象存储

API 参考

文档版本：20190815

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 简介

阿里云对象存储服务（Object Storage Service，简称OSS），是阿里云对外提供的海量、安全、低成本、高可靠的云存储服务。您可以通过本文档提供的简单的REST接口，在任何时间、任何地点、任何互联网设备上进行上传和下载数据。基于OSS，您可以搭建出各种多媒体分享网站、网盘、个人和企业数据备份等基于大规模数据的服务。

使用限制

您使用的OSS资源和相关功能，都有一定的限制，具体请参见OSS使用限制。

使用说明

OSS API参考主要介绍接口的请求语法、相关参数含义以及请求和返回示例。如果要进行快速二次开发，建议您使用SDK开发包。关于SDK的安装和使用，请参见OSS SDK参考。

OSS定价

关于OSS的价格，请参见OSS详细价格信息。关于OSS的计量计费方式，请参见OSS计量项和计费项。

资源术语

中文	英文	说明
存储空间	Bucket	存储空间是您用于存储对象（Object）的容器，所有的对象都必须隶属于某个存储空间。
对象/文件	Object	对象是 OSS 存储数据的基本单元，也被称为OSS的文件。对象由元信息（Object Meta）、用户数据（Data）和文件名（Key）组成。对象由存储空间内部唯一的Key来标识。
地域	Region	地域表示 OSS 的数据中心所在物理位置。您可以根据费用、请求来源等综合选择数据存储的地域。详情请查看OSS已经开通的Region。
访问域名	Endpoint	Endpoint 表示OSS对外服务的访问域名。OSS以HTTP RESTful API的形式对外提供服务，当访问不同地域的时候，需要不同的域名。通过内网和外网访问同一个地域所需要的域名也是不同的。具体的内容请参见各个Region对应的Endpoint。

中文	英文	说明
访问密钥	AccessKey	AccessKey，简称 AK，指的是访问身份验证中用到的AccessKeyId 和AccessKeySecret。OSS通过使用AccessKeyId 和AccessKeySecret对称加密的方法来验证某个请求的发送者身份。AccessKeyId用于标识用户，AccessKeySecret是用户用于加密签名字符串和OSS用来验证签名字符串的密钥，其中AccessKeySecret 必须保密。

2 API概览

本文介绍对象存储OSS提供的相关API接口。

关于Service操作

API	描述
GetService (ListBuckets)	返回请求者拥有的所有Bucket

关于Bucket的操作

API	描述
PutBucket	创建Bucket
PutBucketACL	设置Bucket访问权限
PutBucketLogging	开启Bucket日志
PutBucketWebsite	设置Bucket为静态网站托管模式
PutBucketReferer	设置Bucket的防盗链规则
PutBucketLifecycle	设置Bucket中Object的生命周期规则
GetBucket(ListObject)	列出Bucket中所有Object的信息
GetBucketAcl	获得Bucket访问权限
GetBucketLocation	获得Bucket所属的数据中心位置信息
GetBucketInfo	获取Bucket信息
GetBucketLogging	查看Bucket的访问日志配置情况
GetBucketWebsite	查看Bucket的静态网站托管状态
GetBucketReferer	查看Bucket的防盗链规则
GetBucketLifecycle	查看Bucket中Object的生命周期规则
DeleteBucket	删除Bucket
DeleteBucketLogging	关闭Bucket访问日志记录功能
DeleteBucketWebsite	关闭Bucket的静态网站托管模式
DeleteBucketLifecycle	删除Bucket中Object的生命周期规则
PutBucketEncryption	配置Bucket的加密规则
GetBucketEncryption	获取Bucket的加密规则
DeleteBucketEncryption	删除Bucket的加密规则

关于Object的操作

API	描述
PutObject	上传Object
CopyObject	拷贝一个Object成另外一个Object
GetObject	获取Object
AppendObject	在Object尾追加上传数据
DeleteObject	删除Object
DeleteMultipleObjects	删除多个Object
HeadObject	只返回某个Object的meta信息，不返回文件内容
GetObjectMeta	返回Object的基本meta信息，包括该Object的ETag、Size（文件大小）、LastModified，不返回文件内容
PostObject	使用Post上传Object
PutObjectACL	设置ObjectACL
GetObjectACL	获取ObjectACL信息
Callback	上传回调
PutSymlink	创建软链接
GetSymlink	获取软链接
RestoreObject	解冻文件
SelectObject	用SQL语法查询Object内容
PutObjectTagging	设置或更新对象标签
GetObjectTagging	获取对象标签信息
DeleteObjectTagging	删除指定的对象标签

关于Multipart Upload的操作

API	描述
InitiateMultipartUpload	初始化MultipartUpload事件
UploadPart	分块上传文件
UploadPartCopy	分块复制上传文件
CompleteMultipartUpload	完成整个文件的MultipartUpload上传
AbortMultipartUpload	取消MultipartUpload事件

API	描述
ListMultipartUploads	罗列出所有执行中的MultipartUpload事件
ListParts	罗列出指定UploadID所属的所有已经上传成功Part

跨域资源共享(CORS)

API	描述
PutBucketcors	在指定Bucket设定一个CORS的规则
GetBucketcors	获取指定的Bucket目前的CORS规则
DeleteBucketcors	关闭指定Bucket对应的CORS功能并清空所有规则
OptionObject	跨域访问preflight请求

关于Live Channel的操作

API	描述
PutLiveChannelStatus	切换LiveChannel的状态
PutLiveChannel	创建LiveChannel
GetVodPlaylist	获取播放列表
PostVodPlaylist	生成播放列表
Get LiveChannelStat	获取LiveChannel的推流状态信息
GetLiveChannelInfo	获取LiveChannel的配置信息
GetLiveChannelHistory	获取LiveChannel的推流记录
ListLiveChannel	列举LiveChannel
DeleteLiveChannel	删除LiveChannel

3 公共HTTP头定义

本文介绍了对象存储OSS的公共请求头和公共响应头的详细说明。

公共请求头（Common Request Headers）

OSS的RESTful接口中使用了一些公共请求头。这些请求头可以被所有的OSS请求所使用，其详细定义如下：

名称	类型	描述
Authorization	字符串	用于验证请求合法性的认证信息。 默认值：无 使用场景：非匿名请求
Content-Length	字符串	RFC2616中定义的HTTP请求内容长度。 默认值：无 使用场景：需要向OSS提交数据的请求
Content-Type	字符串	RFC2616中定义的HTTP请求内容类型。 默认值：无 使用场景：需要向OSS提交数据的请求
Date	字符串	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT 默认值：无
Host	字符串	访问Host值，格式为：`<bucketname>.oss-cn-hangzhou.aliyuncs.com`。 默认值：无

公共响应头（Common Response Headers）

OSS的RESTful接口中使用了一些公共响应头。这些响应头可以被所有的OSS请求所使用，其详细定义如下：

名称	类型	描述
Content-Length	字符串	RFC2616中定义的HTTP请求内容长度。 默认值：无 使用场景：需要向OSS提交数据的请求
Connection	枚举	标明客户端和OSS服务器之间的链接状态。 有效值：open、close 默认值：无
Date	字符串	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT 默认值：无
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。 默认值：无
Server	字符串	生成Response的服务器。 默认值：AliyunOSS

名称	类型	描述
x-oss-request-id	字符串	x-oss-request-id是由Aliyun OSS创建，并唯一标识这个response的UUID。如果在使用OSS服务时遇到问题，可以凭借该字段联系OSS工作人员，快速定位问题。 默认值：无

4 访问控制

4.1 用户签名验证

对象存储OSS使用AccessKeyId及AccessKeySecret对称加密的方法来验证某个请求的发送者身份。

说明:

- AccessKeyId用于标示用户。
- AccessKeySecret是用户用于加密签名字符串和OSS用来验证签名字符串的密钥，且AccessKeySecret必须保密。

AccessKey 根据所属账号的类型分为以下三种：

- 阿里云账户AccessKey：阿里云账号提供的AccessKey拥有所属资源的全部操作权限
- RAM账户AccessKey：RAM账户由阿里云账号授权生成，所拥有的AccessKey拥有对特定资源限定的操作权限
- STS临时访问凭证：由阿里云账号或RAM账号生成，所拥有的AccessKey在限定时间内拥有对特定资源限定的操作权限。超出限定时间后STS临时访问权限无效。

详情请参考OSS产品文档中访问身份验证。

当用户想以个人身份向OSS发送请求时，首先需要将发送的请求按照OSS指定的格式生成签名字符串；然后使用AccessKeySecret对签名字符串进行加密产生验证码。 OSS收到请求以后，会通过AccessKeyId找到对应的AccessKeySecret，以同样的方法提取签名字符串和验证码，如果计算出来的验证码和提供的一样即认为该请求是有效的；否则，OSS将拒绝处理这次请求，并返回HTTP 403错误。

4.2 在Header中包含签名

您可以在HTTP请求中增加 `Authorization` 的Header来包含签名（Signature）信息，表明这个消息已被授权。

SDK 签名实现

OSS SDK已经实现签名，用户使用OSS SDK不需要关注签名问题。如果您想了解具体语言的签名实现，请参考OSS SDK的代码。OSS SDK签名实现的文件如下表：

SDK	签名实现
Java SDK	OSSRequestSigner.java
Python SDK	auth.py
.Net SDK	OssRequestSigner.cs
PHP SDK	OssClient.php
C SDK	oss_auth.c
JavaScript SDK	client.js
Go SDK	auth.go
Ruby SDK	util.rb
iOS SDK	OSSModel.m
Android SDK	OSSUtils.java

当您自己实现签名，访问OSS报 `SignatureDoesNotMatch` 错误时，请参见自签名计算失败排除错误。

Authorization字段计算的方法

```
Authorization = "OSS " + AccessKeyId + ":" + Signature
Signature = base64(hmac-sha1(AccessKeySecret,
            VERB + "\n"
            + Content-MD5 + "\n"
            + Content-Type + "\n"
            + Date + "\n"
            + CanonicalizedOSSHeaders
            + CanonicalizedResource))
```

- `AccessKeySecret` 表示签名所需的密钥。
- `VERB`表示HTTP 请求的Method，主要有PUT、GET、POST、HEAD、DELETE等。
- `\n` 表示换行符。
- `Content-MD5` 表示请求内容数据的MD5值，对消息内容（不包括头部）计算MD5值获得128比特位数字，对该数字进行base64编码而得到。该请求头可用于消息合法性的检查（消息内容是否与发送时一致），如”eB5eJF1ptWaXm4bijSPyxw==”，也可以为空。详情请参见RFC2616 Content-MD5。
- `Content-Type` 表示请求内容的类型，如”application/octet-stream”，也可以为空。
- `Date` 表示此次操作的时间，且必须为GMT格式，如”Sun, 22 Nov 2015 08:16:38 GMT”。
- `CanonicalizedOSSHeaders` 表示以 x-oss- 为前缀的HTTP Header的字典序排列。
- `CanonicalizedResource` 表示用户想要访问的OSS资源。

其中，Date和CanonicalizedResource不能为空；如果请求中的Date时间和OSS服务器的时间差15分钟以上，OSS服务器将拒绝该服务，并返回HTTP 403错误。

构建CanonicalizedOSSHeaders的方法

所有以 x-oss- 为前缀的HTTP Header被称为CanonicalizedOSSHeaders。它的构建方法如下：

1. 将所有以 x-oss- 为前缀的HTTP请求头的名字转换成小写 。如`X-OSS-Meta-Name: TaoBao`转换成`x-oss-meta-name: TaoBao`。
2. 如果请求是以STS获得的AccessKeyId和AccessKeySecret发送时，还需要将获得的security-token值以 `x-oss-security-token:security-token` 的形式加入到签名字符串中。
3. 将上一步得到的所有HTTP请求头按照名字的字典序进行升序排列。
4. 删除请求头和内容之间分隔符两端出现的任何空格。如`x-oss-meta-name: TaoBao`转换成：`x-oss-meta-name:TaoBao`。
5. 将每一个头和内容用 `\n` 分隔符分隔拼成最后的CanonicalizedOSSHeaders。

说明：

- CanonicalizedOSSHeaders可以为空，无需添加最后的 `\n`。
- 如果只有一个，则如 `x-oss-meta-a\n`，注意最后的`\n`。
- 如果有多个，则如 `x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n`，注意最后的`\n`。

构建CanonicalizedResource的方法

用户发送请求中想访问的OSS目标资源被称为CanonicalizedResource。它的构建方法如下：

1. 将CanonicalizedResource置成空字符串 `""`。
2. 放入要访问的OSS资源 `/BucketName/ObjectName`（如果没有ObjectName则CanonicalizedResource为”/BucketName/“，如果同时也没有BucketName则为“/”）。
3. 如果请求的资源包括子资源(SubResource)，那么将所有的子资源按照字典序，从小到大排列并以 & 为分隔符生成子资源字符串。在CanonicalizedResource字符串尾添加 ? 和子资源字符串。此时的CanonicalizedResource如：`/BucketName/ObjectName?acl&uploadId=UploadId`。

说明：

- OSS目前支持的子资源(sub-resource)包括：acl，uploads，location，cors，logging，website，referer，lifecycle，delete，append，tagging，objectMeta，uploadId，partNumber，security-token，position，img，style，styleName，replication，replicationProgress，replicationLocation，cname，bucketInfo，comp，qos，live，status，vod，startTime，endTime，symlink，x-oss-process，response-content-type，response-content-language，response-expires，response-cache-control，response-content-disposition，response-content-encoding等。
- 子资源(sub-resource)有三种类型：
  - 资源标识，如子资源中的acl、append、uploadId、symlink等，详见关于Bucket的操作和关于Object的操作。
  - 指定返回Header字段，如 `response-***`，详见GetObject的`Request Parameters`。
  - 文件（Object）处理方式，如 `x-oss-process`，用于文件的处理方式，如图片处理。

计算签名头规则

- 签名的字符串必须为 `UTF-8` 格式。含有中文字符的签名字符串必须先进行 `UTF-8` 编码，再与`AccessKeySecret`计算最终签名。
- 签名的方法用RFC 2104中定义的HMAC-SHA1方法，其中Key为 AccessKeySecret。
- `Content-Type` 和 `Content-MD5` 在请求中不是必须的，但是如果请求需要签名验证，空值的话以换行符 `\n` 代替。
- 在所有非HTTP标准定义的header中，只有以 `x-oss-` 开头的header，需要加入签名字符串；其他非HTTP标准header将被OSS忽略（如下方签名示例中的x-oss-magic是需要加入签名字符串的）。
- 以 `x-oss-` 开头的header在签名验证前需要符合以下规范：
  - header的名字需要变成小写。
  - header按字典序自小到大排序。
  - 分割header name和value的冒号前后不能有空格。
  - 每个Header之后都有一个换行符“\n”，如果没有Header，CanonicalizedOSSHeaders就设置为空。

签名示例

请求	签名字符串计算公式	签名字符串
PUT /nelson HTTP/1.0 Content-MD5: eB5eJF1ptWaXm4bijSPyxw== Content-Type: text/html Date: Thu, 17 Nov 2005 18:49:58 GMT Host: oss-example.oss-cn-hangzhou.aliyuncs.com X-OSS-Meta-Author: foo@bar.com X-OSS-Magic: abracadabra	Signature = base64(hmac-sha1(AccessKeySecret,VERB + “\n” + Content-MD5 + “\n” + Content-Type + “\n” + Date + “\n” + CanonicalizedOSSHeaders + CanonicalizedResource))	“PUT\n eB5eJF1ptWaXm4bijSPyxw==\n text/html\n Thu, 17 Nov 2005 18:49:58 GMT\n x-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nels

假如AccessKeyId是“44CF959******252F707”，AccessKeySecret是“OtxrzxIsfpFjA7Sw******8Bw21TLhquhboDYROV”，可用以下方法计算签名(Signature)：

python示例代码：

```
import base64
import hmac
import sha
h = hmac.new("OtxrzxIsfpFjA7Sw******8Bw21TLhquhboDYROV",
             "PUT\nODBGOERFMDMzQTczRUY3NUE3NzA5QzdFNUYzMDQxNEM=\ntext
/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-magic:abracadabra\nx-oss-
meta-author:foo@bar.com\n/oss-example/nelson", sha)
Signature = base64.b64encode(h.digest())
print("Signature: %s" % Signature)
```

签名(Signature)计算结果应该为 26NBxoKd******Dv6inkoDft/yA=，因为Authorization = “OSS”+ AccessKeyId + “:” + Signature，所以最后Authorization为 “OSS 44CF95900***BF252F707:26NBxoKd******Dv6inkoDft/yA=”，然后加上Authorization头来组成最后需要发送的消息：

```
PUT /nelson HTTP/1.0
Authorization:OSS 44CF95900***BF252F707:26NBxoKd******Dv6inkoDft/yA=
Content-Md5: eB5eJF1ptWaXm4bijSPyxw==
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
X-OSS-Meta-Author: foo@bar.com
X-OSS-Magic: abracadabra
```

细节分析如下：

- 如果传入的AccessKeyId不存在或inactive，返回403 Forbidden。错误码：InvalidAccessKeyId。

- 若用户请求头中Authorization值的格式不对，返回400 Bad Request。错误码：InvalidArgument。
- OSS所有的请求都必须使用HTTP 1.1协议规定的GMT时间格式。其中，日期的格式为：

```
date1 = 2DIGIT SP month SP 4DIGIT; day month year (e.g., 02 Jun
        1982)
```

上述日期格式中，“天”所占位数都是“2 DIGIT”。因此，“Jun 2”、“2 Jun 1982”和“2-Jun-82”都是非法日期格式。

- 如果签名验证的时候，头中没有传入Date或者格式不正确，返回403 Forbidden错误。错误码：AccessDenied。
- 传入请求的时间必须在OSS服务器当前时间之后的15分钟以内，否则返回403 Forbidden。错误码：RequestTimeTooSkewed。
- 如果AccessKeyId是active的，但OSS判断用户的请求发生签名错误，则返回403 Forbidden，并在返回给用户的response中告诉用户正确的用于验证加密的签名字符串。用户可以根据OSS的response来检查自己的签名字符串是否正确。

返回示例：

```
<?xml version="1.0" ?>
<Error>
 <Code>
     SignatureDoesNotMatch
 </Code>
 <Message>
     The request signature we calculated does not match the
signature you provided. Check your key and signing method.
 </Message>
 <StringToSignBytes>
     47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31
31 20 30 37 3a 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72 65 61 6c 74
 65 73 74 3f 61 63 6c
 </StringToSignBytes>
 <RequestId>
     1E446260FF9B****
 </RequestId>
 <HostId>
     oss-cn-hangzhou.aliyuncs.***
 </HostId>
 <SignatureProvided>
     y5H7yzPsA/tP4+0tH1HHvPEwUv8=
 </SignatureProvided>
 <StringToSign>
     GET
Wed, 11 May 2011 07:59:25 GMT
/oss-example?acl
 </StringToSign>
 <OSSAccessKeyId>
     AKIAIVAKMSMOY7VO****
 </OSSAccessKeyId>
```

```
</Error>
```

常见问题

Content-MD5的计算方法

```
Content-MD5的计算
以消息内容为"123456789"来说，计算这个字符串的Content-MD5
正确的计算方式：
标准中定义的算法简单点说就是：
1．先计算MD5加密的二进制数组（128位）。
2．再对这个二进制进行base64编码（而不是对32位字符串编码）。
以Python为例子：
正确计算的代码为：
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
需要注意
正确的是：hash.digest()，计算出进制数组（128位）
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
常见错误是直接对计算出的32位字符串编码进行base64编码。
例如，错误的是：hash.hexdigest()，计算得到可见的32位字符串编码
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
错误的MD5值进行base64编码后的结果：
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

4.3 在URL中包含签名

除了使用Authorization Header，您还可以在URL中加入签名信息，以便将该URL转给第三方实现授权访问。

示例代码

URL中添加签名的Python示例代码：

```
import base64
import hmac
import sha
import urllib
h = hmac.new("OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
             "GET\n\n\n1141889120\n/oss-example/oss-api.pdf",
             sha)
urllib.quote (base64.encodestring(h.digest()).strip())
```

OSS SDK中提供了URL签名方法，详细请参考SDK文档。

OSS SDK的URL签名实现，请参看下表：

SDK	URL签名方法	实现文件
Java SDK	OSSClient.generatePresignedUrl	OSSClient.java
Python SDK	Bucket.sign_url	api.py
.Net SDK	OssClient.GeneratePresignedUri	OssClient.cs
PHP SDK	OssClient.signUrl	OssClient.php
JavaScript SDK	signatureUrl	Object.js
C SDK	oss_gen_signed_url	oss_object.c

实现方式

URL签名示例：

```
http://oss-example.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=nz2pc56s936**9l&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

URL签名必须至少包含Signature、Expires和OSSAccessKeyId三个参数。

- Expires 参数的值是一个Unix time（自UTC时间1970年1月1号开始的秒数），用于标识该URL的超时时间。如果OSS接收到这个URL请求的时候晚于签名中包含的Expires参数时，则返回请求超时的错误码。例如：当前时间是1141889060，开发者希望创建一个60秒后自动失效的URL，则可以设置Expires时间为1141889120。

  说明：

  OSS控制台中默认URL的有效时间为3600秒，最大为64800秒。

- OSSAccessKeyId 即密钥中的AccessKeyId。
- Signature 表示签名信息。所有的OSS支持的请求和各种Header参数，在URL中进行签名的算法和在Header中包含签名的算法基本一样。

```
Signature = urlencode(base64(hmac-sha1(AccessKeySecret,
          VERB + "\n"
          + CONTENT-MD5 + "\n"
          + CONTENT-TYPE + "\n"
          + EXPIRES + "\n"
          + CanonicalizedOSSHeaders
```

```
            + CanonicalizedResource)))
```

与Header中包含签名相比主要区别如下：

- 通过URL包含签名时，之前的Date参数换成Expires参数。
- 不支持同时在URL和Header中包含签名。
- 如果多次传入Signature、Expires或OSSAccessKeyId，以第一次为准。
- 先验证请求时间是否晚于Expires时间，然后再验证签名。
- 将签名字符串放到URL时，注意要对URL进行urlencode。

· 临时用户URL签名时，需要携带`security-token`，格式如下：

```
http://oss-example.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?
OSSAccessKeyId=nz2pc56s936**9l&Expires=1141889120&Signature=
vjbyPxybdZaNmGa%2ByT272YEAiv4%3D&security-token=SecurityToken
```

细节分析

· 使用在URL中签名的方式，会将你授权的数据在过期时间内曝露在互联网上，请预先评估使用风险。
· PUT和GET请求都支持在URL中签名。
· 在URL中添加签名时，Signature、Expires和OSSAccessKeyId顺序可以交换，但是如果缺少Signature、Expires或OSSAccessKeyId其中的一个或者多个，返回403 Forbidden消息。错误码：AccessDenied。
· 如果访问的当前时间晚于请求中设定的Expires时间或时间格式错误，返回403 Forbidden消息。错误码：AccessDenied。
· 如果URL中包含Signature，Expires，OSSAccessKeyId中的一个或者多个，并且Header中也包含签名消息，返回400 Bad Request消息 。错误码：InvalidArgument。
· 生成签名字符串时，除Date被替换成Expires参数外，仍然包含content-type、content-md5等上节中定义的Header（请求中虽然仍有Date这个请求Header，但不需要将Date加入签名字符串中）。

4.4 Bucket权限控制

对象存储OSS提供Bucket级别的权限访问控制。

Bucket目前有三种访问权限：public-read-write，public-read和private，它们的含义如下：

权限值	中文名称	权限对访问者的限制
public-read-write	公共读写	任何人（包括匿名访问者）都可以对该存储空间内文件进行读写操作。 警告: 互联网上任何用户都可以对该 Bucket 内的文件进行访问，并且向该 Bucket 写入数据。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益。除特殊场景外，不建议您配置公共读写权限。
public-read	公共读，私有写	只有该存储空间的拥有者可以对该存储空间内的文件进行写操作，任何人（包括匿名访问者）都可以对该存储空间中的文件进行读操作。 警告: 互联网上任何用户都可以对该 Bucket 内文件进行访问，这有可能造成您数据的外泄以及费用激增，请谨慎操作。
private	私有读写	只有该存储空间的拥有者可以对该存储空间内的文件进行读写操作，其他人无法访问该存储空间内的文件。

说明:

- 用户创建一个Bucket时，如果不指定Bucket权限，OSS会自动为该Bucket设置private权限。
- 对于private资源，需要获得授权才能访问。关于授权管理，可参考权限控制 。
- 对于一个已经存在的Bucket，只有它的创建者可以通过OSS的 Put Bucket Acl接口修改该Bucket的权限。

5 关于Service操作

5.1 GetService (ListBuckets)

对于服务地址做Get请求可以返回请求者拥有的所有Bucket，其中“/”表示根目录。

请求语法

```
GET / HTTP/1.1
Host: oss.example.com
Date: GMT Date
Authorization: SignatureValue
```

请求参数

GetService(ListBucket)时，可以通过prefix、marker和max-keys对list做限定，返回部分结果。

表 5-1: 请求参数

名称	类型	是否必需	描述
prefix	字符串	否	限定返回的bucket name必须以prefix作为前缀，可以不设定，不设定时不过滤前缀信息。 默认值：无
marker	字符串	否	设定结果从marker之后按字母排序的第一个开始返回，可以不设定，不设定时从头开始返回数据。。 默认值：无
max-keys	字符串	否	限定此次返回bucket的最大数，如果不设定，默认为100，max-keys取值不能大于1000 默认值：100

响应元素

名称	类型	描述
ListAllMyBucketsResult	容器	保存Get Service请求结果的容器。 子节点：Owner、Buckets 父节点：None

名称	类型	描述
Prefix	字符串	本次查询结果的前缀，当bucket未全部返回时才有此节点。 父节点：ListAllMyBucketsResult
Marker	字符串	标明这次GetService(ListBucket)的起点，当bucket未全部返回时才有此节点。 父节点：ListAllMyBucketsResult
MaxKeys	字符串	响应请求内返回结果的最大数目，当bucket未全部返回时才有此节点。 父节点：ListAllMyBucketsResult
IsTruncated	枚举字符串	指明是否所有的结果都已经返回：“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。当bucket未全部返回时才有此节点。 有效值：true、false 父节点：ListAllMyBucketsResult
NextMarker	字符串	表示下一次GetService(ListBucket)可以以此为marker，将未返回的结果返回。当bucket未全部返回时才有此节点。 父节点：ListAllMyBucketsResult
Owner	容器	用于存放Bucket拥有者信息的容器。 父节点：ListAllMyBucketsResult
ID	字符串	Bucket拥有者的用户ID。 父节点：ListAllMyBucketsResult.Owner
DisplayName	字符串	Bucket拥有者的名称 (目前和ID一致)。 父节点：ListAllMyBucketsResult.Owner
Buckets	容器	保存多个Bucket信息的容器。 子节点：Bucket 父节点：ListAllMyBucketsResult

名称	类型	描述
Bucket	容器	保存bucket信息的容器。 子节点：Name, CreationDate, Location 父节点：ListAllMyBucketsResult.Buckets
Name	字符串	Bucket名称。 父节点：ListAllMyBucketsResult.Buckets.Bucket
CreateDate	时间 (格式：yyyy-mm-ddThh:mm:ss.timezone, e.g., 2011-12-01T12:27:13.000Z)	Bucket创建时间 父节点：ListAllMyBucketsResult.Buckets.Bucket
Location	字符串	Bucket所在的数据中心。 父节点：ListAllMyBucketsResult.Buckets.Bucket
ExtranetEndpoint	字符串	Bucket访问的外网域名。 父节点：ListAllMyBucketsResult.Buckets.Bucket
IntranetEndpoint	字符串	同区域ECS访问Bucket的内网域名。 父节点：ListAllMyBucketsResult.Buckets.Bucket
StorageClass	字符串	Bucket存储类型，支持“Standard”、“IA”、“Archive”（目前只有部分区域支持“Archive”类型）。 父节点：ListAllMyBucketsResult.Buckets.Bucket

细节分析

- GetService这个API只对验证通过的用户有效。
- 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden。错误码：AccessDenied。

- 当所有的bucket都返回时，返回的xml中不包含Prefix、Marker、MaxKeys、IsTruncated、NextMarker节点，如果还有部分结果未返回，则增加上述节点，其中NextMarker用于继续查询时给marker赋值。

示例

- 请求示例1

```
GET / HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
Authorization: OSS nxj7dtlhcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv2******
```

返回示例1

```
HTTP/1.1 200 OK
Date: Thu, 15 May 2014 11:18:32 GMT
Content-Type: application/xml
Content-Length: 556
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C2300****
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Owner>
    <ID>512**</ID>
    <DisplayName>51264</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <CreationDate>2015-12-17T18:12:43.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-shanghai.aliyuncs.com</ExtranetEn
dpoint>
      <IntranetEndpoint>oss-cn-shanghai-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-shanghai</Location>
      <Name>app-base-oss</Name>
      <StorageClass>Standard</StorageClass>
    </Bucket>
    <Bucket>
      <CreationDate>2014-12-25T11:21:04.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEn
dpoint>
      <IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-hangzhou</Location>
      <Name>atestleo23</Name>
      <StorageClass>IA</StorageClass>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

- 请求示例2

```
GET /?prefix=xz02tphky6fjfiuc&max-keys=1 HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
```

```
Authorization: OSS nxj7dtwhcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv2****
```

返回示例2

```
HTTP/1.1 200 OK
Date: Thu, 15 May 2014 11:18:32 GMT
Content-Type: application/xml
Content-Length: 545
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C2300****
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Prefix>xz02tphky6fjfiuc</Prefix>
  <Marker></Marker>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>true</IsTruncated>
  <NextMarker>xz02tphky6fjfiuc0</NextMarker>
  <Owner>
    <ID>ut_test_put_bucket</ID>
    <DisplayName>ut_test_put_bucket</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <CreationDate>2014-05-15T11:18:32.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEn
dpoint>
      <IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-hangzhou</Location>
      <Name>xz02tphky6fjfiuc0</Name>
      <StorageClass>Standard</StorageClass>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- iOS
- Node.js
- Ruby

6 关于Bucket的操作

6.1 PutBucket

PutBucket接口用于创建存储空间（Bucket）。

> **说明:**
>
> - 此接口不支持匿名访问。
> - 一个用户在同一地域（Region）内最多可创建30个Bucket。
> - 每个地域都有对应的访问域名（Endpoint），地域与访问域名的对应关系参见访问域名和数据中心。

请求语法

```
PUT / HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
x-oss-acl: Permission
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration>
    <StorageClass>Standard</StorageClass>
</CreateBucketConfiguration>
```

请求头

名称	类型	是否必选	描述
x-oss-acl	字符串	否	指定Bucket访问权限。 有效值：public-read-write、public-read、private **说明:** 如果创建的 Bucket 没有指定访问权限，则默认使用private权限。

请求元素

名称	类型	是否必选	描述
StorageClass	字符串		指定Bucket存储类型。 有效值：Standard、IA、Archive
DataRedundancyType	字符串	否	指定Bucket的数据容灾类型。 有效值： · LRS（本地容灾类型，默认值） · ZRS（同城容灾类型）

示例

请求示例

```
PUT / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2017 03:15:40 GMT
x-oss-acl: private
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:77Dvh5wQgIjWjwO/KyRt8dOP
****
<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration>
    <StorageClass>Standard</StorageClass>
</CreateBucketConfiguration>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2017 03:15:40 GMT
Location: /oss-example
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C

- .NET
- Android
- iOS
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
InvalidBucketName	400	创建的Bucket不符合命名规范。
AccessDenied	403	· 发起PutBucket请求时没有传入用户验证信息。 · 没有操作权限。
TooManyBuckets	400	创建的Bucket数量超过上限。 同一用户在同一Region内最多可创建30个Bucket。

6.2 DeleteBucket

DeleteBucket用于删除某个存储空间（Bucket）。

> **说明:**
>
> · 只有Bucket的拥有者才有权限删除该Bucket。
>
> · 为了防止误删除的发生，OSS不允许删除一个非空的Bucket。

请求语法

```
DELETE / HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

· 正常删除的请求示例

```
DELETE / HTTP/1.1
Host: test.oss-cn-hangzhou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
date: Tue, 15 Jan 2019 08:19:04 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
```

```
Content-Length: 0
```

返回示例

```
HTTP/1.1 204 No Content
Server: AliyunOSS
Date: Tue, 15 Jan 2019 08:19:04 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5C3D9778CC1C2AEDF85B****
x-oss-server-time: 190
```

- **删除的Bucket不存在的请求示例**

```
DELETE / HTTP/1.1
Host: test.oss-cn-hangzhou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
date: Tue, 15 Jan 2019 07:53:24 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
Content-Length: 0
```

返回示例

```
HTTP/1.1 404 Not Found
Server: AliyunOSS
Date: Tue, 15 Jan 2019 07:53:25 GMT
Content-Type: application/xml
Content-Length: 288
Connection: keep-alive
x-oss-request-id: 5C3D9175B6FC201293AD****

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchBucket</Code>
  <Message>The specified bucket does not exist.</Message>
  <RequestId>5C3D9175B6FC201293AD****</RequestId>
  <HostId>test.oss-cn-hangzhou.aliyuncs.com</HostId>
  <BucketName>test</BucketName>
</Error>
```

- **删除的Bucket非空的请求示例**

```
DELETE / HTTP/1.1
Host: test.oss-cn-hangzhou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
date: Tue, 15 Jan 2019 07:35:06 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
Content-Length: 0
```

返回示例

```
HTTP/1.1 409 Conflict
```

```
Server: AliyunOSS
Date: Tue, 15 Jan 2019 07:35:06 GMT
Content-Type: application/xml
Content-Length: 296
Connection: keep-alive
x-oss-request-id: 5C3D8D2A0ACA54D87B43****
x-oss-server-time: 16

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>BucketNotEmpty</Code>
  <Message>The bucket you tried to delete is not empty.</Message>
  <RequestId>5C3D8D2A0ACA54D87B43****</RequestId>
  <HostId>test.oss-cn-hangzhou.aliyuncs.com</HostId>
  <BucketName>test</BucketName>
</Error>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- iOS
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403 Forbidden	没有删除该Bucket的权限。只有Bucket的拥有者才能删除该Bucket。

6.3 PutBucketACL

PutBucketACL接口用于修改存储空间（Bucket）的访问权限。只有该Bucket的创建者有权限执行此操作。

说明:

Bucket拥有者发起PutBucketACL请求时，如果Bucket已存在但权限不一致，将更新权限。如果Bucket不存在，将创建Bucket。

请求语法

```
PUT /?acl HTTP/1.1
x-oss-acl: Permission
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

请求头

名称	类型	是否必选	描述
x-oss-acl	字符串	是	指定Bucket的访问权限。 PutBucketACL接口通过Put请求中的x-oss-acl请求头来设置权限，如果没有该请求头，权限设置不生效。 取值：public-read-write、public-read、private

示例

请求示例

```
PUT /?acl HTTP/1.1
x-oss-acl: public-read
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrT
****
```

返回示例

- 正常返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

- 设置权限无效的返回示例

```
HTTP/1.1 400 Bad Request
x-oss-request-id: 56594298207FB3044385****
Date: Fri, 24 Feb 2012 03:55:00 GMT
Content-Length: 309
Content-Type: text/xml; charset=UTF-8
```

```
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>InvalidArgument</Code>
  <Message>no such bucket access control exists</Message>
  <RequestId>5***9</RequestId>
  <HostId>***-test.example.com</HostId>
  <ArgumentName>x-oss-acl</ArgumentName>
  <ArgumentValue>error-acl</ArgumentValue>
</Error>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403	· 发起PutBucketACL请求时，没有传入用户验证信息 · 没有发起PutBucketACL请求的权限。只有该Bucket的创建者有权限执行此操作。

6.4 GetBucketAcl

GetBucketAcl接口用于获取某个存储空间（Bucket）的访问权限（ACL）。只有Bucket的拥有者才能获取Bucket的访问权限。

请求语法

```
GET /?acl HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
```

```
Authorization: SignatureValue
```

响应元素

名称	类型	描述
AccessControlList	容器	存储ACL信息的容器类 父节点：AccessControlPolicy
AccessControlPolicy	容器	保存GetBucketACL结果的容器 父节点：None
DisplayName	字符串	Bucket拥有者的名称（目前和用户ID一致） 父节点：AccessControlPolicy.Owner
Grant	枚举字符串	Bucket的ACL权限 有效值：private、public-read、public-read-write 父节点：AccessControlPolicy.AccessControlList
ID	字符串	Bucket拥有者的用户ID 父节点：AccessControlPolicy.Owner
Owner	容器	保存Bucket拥有者信息的容器 父节点：AccessControlPolicy

示例

请求示例

```
GET /?acl HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 04:11:23 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0Fmgb
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
```

```
Date: Fri, 24 Feb 2012 04:11:23 GMT
Content-Length: 253
Content-Type: application/xml
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" ?>
<AccessControlPolicy>
    <Owner>
        <ID>0022012****</ID>
        <DisplayName>user_example</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>public-read</Grant>
    </AccessControlList>
</AccessControlPolicy>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有操作权限。只有Bucket的拥有者才能获取Bucket的访问权限。

6.5 PutBucketLifecycle

PutBucketLifecycle接口用于设置存储空间（Bucket）的生命周期规则。生命周期规则开启后，OSS将按照配置规则，定期自动删除与规则相匹配的文件（Object）。只有Bucket的拥有者才能发起此请求。

 说明：

- 如果Bucket此前没有设置过生命周期规则，此操作会创建一个新的生命周期规则；如果Bucket此前设置过生命周期规则，此操作会覆写先前的配置。
- PutBucketLifecycle是覆盖语义。如需追加Lifecycle rule，请先调用GetBucketLifecycle获取当前Lifecycle配置，然后追加新的Lifecycle配置，并调用PutBucketLifecycle接口更新Bucket Lifecycle配置。
- PutBucketLifecycle操作可以对Object以及Part（以分片方式上传，但最后未提交的分片）设置过期时间。

请求语法

```
PUT /?lifecycle HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Authorization: SignatureValue
Host: BucketName.oss.aliyuncs.com
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
  <Rule>
    <ID>RuleID</ID>
    <Prefix>Prefix</Prefix>
    <Status>Status</Status>
    <Expiration>
      <Days>Days</Days>
    </Expiration>
    <Transition>
      <Days>Days</Days>
      <StorageClass>StorageClass</StorageClass>
    </Transition>
    <AbortMultipartUpload>
      <Days>Days</Days>
    </AbortMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

请求元素

名称	类型	是否必选	描述
CreatedBeforeDate	字符串	Days和CreatedBeforeDate二选一	指定一个日期，OSS会对该日期之前的数据执行生命周期规则。日期必需服从ISO8601的格式，并总是UTC的零点。 例如：2002-10-11T00:00:00.000Z，表示将最后更新时间早于2002-10-11T00:00:00.000Z 的Object删除或转换成其他存储类型，等于或晚于这个时间的Object不会被删除或转储。 父节点：Expiration或者AbortMultipartUpload

名称	类型	是否必选	描述
Days	正整数	Days和CreatedBeforeDate二选一	指定生命周期规则在Object最后更新过后多少天生效。 父节点：Expiration
Expiration	容器	否	指定Object生命周期规则的过期属性。 子节点：Days或CreatedBeforeDate 父节点：Rule
AbortMultipartUpload	容器	否	指定未完成分片上传的过期属性。 子节点：Days或CreatedBeforeDate 父节点：Rule
ID	字符串	否	规则唯一的ID。最多由255个字节组成。如没有指定，或者该值为空时，OSS会自动生成一个唯一ID。 子节点：无 父节点：Rule
LifecycleConfiguration	容器	是	Lifecycle配置的容器，最多可容纳1000条规则。 子节点：Rule 父节点：无
Prefix	字符串	是	指定规则所适用的前缀。只有匹配前缀的Object才可能被该规则所影响。前缀不可重叠。 子节点：无 父节点：Rule

名称	类型	是否必选	描述
Rule	容器	是	表述一条规则。  说明: · 不支持Archvie Bucket创建转储规则。 · Object设置过期时间必须大于转储为IA或者Archive存储类型的时间。 子节点：ID，Prefix，Status，Expiration 父节点：LifecycleConfiguration
Status	字符串	是	如果值为Enabled，那么OSS会定期执行该规则；如果为Disabled，那么OSS会忽略该规则。 父节点：Rule 有效值：Enabled，Disabled
StorageClass	字符串	如果父节点Transition已设置，则为必选	指定Object转储的存储类型。  说明: IA Bucket中Object可以转储为Archive存储类型，但不支持转储为Standard存储类型。 取值：IA，Archive 父节点：Transition
Transition	容器	否	指定Object在有效生命周期中，OSS何时将Object转储为IA或者Archive存储类型 。  说明: Standard Bucket中的Standard Object可以转储为IA、Archive存储类型，但转储Archive存储类型的时间必须比转储IA存储类型的时间长。例如Transition IA设置Days为30，Transition Archive设置Days必须大于30。

名称	类型	是否必选	描述
Tag	容器	否	指定规则所适用的对象标签，可设置多个。 父节点：Rule 子节点：Key， Value
Key	字符串	若父节点Tag已设置，则为必需	Tag Key 父节点：Tag
Value	字符串	若父节点Tag已设置，则为必需	Tag Value 父节点：Tag

示例

- **不受版本控制的生命周期配置请求示例**

```
PUT /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 443
Date: Thu , 8 Jun 2017 13:08:38 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:PYbzsdWAIWAlMW8luk*****
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
  <Rule>
    <ID>delete objects and parts after one day</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>1</Days>
    </Expiration>
    <AbortMultipartUpload>
      <Days>1</Days>
    </AbortMultipartUpload>
  </Rule>
  <Rule>
    <ID>transit objects to IA after 30, to Archive 60, expire after
10 years</ID>
    <Prefix>data/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>IA</StorageClass>
    </Transition>
    <Transition>
      <Days>60</Days>
      <StorageClass>Archive</StorageClass>
    </Transition>
    <Expiration>
      <Days>3600</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>transit objects to Archive after 60 days</ID>
    <Prefix>important/</Prefix>
```

```
    <Status>Enabled</Status>
    <Transition>
      <Days>6</Days>
      <StorageClass>Archive</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>delete created before date</ID>
    <Prefix>backup/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <CreatedBeforeDate>2017-01-01T00:00:00.000Z</CreatedBeforeDate
>
    </Expiration>
    <AbortMultipartUpload>
      <CreatedBeforeDate>2017-01-01T00:00:00.000Z</CreatedBeforeDate
>
    </AbortMultipartUpload>
  </Rule>
  <Rule>
    <ID>r1</ID>
    <Prefix>rule1</Prefix>
    <Tag><Key>xx</Key><Value>1</Value></Tag>
    <Tag><Key>yy</Key><Value>2</Value></Tag>
    <Status>Enabled</Status>
    <Expiration>
      <Days>30</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>r2</ID>
    <Prefix>rule2</Prefix>
    <Tag><Key>xx</Key><Value>1</Value></Tag>
    <Status>Enabled</Status>
    <Transition>
      <Days>60</Days>
    <StorageClass>Archive</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674A4D890*****
Date: Thu , 8 Jun 2017 13:08:38 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go

- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403	没有操作权限。只有Bucket的拥有者才能发起PutBucketLifecycle请求。
InvalidArgument	400	· OSS支持Standard Bucket中Standard Objects转储为IA、Archive存储类型。Standard Bucket可以针对一个Object同时配置转储IA和Archive存储类型规则。转储Archive存储类型的时间必须比转储IA存储类型的时间长。 · Object的指定过期时间必须比转储为IA或者Archive存储类型的时间长。

6.6 GetBucketLifecycle

GetBucketLifecycle接口用于查看存储空间（Bucket）的生命周期规则（Lifecycle）。只有Bucket的拥有者才有权限查看Bucket的生命周期规则。

请求语法

```
GET /?lifecycle HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
Get /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:29 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
```

返回示例

- 已设置生命周期规则的返回示例

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
```

```
Date: Mon, 14 Apr 2014 01:17:29 GMT
Connection: keep-alive
Content-Length: 255
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
  <Rule>
    <ID>delete after one day</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>1</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

- **未设置生命周期规则的返回示例**

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906****
Date: Mon, 14 Apr 2014 01:17:29 GMT
Connection: keep-alive
Content-Length: 278
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <BucketName>oss-example</BucketName>
  <Code>NoSuchLifecycle</Code>
  <Message>No Row found in Lifecycle Table.</Message>
  <RequestId>534B372974E88A4D8906****</RequestId>
  <HostId> BucketName.oss.example.com</HostId>
</Error>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403 Forbidden	没有权限查看Bucket的生命周期规则。只有Bucket的拥有者才能查看Bucket的生命周期规则。
NoSuchBucket或NoSuchLifecycle	404 Not Found	Bucket不存在或没有配置生命周期规则。

6.7 DeleteBucketLifecycle

DeleteBucketLifecycle接口用于删除指定存储空间（Bucket）的生命周期规则。使用DeleteBucketLifecycle接口删除指定Bucket所有的生命周期规则后，该Bucket中的文件（Object）不会被自动删除。只有Bucket的拥有者才能删除该Bucket的生命周期规则。

请求语法

```
DELETE /?lifecycle HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
DELETE /?lifecycle HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:35 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/Cn
****
```

返回示例

说明:

如果删除不存在的Lifecycle，则返回204状态码。

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906****
Date: Mon, 14 Apr 2014 01:17:35 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有删除该Bucket生命周期规则的权限。只有Bucket的拥有者才可以删除Bucket的生命周期规则。

6.8 GetBucket (ListObjects)

GetBucket接口用于列举存储空间（Bucket）中所有文件（Object）的信息。

说明:

Object中自定义的元信息，在GetBucket请求时不会返回。

请求语法

```
GET / HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

请求参数

名称	类型	是否必选	描述
Delimiter	字符串	否	对Object名字进行分组的字符。所有Object名字包含指定的前缀，第一次出现Delimiter字符之间的Object作为一组元素（即CommonPrefixes）。 默认值：无

名称	类型	是否必选	描述
Marker	字符串	否	设定从Marker之后按字母排序开始返回Object。 Marker用来实现分页显示效果，参数的长度必须小于1024字节。 在做条件查询时，即使Marker在列表中不存在，也会从符合Marker字母排序的下一个开始打印。 默认值：无
Max-keys	字符串	否	指定返回Object的最大数。 取值：大于0小于1000 默认值：100 说明: 如果因为Max-keys的设定无法一次完成列举，返回结果会附加一个<NextMarker>作为下一次列举的Marker。
Prefix	字符串	否	限定返回文件的Key必须以Prefix作为前缀。 如果把Prefix设为某个文件夹名，就可以列举以此Prefix开头的文件，即该文件夹下递归的所有文件和子文件夹。 在设置Prefix的基础上增加设置Delimiter为 / 时，返回值就只列举该文件夹下的文件，文件夹下的子文件夹名返回在CommonPrefixes中，子文件夹下递归的所有文件和文件夹不显示。 例如，一个Bucket中存在三个Object : fun/test.jpg ， fun/movie/001.avi，fun/movie/007.avi。若设定Prefix为 fun/，则返回三个Object；如果增加设定Delimiter为 /，则返回fun/test.jpg和fun/movie/ 。 说明: · 参数的长度必须小于1024字节。 · 使用Prefix查询时，返回的Key中仍会包含Prefix。 默认值：无

名称	类型	是否必选	描述
Encoding-type	字符串	否	对返回的内容进行编码并指定编码的类型。 默认值：无 可选值：url 说明: Delimiter、Marker、Prefix、NextMarker以及Key使用UTF-8字符。如果Delimiter、Marker、Prefix、NextMarker以及Key中包含XML 1.0标准不支持的控制字符，您可以通过指定Encoding-type对返回结果中的Delimiter、Marker、Prefix、NextMarker以及Key进行编码。

响应元素

名称	类型	描述
Contents	容器	保存每个返回Object元信息的容器。 父节点：ListBucketResult
CommonPrefixes	字符串	如果请求中指定了Delimiter参数，则会在返回的响应中包含CommonPrefixes元素。该元素表明以Delimiter结尾，并有共同前缀的Object名称的集合。 父节点：ListBucketResult
Delimiter	字符串	对Object名字进行分组的字符。所有名字包含指定的前缀且第一次出现Delimiter字符之间的Object作为一组元素CommonPrefixes。 父节点：ListBucketResult
EncodingType	字符串	指明了返回结果中编码使用的类型。如果请求的参数中指定了Encoding-type，那会对返回结果中的Delimiter、Marker、Prefix、NextMarker和Key这些元素进行编码。 父节点：ListBucketResult

名称	类型	描述
DisplayName	字符串	Object 拥有者的名字。 父节点：ListBucketResult.Contents.Owner
ETag	字符串	ETag (entity tag) 在每个Object生成的时创建，用于标示一个Object的内容。 父节点：ListBucketResult.Contents 对于PutObject请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。不建议您使用ETag来作为Object内容的MD5校验数据完整性。
ID	字符串	Bucket拥有者的用户ID。 父节点：ListBucketResult.Contents.Owner
IsTruncated	枚举字符串	请求中返回的结果是否被截断。 返回值：`true`、`false` · true表示本次没有返回全部结果。 · false表示本次已经返回了全部结果。 父节点：ListBucketResult
Key	字符串	Object的Key。 父节点：ListBucketResult.Contents
LastModified	时间	Object最后被修改的时间。 父节点：ListBucketResult.Contents
ListBucket Result	容器	保存GetBucket请求结果的容器。 子节点：Name、Prefix、 Marker、MaxKeys、Delimiter、IsTruncated、Nextmarker、Contents 父节点：None

名称	类型	描述
Marker	字符串	标明这次GetBucket（ListObjects）的起点。 父节点：ListBucketResult
MaxKeys	字符串	响应请求内返回结果的最大数目。 父节点：ListBucketResult
Name	字符串	Bucket名字。 父节点：ListBucketResult
Owner	容器	保存Bucket拥有者信息的容器。 子节点：DisplayName、ID 父节点：ListBucketResult
Prefix	字符串	本次查询结果的前缀。 父节点：ListBucketResult
Size	字符串	Object的字节数。 父节点：ListBucketResult.Contents
StorageClass	字符串	Object的存储类型。 父节点：ListBucketResult.Contents

示例

- **简单请求示例**

```
GET / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0ykbo
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1866
Connection: keep-alive
```

```
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
<Name>oss-example</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter></Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>fun/movie/001.avi</Key>
        <LastModified>2012-02-24T08:43:07.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user-example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>fun/movie/007.avi</Key>
        <LastModified>2012-02-24T08:43:27.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user-example</DisplayName>
        </Owner>
    </Contents>
<Contents>
        <Key>fun/test.jpg</Key>
        <LastModified>2012-02-24T08:42:32.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user-example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>oss.jpg</Key>
        <LastModified>2012-02-24T06:07:48.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user-example</DisplayName>
        </Owner>
    </Contents>
</ListBucketResult>
```

- **带Prefix参数的请求示例**

```
GET /?prefix=fun HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
```

```
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0ykbo
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1464
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
<Name>oss-example</Name>
<Prefix>fun</Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter></Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>fun/movie/001.avi</Key>
        <LastModified>2012-02-24T08:43:07.000Z</LastModified>
        <ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>fun/movie/007.avi</Key>
        <LastModified>2012-02-24T08:43:27.000Z</LastModified>
        <ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>fun/test.jpg</Key>
        <LastModified>2012-02-24T08:42:32.000Z</LastModified>
        <ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
</ListBucketResult>
```

- **带Prefix和Delimiter参数的请求示例**

```
GET /?prefix=fun/&delimiter=/ HTTP/1.1
```

```
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrnx7xHk3sgysx7I8U9I9IY
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
<Name>oss-example</Name>
<Prefix>fun/</Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter>/</Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>fun/test.jpg</Key>
        <LastModified>2012-02-24T08:42:32.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&****;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>0022012****</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
   <CommonPrefixes>
        <Prefix>fun/movie/</Prefix>
   </CommonPrefixes>
</ListBucketResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- Node.js
- Browser.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有访问该Bucket的权限。
InvalidArgument	400	· Max-keys小于0或者大于1000。 · Prefix，Marker，Delimiter参数的长度不符合要求。

6.9 PutBucketLogging

PutBucketLogging接口用于为存储空间（Bucket）开启访问日志记录功能。访问日志记录功能开启后，OSS将自动记录访问Bucket请求的详细信息，并以小时为单位将访问日志作为一个文件（Object）写入指定的Bucket。

说明:

- 源Bucket被删除时，对应的日志规则也将被删除。
- OSS以小时为单位生成Bucket访问的LOG文件，但并不表示这个小时的所有请求都记录在这个小时的LOG文件内，也有可能出现在上一个或者下一个LOG文件中。访问日志记录功能不保证记录每一条访问信息。
- OSS生成一个Bucket访问的LOG文件，算作一次PUT操作，并记录其占用的空间，但不会记录产生的流量。LOG生成后，您可以按照普通的Object来操作这些LOG文件。
- OSS会忽略掉所有以“x-”开头的query-string参数，但这个query-string会被记录在访问LOG中。如果你想从海量的访问日志中，标示一个特殊的请求，可以在URL中添加一个“x-”开头的query-string参数。例如：

  ```
  http://oss-example.regionid.example.com/aliyun-logo.png

  http://oss-example.regionid.example.com/aliyun-logo.png?x-user=admin
  ```

  OSS处理上面两个请求，结果是一样的。但是在访问LOG时，你可以通过搜索“x-user=admin”定位出该请求。

请求语法

```
PUT /?logging HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Authorization: SignatureValue
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
```

```
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
    <LoggingEnabled>
        <TargetBucket>TargetBucket</TargetBucket>
        <TargetPrefix>TargetPrefix</TargetPrefix>
    </LoggingEnabled>
</BucketLoggingStatus>
```

请求元素

说明:

访问日志记录功能不支持匿名访问，必须带签名。

名称	类型	是否必需	描述
BucketLoggingStatus	容器	是	访问日志状态信息的容器。 子元素：LoggingEnabled 父元素：无
LoggingEnabled	容器	否	访问日志信息的容器。此元素在开启时需要，关闭时不需要。 子元素：TargetBucket, TargetPrefix 父元素：BucketLoggingStatus
TargetBucket	字符串	在开启访问日志的时候必选	指定存储访问日志的Bucket。 **说明:** · 目标Bucket和源Bucket必须属于同一地域。 · 源Bucket和目标Bucket可以是同一个Bucket，也可以是不同的Bucket；用户也可以将多个源Bucket的日志都保存在同一个目标Bucket内（建议指定不同的TargetPrefix）。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

名称	类型	是否必需	描述
TargetPrefix	字符串	否	指定最终被保存的访问日志文件前缀。可以为空。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

存储访问日志记录的Object命名规则

命名格式：

```
<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString
```

命名参数说明：

参数	描述
TargetPrefix	指定Object的前缀。
YYYY-mm-DD-HH-MM-SS	该Object被创建的时间。分别填入年、月、日、小时、分钟和秒。例如2012-09-10-04-00-00。
UniqueString	OSS为访问日志生成的UUID，用于唯一标识该文件。

Object名称示例：

```
MyLog-oss-example-2012-09-10-04-00-00-0000
```

上述示例中，MyLog-表示用户指定的Object前缀；oss-example 表示源Bucket的名称；2012-09-10-04-00-00表示该Object被创建时的北京时间；0000表示OSS系统生成的字符串。

LOG文件格式

说明：

- OSS的LOG中的任何一个字段，都可能出现 “-”，用于表示未知数据或对于当前请求该字段无效。
- 根据需求，OSS的LOG格式将来会在尾部添加一些字段，请在开发LOG处理工具时考虑兼容性问题。

名称	示例	描述
Remote IP	0.0.0.0	请求发起的IP地址（Proxy代理或用户防火墙可能会屏蔽该字段）
Reserved	-	保留字段
Reserved	-	保留字段
Time	[02/May/2012:00:00:04 +0800]	OSS收到请求的时间
Request-URL	GET /aliyun-logo.png HTTP/1.1	用户请求的URL （包括query-string）
HTTP Status	200	OSS返回的HTTP状态码
SentBytes	5576	用户从OSS下载的流量
RequestTime (ms)	71	完成本次请求的时间（毫秒）
Referer	`http://www.aliyun.com/product/oss`	请求的HTTP Referer
User-Agent	curl/7.15.5	HTTP的User-Agent header
HostName	oss-example.regionid.example.com	请求访问域名
Request ID	505B0****5037C2AF032593A	用于唯一标示该请求的UUID
LoggingFlag	true	是否开启了访问日志功能
Requester Aliyun ID	165713****983691	请求者的阿里云ID；匿名访问为 “-”
Operation	GetObject	请求类型
Bucket	oss-example	请求访问的Bucket名称
Key	/aliyun-logo.png	用户请求的Key
ObjectSize	5576	Object大小
Server Cost Time (ms)	17	OSS服务器处理本次请求所花的时间（毫秒）
Error Code	NoSuchBucket	OSS返回的错误码
Request Length	302	用户请求的长度（Byte）
UserID	1657136****83691	Bucket拥有者ID
Delta DataSize	280	Bucket大小的变化量；若没有变化为 “-”
Sync Request	-	是否是CDN回源请求；若不是为 “-”

名称	示例	描述
Reserved	-	保留字段

示例

- 开启Bucket访问日志的请求示例

```
PUT /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZHiA=
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
<LoggingEnabled>
<TargetBucket>doc-log</TargetBucket>
<TargetPrefix>MyLog-</TargetPrefix>
</LoggingEnabled>
</BucketLoggingStatus>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88***8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

- 关闭Bucket访问日志的请求示例

说明:

当关闭一个Bucket的访问日志记录功能时，只要发送一个空的BucketLoggingStatus即可，具体方法可以参考以下示例。

```
PUT /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Type: application/xml
Content-Length: 86
Date: Fri, 04 May 2012 04:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZHiA=
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
</BucketLoggingStatus>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674***A4D8906008B
Date: Fri, 04 May 2012 04:21:12 GMT
Content-Length: 0
Connection: keep-alive
```

```
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	源Bucket不存在。源Bucket和目标Bucket必须属于同一用户。
InvalidTargetBucketForLogging	400	源Bucket和目标Bucket不属于同一个数据中心。
InvalidDigest	400	上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性，如果不一致会返回此错误码。
MalformedXML	400	请求中的XML不合法。
InvalidTargetBucketForLogging	403	请求发起者不是目标 Bucket（TargetBucket）的拥有者。
AccessDenied	403	请求发起者不是源 Bucket（BucketName）的拥有者。

6.10 GetBucketLogging

GetBucketLogging接口用于查看存储空间（Bucket）的访问日志配置。只有Bucket的拥有者才能查看Bucket的访问日志配置。

请求语法

```
GET /?logging HTTP/1.1
```

```
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
BucketLoggingStatus	容器	访问日志状态信息的容器。 子元素：LoggingEnabled 父元素：无 说明： 如果源Bucket未设置日志规则，OSS仍然返回一个XML消息体，但其中的BucketLoggingStatus元素为空。
LoggingEnabled	容器	访问日志信息的容器。此元素在开启时返回，关闭时不返回。 子元素：TargetBucket, TargetPrefix 父元素：BucketLoggingStatus
TargetBucket	字符	指定存放访问日志的Bucket。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled
TargetPrefix	字符	指定最终被保存的访问日志文件前缀。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

示例

请求示例

```
Get /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 04 May 2012 05:31:04 GMT
```

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
```

返回示例

- **已设置日志规则的返回示例**

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 04 May 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 210
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com
">
    <LoggingEnabled>
        <TargetBucket>mybucketlogs</TargetBucket>
        <TargetPrefix>mybucket-access_log/</TargetPrefix>
    </LoggingEnabled>
</BucketLoggingStatus>
```

- **未设置日志规则的返回示例**

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 04 May 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 110
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com
">
</BucketLoggingStatus>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有查看Bucket访问日志配置的权限。只有Bucket的拥有者才能查看Bucket的访问日志配置。

6.11 DeleteBucketLogging

DeleteBucketLogging用于关闭存储空间（Bucket）的访问日志记录功能。只有Bucket的拥有者才有权限关闭Bucket访问日志记录功能。

请求语法

```
DELETE /?logging HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
DELETE /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:35:24 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/Cn
****
```

返回示例

说明:

如果目标Bucket没有开启访问日志记录功能，则返回204状态码。

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 24 Feb 2012 05:35:24 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python

- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有关闭Bucket访问日志记录功能的权限。只有Bucket的拥有者才可以关闭Bucket访问日志记录功能。

6.12 PutBucketWebsite

PutBucketWebsite接口用于将一个bucket设置成静态网站托管模式，以及设置跳转规则。

website

website接口主要有两个功能：

- 设置默认主页和默认404页。
- 设置RoutingRule。RoutingRule用来指定3xx跳转规则以及镜像回源规则。

> 说明：
> 镜像回源支持公共云、金融云。

website字段样例：

```
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <RuleNumber>1</RuleNumber>
      <Condition>
        <KeyPrefixEquals>abc/</KeyPrefixEquals>
        <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <RedirectType>Mirror</RedirectType>
```

```
        <PassQueryString>true</PassQueryString>
        <MirrorURL>http://www.test.com/</MirrorURL>
        <MirrorPassQueryString>true</MirrorPassQueryString>
        <MirrorFollowRedirect>true</MirrorFollowRedirect>
        <MirrorCheckMd5>false</MirrorCheckMd5>
        <MirrorHeaders>
          <PassAll>true</PassAll>
          <Pass>myheader-key1</Pass>
          <Pass>myheader-key2</Pass>
          <Remove>myheader-key3</Remove>
          <Remove>myheader-key4</Remove>
          <Set>
            <Key>myheader-key5</Key>
            <Value>myheader-value5</Value>
          </Set>
        </MirrorHeaders>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <RuleNumber>2</RuleNumber>
      <Condition>
        <KeyPrefixEquals>abc/</KeyPrefixEquals>
        <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
        <IncludeHeader>
          <Key>host</Key>
          <Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
        </IncludeHeader>
      </Condition>
      <Redirect>
        <RedirectType>AliCDN</RedirectType>
        <Protocol>http</Protocol>
        <HostName>www.test.com</HostName>
        <PassQueryString>false</PassQueryString>
        <ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
        <HttpRedirectCode>301</HttpRedirectCode>
      </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

请求语法

```
PUT /?website HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
    <IndexDocument>
        <Suffix>index.html</Suffix>
    </IndexDocument>
    <ErrorDocument>
        <Key>errorDocument.html</Key>
    </ErrorDocument>
```

```
</WebsiteConfiguration>
```

请求元素（Request Elements）

名称	类型	描述	是否必须
WebsiteConfiguration	容器	根节点 父节点：无	是
IndexDocument	容器	默认主页的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。
Suffix	字符串	默认主页 如果设置，则访问以正斜线（/）结尾的object时都会返回此默认主页。 父节点：IndexDocument	有条件，父节点IndexDocument指定时必须指定。
ErrorDocument	容器	404页面的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。
Key	容器	404页面 如果指定，则访问的object不存在时则返回此404页面。 父节点：ErrorDocument	有条件，父节点ErrorDocument指定时必须指定。
RoutingRules	容器	RoutingRule的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。

RoutingRule	容器	指定跳转规则或者镜像回源规则，最多指定5个RoutingRule。 父节点：RoutingRules	否
RuleNumber	正整数	匹配和执行RoutingRule的序号，匹配时会按照此序号按顺序匹配规则。如果匹配成功，则执行此规则，后续的规则不再执行。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
Condition	容器	匹配的条件 如果指定的项都满足，则执行此规则。此容器下的各个节点关系是与，即需要所有条件都满足才算匹配。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
KeyPrefixEquals	字符串	只有匹配此前缀的object才能匹配此规则。 父节点：Condition	否
HttpErrorCodeReturnedEquals	HTTP状态码	访问指定object时返回此status才能匹配此规则。当跳转规则是镜像回源类型时，此字段必须为404。 父节点：Condition	否
IncludeHeader	容器	只有请求中包含了指定header，且值为指定值时，才能匹配此规则。此容器可以最多重复5个。 父节点：Condition	否
Key	字符串	只有请求中包含了此header，且值为Equals指定值时，才能匹配此规则。 父节点：IncludeHeader	有条件，父节点IncludeHeader指定时必须指定。

Equals	字符串	只有请求中包含了Key指定的header，且值为指定的值时，才能匹配此规则 父节点：IncludeHeader	有条件，父节点IncludeHeader指定时必须指定。
Redirect	容器	指定匹配此规则后执行的动作。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
RedirectType	字符串	指定跳转的类型，取值必须为以下几项。 · Mirror（镜像回源） · External（外部跳转，即OSS会返回一个3xx请求，指定跳转到另外一个地址。） · Internal（内部跳转，即OSS会根据此规则将访问的object1转换成另外一个object2，相当于用户访问的是object2。） · AliCDN（阿里云CDN跳转，主要用于阿里云的CDN。与External不同的是，OSS会额外添加一个header。阿里云CDN识别到此header后会主动跳转到指定的地址，返回给用户获取到的数据，而不是将3xx跳转请求返回给客户。） 父节点：Redirect	有条件，父节点Redirect指定时必须指定
PassQueryString	bool型	执行跳转或者镜像回源时，是否要携带发起请求的请求参数，取值true或者false。 用户请求OSS时携带了请求参数”?a=b&c=d”，并且此项设置为true，那么如果规则是302跳转，则在跳转的Location头中会添加此请求参数。如”Location:www.test.com?a=b&c=d”，跳转类型是镜像回源，则在发起的回源请求中也会携带此请求参数。 默认值：false 父节点：Redirect	否

MirrorURL	字符串	只有在RedirectType为Mirror时有意义，表示镜像回源的源站地址。 如以http://或者https://开头，必须以正斜线（/）结尾，OSS会在此字符串的基础上加上object构成回源的url。比如指定为`http://www.test.com/`，访问的obejct是”myobject”，则回源的url为`http://www.test.com/myobject`，如果指定为`http://www.test.com/dir1/`，那么回源的url为`http://www.test.com/dir1/myobject`。 父节点：Redirect	有条件，如果RedirectType为Mirror则必须指定
MirrorPassQueryString	bool型	与PassQueryString作用相同，优先级比PassQueryString高，但只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorFollowRedirect	bool型	如果镜像回源获取的结果是3xx，是否要继续跳转到指定的Location获取数据。 比如发起镜像回源请求时，如果源站返回了302，并且指定了Location。如果此项为true，则oss会继续请求Location指定的地址（最多跳转10次，如果超过10次，则返回镜像回源失败）。如果为false，那么OSS就会返回302，并将Location透传出去。只有在RedirectType为Mirror时生效。 默认值：true 父节点：Redirect	否

MirrorCheckMd5	bool型	是否要检查回源body的md5。 当此项为true且源站返回的response中含有Content-Md5头时，OSS检查拉取的数据md5是否与此header匹配，如果不匹配，则不保存在oss上。只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorHeaders	容器	用于指定镜像回源时携带的header。只有在RedirectType为Mirror时生效。 父节点：Redirect	否
PassAll	bool型	是否透传请求中所有的header（除了保留的几个header以及以oss-/x-oss-/x-drs-开头的header）到源站。只有在RedirectType为Mirror时生效。 默认值：false 父节点：MirrorHeaders	否
Pass	字符串	透传指定的header到源站，最多10个，每个header长度最多1024个字节，字符集为0-9、A-Z、a-z以及横杠。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Remove	字符串	禁止透传指定的header到源站，这个字段可以重复，最多10个，一般与PassAll一起使用。每个header长度最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否

Set	容器	设置一个header传到源站，不管请求中是否携带这些指定的header，回源时都会设置这些header。该容器可以重复，最多10组。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Key	字符串	设置header的key，最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定
Value	字符串	设置header的value，最多1024个字节，不能出现”\r\n”。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定。
Protocol	字符串	跳转时的协议，只能取值为http或者https。比如访问的文件为test，设定跳转到www.test.com，而且Protocol字段为https，那么Location头为https://www.test.com/test。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType为External或者AliCDN时需要指定。
HostName	字符串	跳转时的域名，需要符合域名规范。比如访问的object为test，Protocol为https，HostName指定为www.test.com，那么Location头为https://www.test.com/test。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType为External或者AliCDN时需要指定。

HttpRedirectCode	HTTP状态码	跳转时返回的状态码，取值为301、302或307。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType为External或者AliCDN时需要指定。
ReplaceKeyPrefixWith	字符串	Redirect的时候object name的前缀将替换成这个值。如果前缀为空则将这个字符串插入在object namde的最前面。ReplaceKeyWith和ReplaceKeyPrefixWith这两个节点只能共存一个。 比如指定了KeyPrefixEquals为abc/，指定了ReplaceKeyPrefixWith为def/，那么如果访问的object为abc/test.txt那么Location头则为`http://www.test.com/def/test.txt`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType为Internal、External或者AliCDN时需要指定。
ReplaceKeyWith	字符串	Redirect的时候object name将替换成这个值，可以支持变量（目前支持的变量是${key}，代表着该请求中的object name）。ReplaceKeyWith和ReplaceKeyPrefixWith这两个节点只能共存一个。 比如设置ReplaceKeyWith为prefix/${key}.suffix，访问的object为test，那么Location头则为`http://www.test.com/prefix/test.suffix`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType为Internal、External或者AliCDN时需要指定。

细节分析

- 静态网站是指所有的网页都由静态内容构成，包括客户端执行的脚本，例如JavaScript；OSS不支持涉及到需要服务器端处理的内容，例如PHP，JSP，APS.NET等。
- 如果你想使用自己的域名来访问基于bucket的静态网站，可以通过域名CNAME来实现。具体配置方法请参见绑定自定义域名。
- 用户将一个bucket设置成静态网站托管模式时，必须指定索引页面，错误页面则是可选的。
- 用户将一个bucket设置成静态网站托管模式时，指定的索引页面和错误页面是该bucket内的一个object。
- 在将一个bucket设置成静态网站托管模式后，对静态网站根域名的匿名访问，OSS将返回索引页面；对静态网站根域名的签名访问，OSS将返回Get Bucket结果。
- 如果用户上传了Content-MD5请求头，OSS会计算body的Content-MD5并检查一致性，如果不一致，将返回InvalidDigest错误码。

示例

请求示例

```
PUT /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 209
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqx******k53otfjbyc:KU5h8YM******0dXqf3JxrT
ZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
</WebsiteConfiguration>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

完整代码

```
PUT /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 2064
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBN******QMf8u:sNKIHT6ci/z231yIT5vYnetDLu4=
```

```
User-Agent: aliyun-sdk-python-test/0.4.0

<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
<RoutingRules>
<RoutingRule>
<RuleNumber>1</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
</Condition>
<Redirect>
<RedirectType>Mirror</RedirectType>
<PassQueryString>true</PassQueryString>
<MirrorURL>http://www.test.com/</MirrorURL>
<MirrorPassQueryString>true</MirrorPassQueryString>
<MirrorFollowRedirect>true</MirrorFollowRedirect>
<MirrorCheckMd5>false</MirrorCheckMd5>
<MirrorHeaders>
<PassAll>true</PassAll>
<Pass>myheader-key1</Pass>
<Pass>myheader-key2</Pass>
<Remove>myheader-key3</Remove>
<Remove>myheader-key4</Remove>
<Set>
<Key>myheader-key5</Key>
<Value>myheader-value5</Value>
</Set>
</MirrorHeaders>
</Redirect>
</RoutingRule>
<RoutingRule>
<RuleNumber>2</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
<IncludeHeader>
<Key>host</Key>
<Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
</IncludeHeader>
</Condition>
<Redirect>
<RedirectType>AliCDN</RedirectType>
<Protocol>http</Protocol>
<HostName>www.test.com</HostName>
<PassQueryString>false</PassQueryString>
<ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
<HttpRedirectCode>301</HttpRedirectCode>
</Redirect>
</RoutingRule>
</RoutingRules>
</WebsiteConfiguration>

HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5B5ADFD6ED3CC49176CBE29D
```

```
x-oss-server-time: 47
```

6.13 GetBucketWebsite

GetBucketWebsite接口用于查看存储空间（Bucket）的静态网站托管状态以及跳转规则。

请求语法

```
GET /?website HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
WebsiteConfiguration	容器	根节点 父节点：无
IndexDocument	容器	默认主页的容器 父节点：WebsiteConfiguration
Suffix	字符串	默认主页 父节点：IndexDocument
ErrorDocument	容器	404页面的容器 父节点：WebsiteConfiguration
Key	容器	404页面 父节点：ErrorDocument
RoutingRules	容器	RoutingRule的容器 父节点：WebsiteConfiguration
RoutingRule	容器	跳转规则或者镜像回源规则 父节点：RoutingRules

名称	类型	描述
RuleNumber	正整数	匹配和执行RoutingRule的序号 匹配时按照此序号顺序进行规则匹配。如果匹配成功，则执行此规则，且后续的规则不再执行。 父节点：RoutingRule
Condition	容器	匹配的条件 如果指定的项都满足，则执行此规则。此容器下的各个节点关系是与，即需所有条件都满足才算匹配。 父节点：RoutingRule
KeyPrefixEquals	字符串	只有匹配此前缀的Object才能匹配此规则。 父节点：Condition
HttpErrorCodeReturnedEquals	HTTP状态码	访问指定Object时返回此状态码才能匹配此规则。当跳转规则是镜像回源类型时，此字段必须为404。 父节点：Condition
IncludeHeader	容器	只有请求中包含了指定Header，且值为指定值时，才能匹配此规则。此容器可以最多重复5个。 父节点：Condition
Key	字符串	只有请求中包含了此头，且值为Equals指定值时，才能匹配此规则。 父节点：IncludeHeader
Equals	字符串	只有请求中包含了Key指定的头，且值为指定的值时，才能匹配此规则。 父节点：IncludeHeader
Redirect	容器	指定匹配此规则后执行的动作。 父节点：RoutingRule

名称	类型	描述
RedirectType	字符串	跳转的类型，取值： · Mirror（镜像回源） · External（外部跳转，即OSS会返回一个3xx请求，指定跳转到另外一个地址。） · Internal（内部跳转，即OSS会根据此规则将访问的object1转换成另外一个object2，相当于用户访问的是object2。） · AliCDN（阿里云CDN跳转，主要用于阿里云的CDN。与External不同的是，OSS会额外添加一个header。阿里云CDN识别到此header后会主动跳转到指定的地址，返回给用户获取到的数据，而不是将3xx跳转请求返回给客户。） 父节点：Redirect
PassQueryString	bool型	执行跳转或者镜像回源时，是否要携带发起请求的请求参数。 当用户请求OSS时携带了请求参数”?a=b&c=d”，且此项设置为true，那么如果规则是302跳转，则在跳转的Location头中会添加此请求参数。例如请求时携带了”Location:www.test.com?a=b&c=d”，跳转类型是镜像回源，则在发起的回源请求中也会携带此请求参数。 默认值：false 父节点：Redirect
MirrorURL	字符串	只有在RedirectType为Mirror时有意义，表示镜像回源的源站地址。 如以http://或https://开头，则必须以正斜线（/）结尾，OSS会在此字符串的基础上加上Object构成回源的URL。例如，访问的Obejct是”myobject”，如指定地址为`http://www.test.com/`，则回源的URL为`http://www.test.com/myobject`；如指定地址为`http://www.test.com/dir1/`，那么回源的URL为`http://www.test.com/dir1/myobject`。 父节点：Redirect

名称	类型	描述
MirrorPassQueryString	bool型	与PassQueryString作用相同，优先级比PassQueryString高，但只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect
MirrorFollowRedirect	bool型	如镜像回源获取的结果是3xx，是否要继续跳转到指定的Location获取数据。 例如，发起镜像回源请求时，源站返回了302，并且指定了Location。如果此项为true，则OSS会继续请求Location指定的地址（最多跳转10次，如果超过10次，则返回镜像回源失败）；如此项为false，则OSS会返回302，并将Location透传出去。此参数仅在RedirectType为Mirror时生效。 默认值：true 父节点：Redirect
MirrorCheckMd5	bool型	是否要检查回源主体的MD5。 当此项为true且源站返回的response中含有Content-MD5头时，OSS会检查拉取的数据MD5是否与此头匹配，如不匹配，则不保存在OSS上。此参数仅在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect
MirrorHeaders	容器	用于指定镜像回源时携带的头。此参数仅在在RedirectType为Mirror时生效。 父节点：Redirect
PassAll	bool型	是否透传请求中所有的header（除了保留的几个header以及以oss-/x-oss-/x-drs-开头的header）到源站。此参数仅在RedirectType为Mirror时生效。 默认值：false 父节点：MirrorHeaders

名称	类型	描述
Pass	字符串	透传指定的header到源站，最多10个，每个header长度最多1024个字节，字符集为0-9、A-Z、a-z以及横杠。此参数仅在RedirectType为Mirror时生效。 父节点：MirrorHeaders
Remove	字符串	禁止透传指定的header到源站，这个字段可以重复，最多10个，一般与PassAll一起使用。每个header长度最多1024个字节，字符集与Pass相同。此参数仅在RedirectType为Mirror时生效。 父节点：MirrorHeaders
Set	容器	设置一个header传到源站，不管请求中是否携带这些指定的header，回源时都会设置这些header。该容器可以重复，最多10组。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders
Key	字符串	设置header的key，最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：Set
Value	字符串	设置header的value，最多1024个字节，且不能出现”\r\n”。仅在RedirectType为Mirror时生效。 父节点：Set
Protocol	字符串	跳转时的协议。例如，访问的文件为test，设定跳转到`www.test.com`，且Protocol字段为https，则Location的头为`https://www.test.com/test`。 此参数仅在RedirectType为External或AliCDN时生效。 取值：http、https 父节点：Redirect

名称	类型	描述
HostName	字符串	跳转时的域名。域名需符合域名规范。例如，访问的Object为test，Protocol为https，Hostname指定为www.test.com，则Location的头为https://www.test.com/test。参数仅在RedirectType为External或者AliCDN时生效。 父节点：Redirect
HttpRedirectCode	HTTP状态码	跳转时返回的状态码。此参数仅在RedirectType为External或者AliCDN时生效。 取值：301、302、307 父节点：Redirect
ReplaceKeyPrefixWith	字符串	Redirect时Object Name的前缀将替换成此值。如果前缀为空则将这个字符串插入在Object Name的最前面。ReplaceKeyWith和ReplaceKeyPrefixWith这两个节点只能共存一个。 例如，指定KeyPrefixEquals为abc/，指定ReplaceKeyPrefixWith为def/，如果访问的Object为abc/test.txt，则Location的头为http://www.test.com/def/test.txt。 此参数仅在RedirectType为Internal、External或者AliCDN时生效。 父节点：Redirect

名称	类型	描述
ReplaceKeyWith	字符串	执行Redirect时文件名将替换成此值。此参数支持变量（目前支持的变量是${key}，代表该请求中的Object Name）。ReplaceKeyWith和ReplaceKeyPrefixWith两个节点只能共存一个。 例如，设置ReplaceKeyWith为prefix/${key}.suffix，访问的Object为test，则Location的头为`http://www.test.com/prefix/test.suffix`。 此参数仅在RedirectType为Internal、External或者AliCDN时生效。 父节点：Redirect

示例

请求示例

```
Get /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqx******k53otfjbyc: BuG4rRK+zNh******
1NNHD39zXw=
```

返回示例

- **已设置LOG规则的返回示例**

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
Content-Length: 218
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com
”>
<IndexDocument>
<Suffix>index.html</Suffix>
    </IndexDocument>
    <ErrorDocument>
        <Key>error.html</Key>
    </ErrorDocument>
</WebsiteConfiguration>
```

- **未设置LOG规则的返回示例**

```
HTTP/1.1 404
```

```
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:56:46 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <Code>NoSuchWebsiteConfiguration</Code>
    <Message>The specified bucket does not have a website configurat
ion.</Message>
    <BucketName>oss-example</BucketName>
    <RequestId>505191BEC4689A033D00236F</RequestId>
    <HostId>oss-example.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

完整代码

```
GET /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:07:41 GMT
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBN******QMf8u:0Jzamofmy******sU9HUWomxsus=
User-Agent: aliyun-sdk-python-test/0.4.0

HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:07:41 GMT
Content-Type: application/xml
Content-Length: 2102
Connection: keep-alive
x-oss-request-id: 5B5AE0DD2F7938C45FCED4BA
x-oss-server-time: 47

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
<RoutingRules>
<RoutingRule>
<RuleNumber>1</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
</Condition>
<Redirect>
<RedirectType>Mirror</RedirectType>
<PassQueryString>true</PassQueryString>
<MirrorURL>http://www.test.com/</MirrorURL>
<MirrorPassQueryString>true</MirrorPassQueryString>
<MirrorFollowRedirect>true</MirrorFollowRedirect>
<MirrorCheckMd5>false</MirrorCheckMd5>
<MirrorHeaders>
<PassAll>true</PassAll>
<Pass>myheader-key1</Pass>
<Pass>myheader-key2</Pass>
<Remove>myheader-key3</Remove>
<Remove>myheader-key4</Remove>
<Set>
<Key>myheader-key5</Key>
```

```
<Value>myheader-value5</Value>
</Set>
</MirrorHeaders>
</Redirect>
</RoutingRule>
<RoutingRule>
<RuleNumber>2</RuleNumber>
<Condition>
<IncludeHeader>
<Key>host</Key>
<Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
</IncludeHeader>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
</Condition>
<Redirect>
<RedirectType>AliCDN</RedirectType>
<Protocol>http</Protocol>
<HostName>www.test.com</HostName>
<PassQueryString>false</PassQueryString>
<ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
<HttpRedirectCode>301</HttpRedirectCode>
</Redirect>
</RoutingRule>
</RoutingRules>
</WebsiteConfiguration>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C++
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有相应的操作权限。只有Bucket的拥有者才可以查看Bucket的静态网站托管状态。

错误码	HTTP 状态码	描述
NoSuchWebsiteConfiguration	404	目标Bucket未设置静态网站托管功能。

6.14 DeleteBucketWebsite

DeleteBucketWebsite接口用于关闭存储空间（Bucket）的静态网站托管模式以及跳转规则。

只有Bucket的拥有者才能关闭Bucket的静态网站托管模式。

请求语法

```
DELETE /?website HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
DELETE /?website HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqx******k53otfjbyc:LnM4AZ1O******5vGFWicOME
kVg=
```

返回示例

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

完整代码

```
DELETE /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:10:52 GMT
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBN******QMf8u:qPrKwuM******fk1pqTCylFs1jY=
User-Agent: aliyun-sdk-python-test/0.4.0

HTTP/1.1 204 No Content
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:10:52 GMT
Content-Length: 0
Connection: keep-alive
```

```
x-oss-request-id: 5B5AE19C188DC1CE81DAD7C8
```

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404 Not Found	目标Bucket不存在。
AccessDenied	403 Forbidden	没有关闭Bucket静态网站托管模式的权限。只有Bucket的拥有者才能关闭Bucket的静态网站托管模式。

6.15 PutBucketReferer

PutBucketReferer接口用于设置存储空间（Bucket）的防盗链（Referer）。您可以使用此接口设置Referer的访问白名单以及是否允许Referer字段为空。

请求语法

```
PUT /?referer HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss.aliyuncs.com
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
    <RefererList>
        <Referer> http://www.aliyun.com</Referer>
        <Referer> https://www.aliyun.com</Referer>
        <Referer> http://www.*.com</Referer>
        <Referer> https://www.?.aliyuncs.com</Referer>
    </RefererList>
</RefererConfiguration>
```

请求元素

名称	类型	是否必需	描述
RefererConfiguration	容器	是	保存Referer配置内容的容器。 子节点：AllowEmptyReferer，RefererList 父节点：无

名称	类型	是否必需	描述
AllowEmptyReferer	枚举字符串	是	指定是否允许Referer字段为空的请求访问。指定的配置将替换原配置。 取值：true（默认值），false 父节点：RefererConfiguration
RefererList	容器	是	保存Referer访问白名单的容器。 说明：PutBucketReferer操作将用RefererList中的白名单列表覆盖之前配置的白名单列表。当您上传的RefererList为空时（不包含Referer请求元素），此操作会覆盖已配置的白名单列表，即删除之前配置的RefererList。 父节点：RefererConfiguration 子节点：Referer
Referer	字符串	否	指定一条Referer访问白名单。 父节点：RefererList

示例

- **请求示例**
  - **不包含Referer的请求示例**

```
PUT /?referer HTTP/1.1
Host: BucketName.oss.example.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZH****

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

  - **包含Referer的请求示例**

```
PUT /?referer HTTP/1.1
Host: BucketName.oss.example.com
Content-Length: 247
```

```
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZ****

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList>
<Referer> http://www.aliyun.com</Referer>
<Referer> https://www.aliyun.com</Referer>
<Referer> http://www.*.com</Referer>
<Referer> https://www.?.aliyuncs.com</Referer>
</ RefererList>
</RefererConfiguration>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403	没有操作权限。只有Bucket的拥有者才能发起PutBucketReferer请求。
InvalidDigest	400	上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性，如果不一致会返回此错误码。

6.16 GetBucketReferer

GetBucketReferer接口用于查看存储空间（Bucket）的防盗链（Referer）相关配置。

请求语法

```
GET /?referer HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
RefererConfiguration	容器	保存Referer配置内容的容器。 父节点：无 子节点：AllowEmptyReferer、RefererList
AllowEmptyReferer	枚举字符串	指定是否允许Referer字段为空的请求访问。 取值：true、false 父节点：RefererConfiguration
RefererList	容器	保存Referer访问白名单的容器。 父节点：RefererConfiguration 子节点：Referer
Referer	字符串	指定一条Referer的访问白名单。 父节点：RefererList

示例

请求示例

```
Get /?referer HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39
****
```

返回示例

- **已设置Referer规则的返回示例**

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
Content-Length: 218
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
    <RefererList>
        <Referer> http://www.aliyun.com</Referer>
        <Referer> https://www.aliyun.com</Referer>
        <Referer> http://www.*.com</Referer>
        <Referer> https://www.?.aliyuncs.com</Referer>
    </RefererList>
</RefererConfiguration>
```

- **未设置Referer规则的返回示例**

说明:

如果Bucket未进行Referer相关配置，OSS会返回默认的AllowEmptyReferer值和空的RefererList。

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Thu, 13 Sep 2012 07:56:46 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- PHP
- Python
- Go
- C
- .NET
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有查看Bucket的Referer配置信息的权限。只有Bucket的拥有者才能查看Bucket的Referer配置信息。

6.17 GetBucketLocation

GetBucketLocation接口用于查看存储空间（Bucket）的位置信息。只有Bucket的拥有者才能查看Bucket的位置信息。

请求语法

```
GET /?location HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
LocationConstraint	字符串	Bucket所在的地域 有效值： oss-cn-hangzhou、oss-cn-qingdao、oss-cn-beijing、oss-cn-hongkong、oss-cn-shenzhen、oss-cn-shanghai、oss-us-west-1、oss-us-east-1、oss-ap-southeast-1 说明: Bucket所在地域与数据中心的对应关系参见访问域名与数据中心。

示例

请求示例

```
Get /?location HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 04 May 2012 05:31:04 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3
****
```

返回示例

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Fri, 15 Mar 2013 05:31:04 GMT
Connection: keep-alive
Content-Length: 90
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
oss-cn-hangzhou</LocationConstraint >
```

SDK

此接口所对应的各语言SDK如下：

- Java
- PHP
- Go
- C

错误码

错误码	HTTP 状态码	描述
AccessDenied	403	没有查看Bucket位置信息的权限。只有Bucket的拥有者才能查看Bucket的位置信息。

6.18 GetBucketInfo

GetBucketInfo接口用于查看存储空间（Bucket）的相关信息。只有Bucket的拥有者才能查看Bucket的信息。

说明:

请求可以从任何一个OSS的Endpoint发起。

请求语法

```
GET /?bucketInfo HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
```

```
Authorization: SignatureValue
```

响应元素

名称	类型	描述
BucketInfo	容器	保存Bucket信息内容的容器 子节点：Bucket节点 父节点：无
Bucket	容器	保存Bucket具体信息的容器 父节点：BucketInfo 节点
CreationDate	时间	Bucket创建时间 时间格式：2013-07-31T10:56:21.000Z 父节点：BucketInfo.Bucket
ExtranetEndpoint	字符串	Bucket的外网域名 父节点：BucketInfo.Bucket
IntranetEndpoint	字符串	同区域ECS访问Bucket的内网域名 父节点：BucketInfo.Bucket
Location	字符串	Bucket的地域 父节点：BucketInfo.Bucket
Name	字符串	Bucket名称 父节点：BucketInfo.Bucket
Owner	容器	用于存放Bucket拥有者信息的容器 父节点：BucketInfo.Bucket
ID	字符串	Bucket拥有者的用户ID 父节点：BucketInfo.Bucket.Owner

名称	类型	描述
DisplayName	字符串	Bucket拥有者的名称 (目前和用户ID一致) 父节点：BucketInfo.Bucket.Owner
AccessControlList	容器	存储ACL信息的容器 父节点：BucketInfo.Bucket
Grant	枚举字符串	Bucket的ACL权限 有效值：private、public-read、public-read-write 父节点：BucketInfo.Bucket.AccessControlList
DataRedundancyType	枚举字符串	数据容灾类型 有效值：LRS、ZRS 父节点：BucketInfo.Bucket
StorageClass	字符串	Bucket存储类型 有效值：Standard、IA、Archive
Comment	字符串	Bucket的备注内容 父节点：BucketInfo.Bucket
ServerSideEncryptionRule	容器	服务端加密规则的容器。 父节点：BucketInfo.Bucket
ApplyServerSideEncryptionByDefault	容器	服务端默认加密方式的容器。 父节点：BucketInfo.Bucket
SSEAlgorithm	字符串	显示服务端默认加密方式。 有效值：KMS、AES256

名称	类型	描述
KMSMasterKeyID	字符串	显示当前使用的KMS密钥ID。仅当SSEAlgorithm为KMS，且指定了密钥ID时返回取值。其他情况下，返回为空。

示例

请求示例

```
Get /?bucketInfo HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Sat, 12 Sep 2015 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39
****
```

返回示例

- 成功获取Bucket信息的返回示例

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906****
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 531
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<BucketInfo>
  <Bucket>
    <CreationDate>2013-07-31T10:56:21.000Z</CreationDate>
    <ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEndpoint
>
    <IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</
IntranetEndpoint>
    <Location>oss-cn-hangzhou</Location>
    <Name>oss-example</Name>
    <Owner>
      <DisplayName>username</DisplayName>
      <ID>27183473914****</ID>
    </Owner>
    <AccessControlList>
      <Grant>private</Grant>
    </AccessControlList>
    <Comment>test</Comment>
  </Bucket>
</BucketInfo>
```

- 获取不存在的Bucket信息的返回示例

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906****
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Error>
  <Code>NoSuchBucket</Code>
  <Message>The specified bucket does not exist.</Message>
  <RequestId>568D547F31243C673BA1****</RequestId>
  <HostId>nosuchbucket.oss.aliyuncs.com</HostId>
  <BucketName>nosuchbucket</BucketName>
</Error>
```

- 获取没有权限访问的Bucket信息的返回示例

```
HTTP/1.1 403
x-oss-request-id: 534B371674E88A4D8906****
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 209
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>568D5566F2D0F89F5C0E****</RequestId>
  <HostId>test.oss.aliyuncs.com</HostId>
</Error>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有查看该Bucket信息的权限。只有Bucket的拥有者才能查看Bucket的信息。

6.19 PutBucketEncryption

PutBucketEncryption接口用于配置Bucket的加密规则。

请求语法

```
PUT /?encryption HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss.aliyuncs.com
```

```
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
<ServerSideEncryptionRule>
  <ApplyServerSideEncryptionByDefault>
    <SSEAlgorithm>AES256</SSEAlgorithm>
    <KMSMasterKeyID></KMSMasterKeyID>
  </ApplyServerSideEncryptionByDefault>
</ServerSideEncryptionRule>
```

请求元素

名称	类型	是否必需	描述
ServerSideEncryptionRule	容器	是	服务端加密规则的容器。 子元素：ApplyServerSideEncryptionByDefault
ApplyServerSideEncryptionByDefault	容器	是	服务端默认加密方式的容器。 子元素：SSEAlgorithm, KMSMasterKeyID
SSEAlgorithm	字符串	是	设置服务端默认加密方式。 取值：KMS、AES256
KMSMasterKeyID	字符串	否	当SSEAlgorithm值为KMS，且使用指定的密钥加密时，需输入密钥ID。其他情况下，必须为空。

细节分析

- 只有Bucket的拥有者及授权子账户才能为Bucket设置加密规则，否则返回403错误。
- 使用KMS密钥功能时会产生少量的KMS密钥API调用费用，费用详情请参考KMS计费标准。
- 当SSEAlgorithm取值不是KMS或者AES256，则返回400 InvalidEncryptionAlgorithmError错误，错误提示：The Encryption request you specified is not valid. Supported value: AES256/KMS。
- 当SSEAlgorithm取值为AES256，填写了KMSMasterKeyID，则返回400 InvalidArgument错误，错误提示：KMSMasterKeyID is not applicable if the default sse algorithm is not KMS。

示例

- 请求示例

```
PUT /?encryption HTTP/1.1
```

```
Date: Tue, 20 Dec 2018 11:09:13 GMT
Content-Length: ContentLength
Content-Type: application/xml
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS qn6qrrqxo2oawuk53otf****:ceOEyZavKY4QcjoUWYSpYbJ3
****
<?xml version="1.0" encoding="UTF-8"?>
<ServerSideEncryptionRule>
  <ApplyServerSideEncryptionByDefault>
    <SSEAlgorithm>KMS</SSEAlgorithm>
    <KMSMasterKeyID>9468da86-3509-4f8d-a61e-6eab1eac****</KMSMasterK
eyID>
  </ApplyServerSideEncryptionByDefault>
</ServerSideEncryptionRule>
```

- **返回示例**

```
HTTP/1.1 200 OK
x-oss-request-id: 5C1B138A109F4E405B2D8AEF
Date: Thu, 20 Dec 2018 11:11:06 GMT
```

6.20 GetBucketEncryption

GetBucketEncryption接口用于获取Bucket加密规则。

请求语法

```
Get /?encryption HTTP/1.1
Date: GMT Date
Host: BucketName.oss.aliyuncs.com
Authorization: SignatureValue
```

响应元素

名称	类型	是否必需	描述
ServerSideEncryptionRule	容器	是	服务端加密规则的容器。 子元素：ApplyServerSideEncryptionByDefault
ApplyServerSideEncryptionByDefault	容器	是	服务端默认加密方式的容器。 子元素：SSEAlgorithm, KMSMasterKeyID
SSEAlgorithm	字符串	是	显示服务端默认加密方式。 取值：KMS、AES256

名称	类型	是否必需	描述
KMSMasterKeyID	字符串	否	显示当前使用的KMS密钥ID。仅当SSEAlgorithm为KMS且指定了密钥ID时返回，其他情况下，取值为空。

细节分析

- 只有Bucket的拥有者及授权子账户才能查看Bucket的加密规则，否则返回403 Forbidden错误，错误码：AccessDenied。
- 如果Bucket不存在，返回404错误。错误码：NoSuchBucket。
- 如果Bucket未设置加密规则，返回404错误。错误码：NoSuchServerSideEncryptionRule。

示例

- 请求示例

```
Get /?encryption HTTP/1.1
Date: Tue, 20 Dec 2018 11:20:10 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS qn6qrrqxo2oawuk53otf****:ceOEyZavKY4QcjoUWYSpYbJ3
****
```

- 返回示例

```
HTTP/1.1 204 NoContent
x-oss-request-id: 5C1B138A109F4E405B2D8AEF
Date: Tue, 20 Dec 2018 11:22:05 GMT
<?xml version="1.0" encoding="UTF-8"?>
<ServerSideEncryptionRule>
  <ApplyServerSideEncryptionByDefault>
    <SSEAlgorithm>KMS</SSEAlgorithm>
    <KMSMasterKeyID>9468da86-3509-4f8d-a61e-6eab1eac****</KMSMasterK
eyID>
  </ApplyServerSideEncryptionByDefault>
</ServerSideEncryptionRule>
```

6.21 DeleteBucketEncryption

DeleteBucketEncryption接口用于删除Bucket加密规则。

请求语法

```
DELETE /?encryption HTTP/1.1
Date: GMT Date
Host: BucketName.oss.aliyuncs.com
```

```
Authorization: SignatureValue
```

细节分析

- 只有Bucket的拥有者及授权子账户才能删除Bucket的加密规则，否则返回403 Forbidden错误，错误码：AccessDenied。
- 如果Bucket不存在，返回404错误。错误码：NoSuchBucket。

示例

- 请求示例

```
DELETE /?encryption HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 20 Dec 2018 11:35:24 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:6ZVHOehYzxoC1yxRydPQs/Cn
****
```

- 响应示例

```
HTTP/1.1 204 OK
x-oss-request-id: 5C22E0EFD127F6810B1A92A8
Date: Tue, 20 Dec 2018 11:37:05 GMT
Connection: keep-alive
Content-Length: 0
```

6.22 PutBucketRequestPayment

PutBucketRequestPayment接口用于设置请求者付费模式。

 说明:

您可以指定Bucket的付费类型为BucketOwner或Requester。

- 一旦Bucket设置为Requester付费，匿名请求将被拒绝访问。
- 非Owner访问设置为Requester付费的Bucket时，需要携带请求头"x-oss-request-payer: requester"，表明请求者知晓付费策略，否则将被拒绝访问。服务器响应中将携带"x-oss-request-charged: requester"。

请求语法

```
PUT /?requestPayment HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss.aliyuncs.com
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration>
  <Payer>Requester</Payer>
```

```
</RequestPaymentConfiguration>
```

请求元素

名称	类型	是否必需	描述
RequestPaymentConfiguration	容器	是	请求付费配置的容器。 子节点：Payer
Payer	字符串	是	指定Bucket付费类型。 取值：BucketOwner、Requester 父节点：RequestPaymentConfiguration

示例

请求示例

```
PUT /?requestPayment
Content-Length: 83
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 23 Jul 2019 01:33:47 GMT
Authorization: OSS LTAIC**********:FsDgQiO+RMwLq**********
<RequestPaymentConfiguration><Payer>Requester</Payer></RequestPaymentConfiguration>
```

返回示例

```
200 (OK)
content-length: 0
x-oss-request-id: 5D3663FBB007B79097FC****
date: Tue, 23 Jul 2019 01:33:47 GMT
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	访问的Bucket不存在。

6.23 GetBucketRequestPayment

GetBucketRequestPayment接口用于获取请求者付费模式配置信息。

请求语法

```
GET /?requestPayment HTTP/1.1
Date: GMT Date
Host: BucketName.oss.aliyuncs.com
Authorization: authorization string
```

响应元素

名称	类型	描述
RequestPaymentConfiguration	容器	请求付费配置的容器。 子节点：Payer
Payer	字符串	指定Bucket付费类型。 取值：BucketOwner、Requester 父节点：RequestPaymentConfiguration

示例

请求示例

```
GET /?requestPayment
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 23 Jul 2019 01:23:20 GMT
Authorization: OSS LTAICQmy1********:Rk7tT30yhDO3D**********
```

返回示例

```
200 (OK)
content-length: 129
server: AliyunOSS
x-oss-request-id: 5D366188B007B79097EC****
date: Tue, 23 Jul 2019 01:23:20 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration>
```

```
  <Payer>BucketOwner</Payer>
</RequestPaymentConfiguration>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	访问的Bucket不存在。

7 关于Object操作

7.1 PutObject

PutObject接口用于上传文件（Object）。

说明:

- 添加的文件大小不得超过5 GB。
- 如果已经存在同名的Object，并且有访问权限，则新添加的文件将覆盖原来的文件，并成功返回200 OK。
- OSS没有文件夹的概念，所有元素都是以文件来存储，但您可以通过创建一个空的Object创建模拟文件夹，具体参见OSS文件夹。

请求语法

```
PUT /ObjectName HTTP/1.1
Content-Length: ContentLength
Content-Type: ContentType
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

请求头

说明:

OSS支持HTTP协议规定的5个请求头：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type。如果上传Object时设置了这些请求头，则该Object被下载时，相应的请求头值会被自动设置成上传时的值。

名称	类型	是否必选	描述
Authorization	字符串	否	表示请求本身已被授权，详细描述参考RFC2616。 一般情况下Authorization是必选请求头，但是若采用URL包含签名，可以不用携带该请求头。详细描述参考在URL中包含签名。 默认值：无

名称	类型	是否必选	描述
Cache-Control	字符串	否	指定该Object被下载时网页的缓存行为，详细描述参考RFC2616。 默认值：无
Content-Disposition	字符串	否	指定该Object被下载时的名称，详细描述参考RFC2616。 默认值：无
Content-Encoding	字符串	否	指定该Object被下载时的内容编码格式，详细描述参考RFC2616。 默认值：无
Content-MD5	字符串	否	用于检查消息内容是否与发送时一致。Content-MD5是一串由MD5算法生成的值。上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性。 默认值：无
Content-Length	字符串	否	用于描述HTTP消息体的传输大小。 如果请求头中的Content-Length值小于实际请求体中传输的数据大小，OSS仍将成功创建文件，但Object的大小只等于Content-Length中定义的大小，其他数据将被丢弃。
ETag	字符串	否	Object生成时会创建相应的ETag (entity tag)，ETag用于标示一个Object的内容。对于PutObject请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。不建议用户使用ETag来作为Object内容的MD5校验数据完整性。 默认值：无

名称	类型	是否必选	描述
Expires	字符串	否	过期时间，详细描述参考照RFC2616。 默认值：无
x-oss-server-side-encryption	字符串	否	指定OSS创建Object时的服务器端加密编码算法。 取值：AES256 或 KMS（您需要购买KMS套件，才可以使用 KMS 加密算法，否则会报 KmsServiceNotEnabled 错误码） 指定此参数后，在响应头中会返回此参数，OSS会对上传的Object进行加密编码存储。当下载该Object时，响应头中会包含x-oss-server-side-encryption，且该值会被设置成该Object的加密算法。
x-oss-server-side-encryption-key-id	字符串	否	KMS托管的用户主密钥。 该参数在x-oss-server-side-encryption为KMS时有效。
x-oss-object-acl	字符串	否	指定OSS创建Object时的访问权限。 合法值：public-read，private，public-read-write
x-oss-storage-class	字符串	否	指定Object的存储类型。 对于任意存储类型的Bucket，若上传Object时指定此参数，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。 取值：Standard、IA、Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。

名称	类型	是否必选	描述
x-oss-meta-*	字符串	否	使用PutObject接口时，如果配置以x-oss-meta-*为前缀的参数，则该参数视为元数据，例如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的元数据总大小不能超过8KB。元数据支持短横线（-）、数字、英文字母（a-z），英文字符的大写字母会被转成小写字母，不支持下划线（_）在内的其他字符。
x-oss-tagging	字符串	否	指定Object的标签，可同时设置多个标签，例如：TagA=A&TagB=B。  说明： Key和Value需要先进行URL编码，如果某项没有"="，则看作Value为空字符串。

示例

- **简单上传的请求示例**

```
PUT /test.txt HTTP/1.1
Host: test.oss-cn-zhangjiakou.aliyuncs.com
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
Content-Type: text/plain
date: Tue, 04 Dec 2018 15:56:37 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2P
****
Transfer-Encoding: chunked
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 04 Dec 2018 15:56:38 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5C06A3B67B8B5A3DA422299D
ETag: "D41D8CD98F00B204E9800998ECF8****"
x-oss-hash-crc64ecma: 0
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
x-oss-server-time: 7
```

- **带有归档存储类型的请求示例**

```
PUT /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com Cache-control: no-
cache
Expires: Fri, 28 Feb 2012 05:38:42 GMT
```

```
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Fri, 24 Feb 2012 06:03:28 GMT
Content-Type: image/jpg
Content-Length: 344606
x-oss-storage-class: Archive
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2P
****
[344606 bytes of object data]
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Sat, 21 Nov 2015 18:52:34 GMT
Content-Type: image/jpg
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5650BD72207FB30443962F9A
x-oss-bucket-version: 1418321259
ETag: "A797938C31D59EDD08D86188F6D5B872"
```

SDK

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- iOS
- Node.js
- Browser.js
- Ruby

常见问题

如何计算Content-MD5?

首先计算MD5加密的二进制数组（128位），然后再对这个二进制数组进行base64编码（而不是对32位字符串编码）。例如，用Python计算`0123456789`的Content-MD5，代码为：

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
```

```
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```

说明:

正确的计算方式为：`hash.digest()`，**计算出进制数组（128位）** `>>> hash.digest() 'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'`。

常见错误是直接对计算出的32位字符串编码进行base64编码。 例如：使用`hash.hexdigest()`**计算方式，计算得到可见的32位字符串编码** `>>> hash.hexdigest() '781e5e245d69b566979b86e28d23f2c7'` **。错误的MD5值进行base64编码后的结果为：** `>>> base64.b64encode(hash.hexdigest()) 'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='`。

错误码

错误码	HTTP 状态码	描述
MissingContentLength	411	请求头没有采用chunked encoding编码方式，且没有Content-Length参数。
InvalidEncryptionAlgorithmError	400	指定x-oss-server-side-encryption的值无效。有效值为AES256或KMS。
AccessDenied	403	对试图添加Object的Bucket没有访问权限。
NoSuchBucket	404	试图添加Object的Bucket不存在。
InvalidObjectName	400	传入的Object key长度大于1023字节。
InvalidArgument	400	· 添加的文件大小超过5 GB。 · x-oss-storage-class等参数的值无效。
RequestTimeout	400	指定了Content-Length，但是没有发送消息体，或者发送的消息体小于给定大小。这种情况下服务器会一直等待，直到time out。
KmsServiceNotEnabled	403	将x-oss-server-side-encryption指定为KMS，但没有预先购买KMS套件。

7.2 CopyObject

CopyObject接口用于在存储空间（Bucket） 内或同地域的Bucket之间拷贝文件（Object）。使用CopyObject接口可以发送一个Put请求给OSS，OSS会自动判断出这是一个拷贝操作，并直接在服务器端执行该操作。

使用限制

- CopyObject接口仅支持拷贝小于1 GB的Object。如果要拷贝大于1 GB的Object，您可以使用UploadPartCopy接口进行操作。
- CopyObject接口支持修改Object的元数据(源Object与目标Object一致)，最大可支持修改48.8 TB的Object。
- 使用此接口须对源Object有读权限。
- 源Object和目标Object必须属于同一地域。
- 不能拷贝通过追加上传方式产生的Object。
- 如果源Object为软链接，则只拷贝软链接，无法拷贝软链接指向的文件内容。

计量计费

- 单次使用CopyObject接口会对源Object所在的Bucket增加一次Get请求次数。
- 单次使用CopyObject接口会对目标Object所在的Bucket增加一次Put请求次数。
- 使用CopyObject接口会对目标Object所在的Bucket增加相应的存储量。
- 使用CopyObject接口更改Object存储类型涉及到数据覆盖，如果IA或Archive类型的Object分别在创建后30和60天内被覆盖，则它们会产生提前删除费用。比如，IA类型Object创建10天后，被覆盖成Archive或Standard，则会产生20天的提前删除费用。

请求语法

```
PUT /DestObjectName HTTP/1.1
Host: DestBucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
x-oss-copy-source: /SourceBucketName/SourceObjectName
```

请求头

说明:

拷贝操作涉及到的请求头都是以x-oss-开头，所以所有请求头都要加到签名字符串中。

名称	类型	是否必选	描述
x-oss-copy-source	字符串	是	指定拷贝的源地址。 默认值：无
x-oss-copy-source-if-match	字符串	否	如果源Object的ETag值和您提供的ETag相等，则执行拷贝操作，并返回200 OK；否则返回412 Precondition Failed错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-none-match	字符串	否	如果源Object的ETag值和您提供的ETag不相等，则执行拷贝操作，并返回200 OK；否则返回304 Not Modified错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-unmodified-since	字符串	否	如果指定的时间等于或者晚于文件实际修改时间，则正常拷贝文件，并返回200 OK；否则返回412 Precondition Failed错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-modified-since	字符串	否	如果源Object在用户指定的时间以后被修改过，则执行拷贝操作；否则返回304 Not Modified错误码（预处理失败）。 默认值：无

名称	类型	是否必选	描述
x-oss-metadata-directive	字符串	否	指定如何设置目标Object的元信息。取值如下： · COPY （默认值） 复制源Object的元数据到目标Object。 说明： 不会复制源Object的x-oss-server-side-encryption。目标Object是否进行服务器端加密编码只根据当前拷贝操作是否指定了x-oss-server-side-encryption来决定。 · REPLACE 忽略源Object的元数据，直接采用请求中指定的元数据。 说明： 如果拷贝操作的源Object地址和目标Object地址相同，则无论x-oss-metadata-directive为何值，都会直接替换源Object的元数据。

名称	类型	是否必选	描述
x-oss-server-side-encryption	字符串	否	指定OSS创建目标Object时，服务器端熵编码加密算法 。 取值：AES256 、KMS（您需要购买 KMS 套件，才可以使用KMS加密算法，否则会返回KmsServiceNotEnabled错误码）  说明： · 如果拷贝操作中未指定x-oss-server-side-encryption，则无论源Object是否进行过服务器端加密编码，拷贝之后的目标Object都未进行服务器端加密编码。 · 如果拷贝操作中指定了x-oss-server-side-encryption，则无论源Object是否进行过服务器端加密编码，拷贝之后的目标Object都会进行服务器端加密编码。并且拷贝操作的响应头中会包含x-oss-server-side-encryption，值为目标Object的加密算法。在目标Object被下载时，响应头中也会包含x-oss-server-side-encryption，值为该Object的加密算法。
x-oss-server-side-encryption-key-id	字符串	否	表示KMS托管的用户主密钥。 该参数在x-oss-server-side-encryption为KMS时有效。
x-oss-object-acl	字符串	否	指定OSS创建目标Object时的访问权限。 取值：public-read、private、public-read-write、default

名称	类型	是否必选	描述
x-oss-storage-class	字符串	否	指定Object的存储类型。 取值：Standard、IA、Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject  说明: · IA与Archive类型的单个文件如不足64 KB，会按64 KB 计量计费。建议您在使用CopyObject接口时不要将Object的存储类型指定为IA或Archive。 · 对于任意存储类型的Bucket，若上传Object时指定此参数，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。 · 更改Object存储类型涉及到数据覆盖，如果IA或Archive类型Object分别在创建后30和60天内被覆盖，则它们会产生提前删除费用。
x-oss-tagging	字符串	否	指定Object的对象标签，可同时设置多个标签，例如： TagA=A&TagB=B。  说明: Key和Value需要先进行URL编码，如果某项没有"="，则看作Value为空字符串。
x-oss-tagging-directive	字符串	否	指定如何设置目标Object的对象标签。取值如下： · Copy（默认值）：复制源Object的对象标签到目标 Object。 · Replace：忽略源Object的对象标签，直接采用请求中 指定的对象标签。

响应元素

名称	类型	描述
CopyObjectResult	字符串	CopyObject的结果。 默认值：无
ETag	字符串	目标Object的ETag值。 父元素：CopyObjectResult
LastModified	字符串	目标Object最后更新时间。 父元素：CopyObjectResult

示例

- 请求示例1

```
PUT /copy_oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 07:18:48 GMT
x-oss-storage-class: Archive
x-oss-copy-source: /oss-example/oss.jpg
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:gmnwPKuu20LQEjd+iPkL259A
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Content-Type: application/xml
Content-Length: 193
Connection: keep-alive
Date: Fri, 24 Feb 2012 07:18:48 GMT
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
 <LastModified>Fri, 24 Feb 2012 07:18:48 GMT</LastModified>
 <ETag>"5B3C1A2E053D763E1B002CC607C5****"</ETag>
</CopyObjectResult>
```

- 请求示例2

```
PUT /test%2FAK.txt HTTP/1.1
Host: tesx.oss-cn-zhangjiakou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
x-oss-copy-source: /test/AK.txt
date: Fri, 28 Dec 2018 09:41:55 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:gmnwPKuu20LQEjd+iPkL259A
****
```

```
Content-Length: 0
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 28 Dec 2018 09:41:56 GMT
Content-Type: application/xml
Content-Length: 184
Connection: keep-alive
x-oss-request-id: 5C25EFE4462CE00EC6D87156
ETag: "F2064A169EE92E9775EE5324D0B1****"
x-oss-hash-crc64ecma: 12753002859196105360
x-oss-server-time: 150
<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
  <ETag>"F2064A169EE92E9775EE5324D0B1****"</ETag>
  <LastModified>2018-12-28T09:41:56.000Z</LastModified>
</CopyObjectResult>
```

说明:

x-oss-hash-crc64ecma表明Object的64位CRC值。该64位CRC值根据ECMA-182标准计算得出。进行CopyObject操作时，生成的Object不保证具有此值。

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android
- iOS
- Node.js
- Ruby

错误码

错误码	HTTP 状态码	描述
InvalidArgument	400	x-oss-storage-class等参数的值无效。

错误码	HTTP 状态码	描述
Precondition Failed	412	· 指定了x-oss-copy-source-if-match请求头，但源Object的ETag值和您提供的ETag不相等。 · 指定了x-oss-copy-source-if-unmodified-since请求头，且指定的时间早于文件实际修改时间。
Not Modified	304	· 指定了x-oss-copy-source-if-none-match请求头，且源Object的ETag值和您提供的ETag相等。 · 指定了x-oss-copy-source-if-modified-since请求头，但源Object在指定的时间后没被修改过。
KmsService NotEnabled	403	将x-oss-server-side-encryption指定为KMS，但没有预先购买KMS套件。

7.3 GetObject

GetObject接口用于获取某个文件（Object）。此操作需要对该Object有读权限。

> 说明:
> 如果Object类型为归档类型，需要先完成RestoreObject请求且该请求不能超时。

请求语法

```
GET /ObjectName HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
Range: bytes=ByteRange(可选)
```

请求头

> 说明:
> · OSS支持在GET请求中通过请求头来自定义响应头，但只有请求成功（即返回码为 200 OK ）才会将响应头的值设置成GET请求头中指定的值。
> · OSS不支持在匿名访问的GET请求中自定义响应头。
> · 用户发送的GET请求必须携带签名。

名称	类型	是否必选	描述
response-content-type	字符串	否	指定OSS返回请求的content-type头。 默认值：无
response-content-language	字符串	否	指定OSS返回请求的content-language头。 默认值：无
response-expires	字符串	否	指定OSS返回请求的expires头。 默认值：无
response-cache-control	字符串	否	指定OSS返回请求的cache-control头。 默认值：无
response-content-disposition	字符串	否	指定OSS返回请求的content-disposition头。 默认值：无
response-content-encoding	字符串	否	指定OSS返回请求的content-encoding头。 默认值：无
Range	字符串	否	指定文件传输的范围。 默认值：无 · 如果指定的范围符合规范，返回消息中会包含整个Object的大小和此次返回的范围。例如：Content-Range: bytes 0-9/44，表示整个Object大小为44，此次返回的范围为0-9。 · 如果指定的范围不符合范围规范，则传送整个Object，并且不在结果中提及Content-Range。

名称	类型	是否必选	描述
If-Modified-Since	字符串	否	如果指定的时间早于实际修改时间或指定的时间不符合规范，会直接返回Object，并返回200 OK；否则返回304 Not Modified。 默认值：无 时间格式：GMT，例如`Fri, 13 Nov 2015 14:47:53 GMT`
If-Unmodified-Since	字符串	否	如果指定的时间等于或者晚于Object实际修改时间，则正常传输Object，并返回200 OK；否则返回412 Precondition Failed。 默认值：无 时间格式：GMT，例如`Fri, 13 Nov 2015 14:47:53 GMT` If-Modified-Since和If-Unmodified-Since可以同时使用。
If-Match	字符串	否	如果传入期望的ETag和Object的ETag匹配，则正常传输Object，并返回200 OK；否则返回412 Precondition Failed。 默认值：无
If-None-Match	字符串	否	如果传入的ETag值和Object的ETag不匹配，则正常传输Object，并返回200 OK；否则返回304 Not Modified。 默认值：无 If-Match和If-None-Match可以同时使用。

名称	类型	是否必选	描述
Accept-Encoding	字符串	否	指定客户端的编码类型。 如果需要对返回内容进行GZIP压缩传输，您需要在请求头中以显示方式加入Accept-Encoding:gzip。OSS会根据Object的Content-Type和Object大小（不小于1 KB）判断是否返回经过GZIP压缩的数据。 说明: · 如果采用了GZIP压缩则不会附带ETag信息。 · 目前OSS支持GZIP压缩的Content-Type为HTML、Javascript、CSS、XML、RSS、JSON。

响应头

说明:
如果Object类型为符号链接，会返回目标Object的内容。响应头中Content-Length、ETag、Content-Md5均为目标Object的元数据；Last-Modified取目标Object和符号链接对应的最大值（即在两者中取较晚的时间）；其他均为符号链接的元数据。

名称	类型	描述
x-oss-server-side-encryption	字符串	若Object在服务器端采用熵编码加密存储，使用GET请求时，系统会自动解密返回给用户，并且在响应头中返回x-oss-server-side-encryption，表明该Object的服务器端加密算法。
x-oss-tagging-count	字符串	对象关联的标签的个数。仅当用户有读取标签权限时返回

示例

· 简单的GET请求示例

```
GET /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 06:38:30 GMT
```

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:UNQDb7GapEgJkcde6OhZ9J
*****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 3a8f-2e2d-7965-3ff9-51c875b*****
x-oss-object-type: Normal
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E0563E1B002CC607C*****"
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
[344606 bytes of object data]
```

- **带有Range参数的请求示例**

```
GET //oss.jpg HTTP/1.1
Host:oss-example. oss-cn-hangzhou.aliyuncs.com
Date: Fri, 28 Feb 2012 05:38:42 GMT
Range: bytes=100-900
Authorization: OSS qn6qrrqxo2oawuk5jbyc:qZzjF3DUtd+yK16BdhGtFcC*****
```

返回示例

```
HTTP/1.1 206 Partial Content
x-oss-request-id: 28f6-15ea-8224-234e-c0ce407*****
x-oss-object-type: Normal
Date: Fri, 28 Feb 2012 05:38:42 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E05E1B002CC607C*****"
Accept-Ranges: bytes
Content-Range: bytes 100-900/344606
Content-Type: image/jpg
Content-Length: 801
Server: AliyunOSS
[801 bytes of object data]
```

- **带自定义返回消息头的请求示例**

```
GET /oss.jpg?response-expires=Thu%2C%2001%20Feb%202012%2017%3A00
%3A00%20GMT& response-content-type=text&response-cache-control=No
-cache&response-content-disposition=attachment%253B%2520filename%
253Dtesting.txt&response-content-encoding=utf-8&response-content-
language=%E4%B8%AD%E6%96%87 HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com:
Date: Fri, 24 Feb 2012 06:09:48 GMT
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC75A644*****
x-oss-object-type: Normal
Date: Fri, 24 Feb 2012 06:09:48 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D1B002CC607*****"
Content-Length: 344606
Connection: keep-alive
Content-disposition: attachment; filename:testing.txt
```

```
Content-language: 中文
Content-encoding: utf-8
Content-type: text
Cache-control: no-cache
Expires: Fri, 24 Feb 2012 17:00:00 GMT
Server: AliyunOSS
[344606 bytes of object data]
```

- **Object类型为符号链接的请求示例**

```
GET /link-to-oss.jpg HTTP/1.1
Accept-Encoding: identity
Date: Tue, 08 Nov 2016 03:17:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS qn6qrrqxok53otfjbyc:qZzjF3DUtd+yK16BdhGtFc*****
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 08 Nov 2016 03:17:58 GMT
Content-Type: application/octet-stream
Content-Length: 20
Connection: keep-alive
x-oss-request-id: 582143E6A212AD*****
Accept-Ranges: bytes
ETag: "8086265EFC021F9A2F09BF4****"
Last-Modified: Tue, 08 Nov 2016 03:17:58 GMT
x-oss-object-type: Symlink
Content-MD5: gIYmXvwCEe0fmi8Jv0Y****
```

- **Restore操作已经完成的请求示例**

```
GET /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 09:38:30 GMT
Authorization: OSS qn6qrrqxo2o***k53otfjbyc:zUglwRPGkbByZxm1+y4eyu
+*****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 58F723829F29F18D7F00*****
x-oss-object-type: Normal
x-oss-restore: ongoing-request="false", expiry-date="Sun, 16 Apr
2017 08:12:33 GMT"
Date: Sat, 15 Apr 2017 09:38:30 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "5B3C1A2E0763E1B002CC607C*****"
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
[354606 bytes of object data]
```

SDK

GetObject接口所对应的各语言SDK如下：

- Java

- Python
- PHP
- Go
- C
- .NET
- Android
- Node.js
- Browser.js
- Ruby

错误码

错误码	HTTP状态码	说明
NoSuchKey	404	目标Object不存在。
SymlinkTargetNotExist	404	Object类型为符号链接，且目标Object不存在。
InvalidTargetType	400	Object类型为符号链接，且目标Object类型仍为符号链接。
InvalidObjectState	403	下载归档类型的Object时， · 没有提交RestoreObject请求或者上一次提交RestoreObject已经超时。 · 已经提交RestoreObject请求，但数据的RestoreObject操作还没有完成。
Not Modified	304	· 指定了If-Modified-Since请求头，但源Object在指定的时间后没被修改过。 · 指定了If-None-Match请求头，且源Object的ETag值和您提供的ETag相等。
Precondition Failed	412	· 指定了If-Unmodified-Since，但指定的时间早于Object实际修改时间 。 · 指定了If-Match，但源Object的ETag值和您提供的ETag不相等。

7.4 AppendObject

AppendObject接口用于以追加写的方式上传文件（Object）。通过AppendObject操作创建的Object类型为Appendable Object，而通过PutObject上传的Object是Normal Object。

使用限制

- AppendObject产生的Object大小不得超过5 GB。
- 处于WORM保护期的Object不支持AppendObject操作。
- AppendableObject不支持指定CMK ID进行服务端KMS加密。

请求语法

```
POST /ObjectName?append&position=Position HTTP/1.1
Content-Length: ContentLength
Content-Type: ContentType
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

请求头

说明:

append和position均为CanonicalizedResource，需要包含在签名中。

名称	类型	是否必选	描述
Append	字符串	是	用于指定这是一个AppendObject操作。

名称	类型	是否必选	描述
Position	字符串	是	用于指定从何处进行追加。 首次追加操作的position必须为0，后续追加操作的position是Object的当前大小。例如，第一次AppendObject请求指定position值为0，content-length是65536，则第二次AppendObject需要指定position为65536。 每次操作成功后，响应消息头x-oss-next-append-position也会标明下一次追加的position。 说明: · 当position值为0时，如果没有同名Object存在，那么AppendObject可以和PutObject请求一样，设置x-oss-server-side-encryption等请求头。如果加入了正确的x-oss-server-side-encryption头，那么后续的AppendObject响应头部也会包含x-oss-server-side-encryption头。后续如需要更改元数据，可以使用CopyObject接口。 · 每次执行AppendObject都会更新该Object的最后修改时间。 · 在position值正确的情况下，对已存在的Appendable Object追加一个大小为0的内容，该操作不会改变Object的状态。
Cache-Control	字符串	否	指定该Object的网页缓存行为。详情参考RFC2616。 默认值：无
Content-Disposition	字符串	否	指定该Object被下载时的名称。详情参考RFC2616。 默认值：无
Content-Encoding	字符串	否	指定该Object的内容编码格式。详情参考RFC2616。 默认值：无

名称	类型	是否必选	描述
Content-MD5	字符串	否	Content-MD5是一串由MD5算法生成的值，该请求头用于检查消息内容是否与发送时一致。 获取Content-MD5值：对消息内容（不包括头部）执行MD5算法获得128比特位数字，然后对该数字进行base64编码。 默认值：无 限制：无
Expires	GMT	否	过期时间。详情参考RFC2616。 默认值：无
x-oss-server-side-encryption	字符串	否	指定服务器端加密编码算法。 合法值：AES256 或 KMS 说明: 您需要购买KMS套件，才可以使用KMS加密算法，否则会返回KmsServiceNotEnabled。
x-oss-object-acl	字符串	否	指定Object的访问权限。 合法值：public-read、private、public-read-write

名称	类型	是否必选	描述
x-oss-storage-class	字符串	否	指定Object的存储类型。 取值：Standard、IA、Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。 说明： · 对于任意存储类型的Bucket，若上传Object时指定此参数，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。 · 该值仅在首次执行AppendObject操作时有效，后续追加时不会生效。

说明：

· 使用AppendObject接口时，如果配置以x-oss-meta-*为前缀的参数，则该参数视为元数据，例如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的元数据总大小不能超过8KB。元数据支持短横线（-）、数字、英文字母（a-z），英文字符的大写字母会被转成小写字母，不支持下划线（_）在内的其他字符。
· 创建AppendObject时可以添加x-oss-meta-*，继续追加时不可以携带此参数。

响应头

响应消息头	类型	描述
x-oss-next-append-position	64位整型	下一次请求应当提供的position，即当前Object大小。 当AppendObject执行成功，或是因position和Object大小不匹配而引起的409错误时，会返回此消息头。
x-oss-hash-crc64ecma	64位整型	表明Object的64位CRC值。该64位CRC由ECMA-182标准计算得出。

CRC64的计算方式

Appendable Object的CRC采用ECMA-182标准。您可以使用以下方法进行计算：

用Boost CRC模块的方式计算

```
typedef boost::crc_optimal<64, 0x42F0E1EBA9EA3693ULL, 0xffffffff
ffffffffULL, 0xffffffffffffffffULL, true, true> boost_ecma;
uint64_t do_boost_crc(const char* buffer, int length)
{
    boost_ecma crc;
    crc.process_bytes(buffer, length);
    return crc.checksum();
}
```

用Python crcmod的方式计算

```
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)
print do_crc64(“123456789”)
```

示例

- **请求示例**

```
POST /oss.jpg?append&position=0 HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Cache-control: no-cache
Expires: Wed, 08 Jul 2015 16:57:01 GMT
Content-Encoding: utf-8
x-oss-storage-class: Archive
Content-Disposition: attachment;filename=oss_download.jpg
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1717
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2P
****
[1717 bytes of object data]
```

返回示例

```
HTTP/1.1 200 OK
Date: Wed, 08 Jul 2015 06:57:01 GMT
ETag: "0F7230CAA4BE94CCBDC99C550000****"
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
x-oss-hash-crc64ecma: 14741617095266562575
x-oss-next-append-position: 1717
x-oss-request-id: 559CC9BDC755F95A64485981
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go

- C
- .NET
- Android
- iOS
- Ruby

和其他操作的关系

其他操作	关系描述
PutObject	如果对一个已经存在的Appendable Object进行PutObject操作，该Appendable Object会被新的Object覆盖，类型变为Normal Object。
HeadObject	对Appendable Object执行HeadObject操作会返回x-oss-next-append-position、x-oss-hash-crc64ecma以及x-oss-object-type。Appendable Object的x-oss-object-type为Appendable。
GetBucket	GetBucket请求的响应中，会把Appendable Object的Type设为Appendable。
CopyObject	不能使用CopyObject来拷贝一个Appendable Object，也不能使用CopyObject改变Appendable Object的服务器端加密属性。可以使用CopyObject来改变自定义的元信息。

错误码

错误代码	HTTP 状态码	说明
ObjectNotAppendable	409	对一个非Appendable Object进行AppendObject操作。
PositionNotEqualToLength	409	· position的值和当前Object的长度不一致。 说明: 您可以通过响应头x-oss-next-append-position得到下一次position，并再次进行请求。由于并发的关系，即使把position的值设为了x-oss-next-append-position，该请求依然可能因为PositionNotEqualToLength而失败。 · position值为0时，如果没有同名Appendable Object，或者同名Appendable Object长度为0，该请求成功；其他情况均视为position和Object长度不匹配的情形，返回此错误码。
InvalidArgument	400	x-oss-storage-class等值无效。

7.5 DeleteObject

DeleteObject用于删除某个文件（Object）。使用DeleteObject需要对该Object有写权限。

说明:

如果Object类型为软链接，使用DeleteObject仅会删除该软链接。

请求语法

```
DELETE /ObjectName HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

- **请求示例**

```
DELETE /AK.txt HTTP/1.1
Host: test.oss-cn-zhangjiakou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
date: Wed, 02 Jan 2019 13:28:38 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+
NIUs=zV0****
Content-Length: 0
```

返回示例

```
HTTP/1.1 204 No Content
Server: AliyunOSS
Date: Wed, 02 Jan 2019 13:28:38 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5C2CBC8653718B5511EF4535
x-oss-server-time: 134
```

SDK

DeleteObject接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET

- Android
- iOS
- Node.js
- Browser.js
- Ruby

错误码

错误码	HTTP 状态码	描述
Not Found	404	目标Bucket不存在。
No Content	204	目标Object不存在。

7.6 DeleteMultipleObjects

DeleteMultipleObjects接口用于删除同一个存储空间（Bucket）中的多个文件（Object）。

说明:

DeleteMultipleObjects接口一次最多允许删除1000个文件。

请求语法

```
POST /?delete HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Content-Length: ContentLength
Content-MD5: MD5Value
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>key</Key>
  </Object>
…
</Delete>
```

请求头

OSS会根据以下请求头验证收到的消息体，消息体正确才会执行删除操作。

名称	类型	是否必选	描述
Encoding-type	字符串	否	Key使用UTF-8字符。如果Key中包含XML 1.0标准不支持的控制字符，您可以通过指定encoding-type对返回结果中的Key进行编码。 默认值：无 可选值：url
Content-Length	字符串	是	用于描述HTTP消息体的传输长度。 OSS会根据此请求头验证收到的消息体，消息体正确才会执行删除操作。
Content-MD5	字符串	是	Content-MD5是一串由MD5算法生成的值，该请求头用于检查消息内容是否与发送时一致。上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性。 说明: 将DeleteMultipleObjects的请求消息体经过MD5加密后得到一个128位字节数组。然后将该字节数组用base64算法编码，编码后得到的字符串即Content-MD5字段内容。

请求元素

名称	类型	是否必选	描述
Delete	容器	是	保存DeleteMultipleObjects请求的容器。 子节点：一个或多个Object元素，Quiet元素 父节点： None
Object	容器	是	保存一个Object信息的容器。 子节点：Key 父节点：Delete

名称	类型	是否必选	描述
Key	字符串	是	被删除Object的名字。 父节点：Object
Quiet	枚举字符串	是	打开简单响应模式的开关。 DeleteMultipleObjects提供以下两种消息返回模式： · 简单模式(quiet)：OSS返回的消息体中只包含删除过程中出错的Object结果。如果所有删除都成功，则没有消息体。 · 详细模式(verbose)：OSS返回的消息体中会包含所有删除Object的结果。默认采用详细模式。 有效值：true（开启简单模式）、false（开启详细模式） 默认值：false 父节点：Delete

响应元素

名称	类型	描述
Deleted	容器	保存被成功删除的Object的容器。 子节点：Key 父节点：DeleteResult
DeleteResult	容器	保存DeleteMultipleObjects请求结果的容器。 子节点：Deleted 父节点：None

名称	类型	描述
Key	字符串	被删除Object的名字。 父节点：Deleted
EncodingType	字符串	返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，则返回的结果会对Key进行编码。 父节点：容器

示例

- **关闭简单响应模式的请求示例**

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:+z3gBfnFAxBcBDgx27Y/jEfb
****
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>false</Quiet>
  <Object>
    <Key>multipart.data</Key>
  </Object>
  <Object>
    <Key>test.jpg</Key>
  </Object>
  <Object>
    <Key>demo.jpg</Key>
  </Object>
</Delete>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 78320852-7eee-b697-75e1-b6db0f4849e7
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length: 244
Content-Type: application/xml
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <Deleted>
       <Key>multipart.data</Key>
    </Deleted>
    <Deleted>
       <Key>test.jpg</Key>
    </Deleted>
    <Deleted>
       <Key>demo.jpg</Key>
```

```
    </Deleted>
</DeleteResult>
```

- **打开简单响应模式的请求示例**

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:WuV0Jks8RyGSNQrBca64kEEx
****
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>multipart.data</Key>
  </Object>
  <Object>
    <Key>test.jpg</Key>
  </Object>
  <Object>
    <Key>demo.jpg</Key>
  </Object>
</Delete>
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

- Java
- Python
- PHP
- Go
- C
- .NET
- iOS
- Node.js
- Browser.js
- Ruby

错误码

错误码	HTTP 状态码	描述
InvalidDigest	400	上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性，如果不一致会返回此错误码。
MalformedXML	400	· 消息体最大允许2 MB的内容，超过2 MB会返回此错误码。 · DeleteMultipleObjects接口最多支持单次请求删除1000个Object。单次请求超过1000个Object返回此错误码。

7.7 HeadObject

HeadObject接口用于获取某个文件（Object）的元信息。使用此接口不会返回文件内容。

说明:

如果您在PutObject的时候传入以x-oss-meta-为开头的user meta，比如x-oss-meta-location，返回消息时，返回这些user meta。

请求语法

```
HEAD /ObjectName HTTP/1.1
Host: BucketName/oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

请求头

名称	类型	是否必选	描述
If-Modified-Since	字符串	否	如果传入参数中的时间早于实际修改时间，则返回200 OK和Object Meta；否则返回304 not modified。 默认值：无
If-Unmodified-Since	字符串	否	如果传入参数中的时间等于或者晚于文件实际修改时间，则返回200 OK和Object Meta；否则返回412 precondition failed。 默认值：无

名称	类型	是否必选	描述
If-Match	字符串	否	如果传入期望的ETag和Object的 ETag匹配，则返回200 OK和Object Meta；否则返回412 precondition failed。 默认值：无
If-None-Match	字符串	否	如果传入期望的ETag值和Object的ETag不匹配，则返回200 OK和Object Meta；否则返回304 Not Modified。 默认值：无

响应头

说明:

如果文件类型为软链接， 响应头中Content-Length、ETag、Content-Md5 均为目标文件的元信息；Last-Modified是目标文件和软链接的最大值；其他均为软链接元信息。

名称	类型	描述
x-oss-meta-*	字符串	以x-oss-meta-为前缀的参数作为用户自定义meta header。当用户在PutObject时设置了以x-oss-meta-为前缀的自定义meta，则响应中会包含这些自定义meta。
非x-oss-meta-开头的自定义header	字符串	当用户在PutObject时，自定义一些非x-oss-meta为前缀的Header，如x-oss-persistent-headers:key1:base64_encode(value1),key2:base64_encode(valu，响应中会增加相应的自定义Header。 详情请参见OSS如何添加非x-oss-meta-开头的自定义。
x-oss-server-side-encryption	字符串	若该Object为进行服务器端熵编码加密存储的，则在响应头头中会返回此参数，其值表明该Object的服务器端加密算法。
x-oss-server-side-encryption-key-id	字符串	如果用户在创建Object时使用了服务端加密，且加密方法为KMS，则响应中会包含此Header，表示加密所使用的用户KMS key ID。

名称	类型	描述
x-oss-storage-class	字符串	表示Object的存储类型，分别为：标准存储类型（Standard）、低频访问存储类型（Infrequent Access）、归档存储类型（Archive）。 · 标准存储类型提供高可靠、高可用、高性能的对象存储服务，能够支持频繁的数据访问。 · 低频访问存储类型适合需要长期存储但不经常被访问的数据（平均每月访问频率1到2次）。 · 归档存储类型适合需要长期存储（建议半年以上）的归档数据，在存储周期内极少被访问，数据进入到可读取状态需要1分钟的解冻时间。
x-oss-object-type	字符串	表示Object的类型。 · 通过PutObject上传的Object类型为Normal。 · 通过AppendObject上传的Object类型为Appendable · 通过MultipartUpload上传的Object类型为Multipart。
x-oss-next-append-position	字符串	对于Appendable类型的Object会返回此Header，指明下一次请求应当提供的position。
x-oss-hash-crc64ecma	字符串	表示该Object的64位CRC值。该64位CRC根据ECMA-182标准计算得出。请注意，有些较老的Object可能没有这个Header。
x-oss-expiration	字符串	如果用户为该Object设置了生命周期规则（Lifecycle），响应中将包含x-oss-expiration header。其中expiry-date的值表示该Object的过期日期，rule-id的值表示相匹配的规则ID。
x-oss-restore	字符串	如果Bucket类型为Archive，且用户已经提交Restore请求，则响应头中会以x-oss-restore返回该Object的Restore状态，分如下几种情况： · 如果没有提交Restore或者Restore已经超时，则不返回该字段。 · 如果已经提交Restore，且Restore没有完成，则返回的x-oss-restore值为: ongoing-request=”true”。 · 如果已经提交Restore，且Restore已经完成，则返回的x-oss-restore值为: ongoing-request=”false”, expiry-date=”Sun, 16 Apr 2017 08:12:33 GMT”，其中expiry-date是Restore完成后Object进入可读状态的过期时间。
x-oss-process-status	字符串	当用户通过MNS消息服务创建OSS事件通知后，在进行请求OSS相关操作时如果有匹配的事件通知规则，则响应中会携带这个Header，值为经过Base64编码json格式的事件通知结果。

名称	类型	描述
x-oss-request-charged	字符串	当Object所属的Bucket被设置为请求者付费模式，且请求者不是Bucket的拥有者时，响应中将携带此Header，值为requester。
Content-Md5	字符串	对于Normal类型的Object，根据RFC 1864标准对消息内容（不包括Header）计算Md5值获得128比特位数字，对该数字进行base64编码作为一个消息的Content-Md5值。Multipart和Appendable类型的文件不会返回这个Header。
Last-Modified	字符串	Object最后一次修改的日期，格式为HTTP 1.1协议中规定的GMT时间。
Access-Control-Allow-Origin	字符串	当Object所在的Bucket配置了CORS规则，如果请求的Origin满足指定的CORS规则时会在响应中包含这个Origin。
Access-Control-Allow-Methods	字符串	当Object所在的Bucket配置了CORS规则，如果请求的Access-Control-Request-Method满足指定的CORS规则时会在响应中包含允许的Methods。
Access-Control-Max-Age	字符串	当Object所在的Bucket配置了CORS规则，如果请求满足Bucket配置的CORS规则时会在响应中包含MaxAgeSeconds。
Access-Control-Allow-Headers	字符串	当Object所在的Bucket配置了CORS规则，如果请求满足指定的CORS规则时会在响应中包含这些Headers。
Access-Control-Expose-Headers	字符串	表示允许访问客户端JavaScript程序的headers列表。当Object所在的Bucket配置了CORS规则，如果请求满足指定的CORS规则时会在响应中包含ExposeHeader。
x-oss-tagging-count	字符串	对象关联的标签的个数。仅当用户有读取标签权限时返回。

示例

- **正常请求示例**

```
HEAD /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
```

```
Date: Fri, 24 Feb 2012 07:32:52 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:JbzF2LxZUtanlJ5dLA092wpD
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
x-oss-object-type: Normal
x-oss-storage-class: Archive
Date: Fri, 24 Feb 2012 07:32:52 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a3986****"
Content-Length: 344606
Content-Type: image/jpg
Connection: keep-alive
Server: AliyunOSS
```

- **提交Restore请求但Restore没有完成时的请求示例**

```
HEAD /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 07:32:52 GMT
Authorization: OSS e1Unnbm1rgdnpI:KKxkdNrUBu2t1kqlDh0MLbDb****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 58F71A164529F18D7F000045
x-oss-object-type: Normal
x-oss-storage-class: Archive
x-oss-restore: ongoing-request="true"
Date: Sat, 15 Apr 2017 07:32:52 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a3986****"
Content-Length: 344606
Content-Type: image/jpg
Connection: keep-alive
Server: AliyunOSS
```

- **提交Restore请求且Restore已经完成时的请求示例**

```
HEAD /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 09:35:51 GMT
Authorization: OSS e1Unnbm1rgdnpI:21qtGJ+ykDVmdu6O6FMJnn+W****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 58F725344529F18D7F000055
x-oss-object-type: Normal
x-oss-storage-class: Archive
x-oss-restore: ongoing-request="false", expiry-date="Sun, 16 Apr
2017 08:12:33 GMT"
Date: Sat, 15 Apr 2017 09:35:51 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a3986****"
```

```
Content-Length: 344606
```

错误码

错误码	HTTP 状态码	描述
NoSuchKey	404	文件不存在。
SymlinkTargetNotExist	404	文件类型为软链接。
InvalidTargetType	400	文件类型为软链接，且目标文件类型是软链接。
Not Modified	304	· 指定了If-Modified-Since请求头，但源Object在指定的时间后没被修改过。 · 指定了If-None-Match请求头，且源Object的ETag值和您提供的ETag相等。
Precondition Failed	412	· 指定了If-Unmodified-Since，但指定的时间早于Object实际修改时间 。 · 指定了If-Match，但源Object的ETag值和您提供的ETag不相等。

7.8 GetObjectMeta

GetObjectMeta接口用于获取一个文件（Object）的元数据信息，包括该Object的ETag、Size、LastModified信息，并且不返回该Object的内容。

说明:

· 如果Object类型为符号链接，会返回符号链接自身信息。

· 无论正常返回还是非正常返回，GetObjectMeta均不返回消息体。

请求语法

```
HEAD /ObjectName?objectMeta HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应头

响应头	类型	描述
Content-Length	字符串	Object的文件大小。

响应头	类型	描述
ETag	字符串	Object生成时会创建ETag (entity tag)，ETag用于标示一个Object的内容。 对于PutObject请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。不建议用户使用ETag来作为Object内容的MD5校验数据完整性。 默认值：无

示例

- 请求示例

```
HEAD /oss.jpg?objectMeta HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0Fmgb
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5****"
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 344606
Connection: keep-alive
Server: AliyunOSS
```

SDK

GetObjectMeta接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Android

- iOS

错误码

错误码	HTTP 状态码	描述
Not Found	404	目标Object不存在。

7.9 PutObjectACL

PutObjectACL接口用于修改文件（Object）的访问权限（ACL）。此操作只有Bucket Owner有权限执行，且需对Object有读写权限。

说明:

- Object ACL优先级高于Bucket ACL。例如Bucket ACL是private的，而Object ACL是public-read-write的，则所有用户都拥有这个Object的访问权限，即使这个Bucket是private bucket。如果某个Object从来没设置过ACL，则访问权限遵循Bucket ACL。
- Object的读操作包括：GetObject、HeadObject、CopyObject和UploadPartCopy中的对源Object的读；Object的写操作包括：PutObject、PostObject、AppendObject、DeleteObject、DeleteMultipleObjects、CompleteMultipartUpload以及CopyObject对新Object的写。
- 您还可以在Object的写操作时，在请求头中带上x-oss-object-acl来设置Object ACL，效果与PutObjectA ACL等同。例如PutObject时在请求头中带上x-oss-object-acl可以在上传一个Object的同时设置此Object的ACL。

ACL说明

PutObjectACL接口通过Put请求中的`x-oss-object-acl`头来设置。目前Object有以下四种访问权限。

名称	描述
`private`	Object是私有资源。只有该Object的Owner拥有该Object的读写权限，其他用户没有权限操作该Object。
`public-read`	Object是公共读资源。Object Owner拥有该Object的读写权限。非Object Owner只有该Object的读权限。
`public-read-write`	Object是公共读写资源。所有用户拥有对该Object的读写权限。
`default`	Object遵循其所在Bucket的读写权限，即Bucket是什么权限，Object就是什么权限。

请求语法

```
PUT /ObjectName?acl HTTP/1.1
x-oss-object-acl: Permission
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
PUT /test-object?acl HTTP/1.1
x-oss-object-acl: public-read
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZ
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET
- Node.js
- Browser.js
- Ruby

错误码

错误码	HTTP 状态码	描述
AccessDenied	403	接口使用者不是Bucket Owner，或Bucket Owner对Object没有读写权限。

错误码	HTTP 状态码	描述
InvalidArgument	400	指定的x-oss-object-acl值无效。

7.10 GetObjectACL

GetObjectACL接口用来获取某个存储空间（Bucket）下的某个文件（Object）的访问权限（ACL）。

说明:

如果一个Object从未设置过ACL，则调用GetObjectACL时，返回的ObjectACL为default，表示该Object的ACL遵循Bucket ACL。即如果Bucket的访问权限是private，则该Object的访问权限也是private。

请求语法

```
GET /ObjectName?acl HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
AccessControlList	容器	存储ACL信息的容器 父节点：AccessControlPolicy
AccessControlPolicy	容器	保存Get Object ACL结果的容器 父节点：None
DisplayName	字符串	Bucket拥有者的名称(目前和用户ID一致) 父节点：AccessControlPolicy.Owner
Grant	枚举字符串	Object的ACL权限 有效值：private，public-read，public-read-write 父节点：AccessControlPolicy.AccessControlList

名称	类型	描述
ID	字符串	Bucket拥有者的用户ID 父节点：AccessControlPolicy.Owner
Owner	容器	保存Bucket拥有者信息的容器 父节点：AccessControlPolicy

示例

- 请求示例

```
GET /test-object?acl HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0Fmgb
****
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 253
Content-Tupe: application/xml
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" ?>
<AccessControlPolicy>
    <Owner>
        <ID>00220120222</ID>
        <DisplayName>00220120222</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>public-read </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- .NET

错误码

错误码	HTTP 状态码	错误提示	描述
AccessDenied	403	You do not have read acl permission on this object.	没有操作权限。只有Bucket的拥有者才能使用GetObjectACL这个接口来获取该Bucket下某个Object的ACL。

7.11 PostObject

PostObject使用HTML表单上传Object到指定Bucket。

PostObject

- 请求语法

```
POST / HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
User-Agent: browser_data
Content-Length: ContentLength
Content-Type: multipart/form-data; boundary=9431149156168
--9431149156168
Content-Disposition: form-data; name="key"
key
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"
success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
attachment;filename=oss_download.jpg
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-uuid"
myuuid
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-tag"
mytag
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
access-key-id
--9431149156168
Content-Disposition: form-data; name="policy"
encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"
signature
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.
jpg"
Content-Type: image/jpeg
file_content
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OSS
```

```
--9431149156168--
```

- 请求Header

> 说明:
> PostObject的消息实体通过多重表单格式（multipart/form-data）编码，在PutObject操作中参数通过HTTP请求头传递，在PostObject操作中则作为消息体中的表单域传递。

名称	类型	描述	必须
OSSAccessKeyId	字符串	Bucket 拥有者的AccessKeyId。 默认值：无 限制：当Bucket为非public-read-write或者提供了policy（或Signature）表单域时，必须提供OSSAccessKeyId表单域。	有条件
policy	字符串	Policy规定了请求表单域的合法性。不包含policy表单域的请求被认为是匿名请求，并只能访问public-read-write的Bucket。 默认值：无 限制：当Bucket为非public-read-write或者提供了OSSAccessKeyId（或Signature）表单域时，必须提供policy表单域。	有条件
Signature	字符串	根据AccessKeySecret和policy计算的签名信息，OSS验证该签名信息从而验证该Post请求的合法性。更详细描述请参考下文 Post Signature。 默认值：无 限制：当Bucket为非public-read-write或者提供了OSSAccessKeyId（或policy）表单域时，必须提供Signature表单域。	有条件

名称	类型	描述	必须
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	字符串	HTTP请求Header，更详细信息请参见PutObject。 默认值：无	可选
file	字符串	文件或文本内容，必须是表单中的最后一个域。浏览器会自动根据文件类型来设置Content-Type，并覆盖用户的设置。 OSS一次只能上传一个文件。 默认值：无	必须
key	字符串	上传Object的名称。 如果名称包含路径，如a/b/c/b.jpg，则OSS会自动创建相应的文件夹。 默认值：无	必须
success_action_redirect	字符串	上传成功后客户端跳转到的URL。如果未指定该表单域，返回结果由success_action_status表单域指定。如果上传失败，OSS返回错误码，并不进行跳转。 默认值：无	可选

名称	类型	描述	必须
success_action_status	字符串	未指定success_action_redirect表单域时，该表单域指定了上传成功后返回给客户端的状态码。 默认值：无 有效值：200、201、204（默认）。 说明： - 如果该域的值为200或者204，OSS返回一个空文档和相应的状态码。 - 如果该域的值设置为201，OSS返回一个XML文件和201状态码。 - 如果该域的值未设置或者设置成一个非法值，OSS返回一个空文档和204状态码。	非必须
x-oss-meta-*	字符串	用户指定的user meta值。 默认值：无	可选
x-oss-server-side-encryption	字符串	指定OSS创建Object时的服务器端加密编码算法。 取值：AES256 或 KMS（您需要购买KMS套件，才可以使用 KMS 加密算法，否则会报 KmsServiceNotEnabled 错误码） 指定此参数后，在响应头中会返回此参数，OSS会对上传的Object进行加密编码存储。当下载该Object时，响应头中会包含x-oss-server-side-encryption，且该值会被设置成该Object的加密算法。	可选

名称	类型	描述	必须
x-oss-server-side-encryption-key-id	字符串	表示KMS托管的用户主密钥。 该参数在x-oss-server-side-encryption的值为KMS时有效。	可选
x-oss-object-acl	字符串	指定OSS创建Object时的访问权限。 合法值：public-read、private、public-read-write	可选
x-oss-security-token	字符串	若本次访问是使用STS临时授权方式，则需要指定该项为SecurityToken的值，同时OSSAccessKeyId需要使用与之配对的临时AccessKeyId。计算签名时，与使用普通AccessKeyId签名方式一致。 默认值：无	可选

- 响应Header

名称	类型	描述
x-oss-server-side-encryption	字符串	如果请求指定了x-oss-server-side-encryption熵编码，则响应Header中包含了该头部，指明了所使用的加密算法。

- 响应元素

名称	类型	描述
PostResponse	容器	保存Post请求结果的容器。 子节点：Bucket, ETag, Key, Location
Bucket	字符串	Bucket名称。 父节点：PostResponse

名称	类型	描述
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，Post请求创建的Object，ETag值是该Object内容的uuid，可以用于检查该Object内容是否发生变化。 父节点：PostResponse
Location	字符串	新创建Object的URL。 父节点：PostResponse

- 细节分析
  - Post操作需要对Bucket拥有写权限。如果Bucket为public-read-write，可以不上传签名信息，否则要求对该操作进行签名验证。与Put操作不同，Post操作使用AccessKeySecret对policy进行签名，计算出签名字符串作为Signature表单域的值，OSS会验证该值从而判断签名的合法性。
  - 无论 Bucket 是否为 public-read-write，一旦上传 OSSAccessKeyId、policy、Signature 表单域中的任意一个，则另两个表单域为必选项，缺失时OSS会返回错误码：InvalidArgument。
  - Post操作提交表单编码必须为`multipart/form-data`，即Header中Content-Type为`multipart/form-data;boundary=xxxxxx`这样的形式，boundary为边界字符串。
  - 提交表单的URL为Bucket域名即可，不需要在URL中指定Object。即请求行是`POST / HTTP/1.1`，不能写成`POST /ObjectName HTTP/1.1`。
  - 表单和policy必须使用UTF-8编码。
  - 如果用户上传了Content-MD5请求Header，OSS会计算body的Content-MD5并检查一致性，如果不一致，将返回InvalidDigest错误码。
  - 如果POST请求中包含Header签名信息或URL签名信息，OSS不会对它们做检查。
  - 如果请求中携带以x-oss-meta-为前缀的表单域，则视为user meta，比如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的user meta总大小不能超过 8 KB。
  - Post请求的body总长度不允许超过5GB。若文件长度过大，则返回错误码：EntityTooLarge。
  - 如果上传指定了x-oss-server-side-encryption Header请求域，则必须设置其值为AES256与KMS，否则会返回400 错误，错误码：InvalidEncryptionAlgorithmError。指定该Header后，在响应头中也会返回该Header，OSS会对上传的Object进行加密编码存

储，当这个Object被下载时，响应Header中会包含x-oss-server-side-encryption，值被设置成该Object的加密算法。

- 表单域对大小写不敏感，但表单域的值对大小写敏感。

· 示例

- 请求示例：

```
POST / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 344606
Content-Type: multipart/form-data; boundary=9431149156168
--9431149156168
Content-Disposition: form-data; name="key"
/user/a/objectName.txt
--9431149156168
Content-Disposition: form-data; name="success_action_status"
200
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
content_disposition
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-uuid"
uuid
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-tag"
metadata
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
44CF9590006BF252F707
--9431149156168
Content-Disposition: form-data; name="policy"
eyJleHBpcmF0aW9uIjoiMjAxMy0xMi0wMVQxMjowMDowMFoiLCJjb25kaXRp
b25zIjpbWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDAsIDEwNDg1NzYwXSx7
ImJ1Y2tldCI6ImFoYWhhIn0sIHsiQSI6ICJhIn0seyJrZXkiOiAiQUJDIn1dfQ==
--9431149156168
Content-Disposition: form-data; name="Signature"
kZoYNv66bsmc10+dcGKw5x2PRrk=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.
txt"
Content-Type: text/plain
abcdefg
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OSS
--9431149156168--
```

- 返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 61d2042d-1b68-6708-5906-33d81921362e
Date: Fri, 24 Feb 2014 06:03:28 GMT
ETag: 5B3C1A2E053D763E1B002CC607C5****
Connection: keep-alive
Content-Length: 0
```

```
Server: AliyunOSS
```

Post Policy

Post请求的policy表单域用于验证请求的合法性。 policy为一段经过UTF-8和base64编码的JSON文本，声明了Post请求必须满足的条件。虽然对于public-read-write的Bucket上传时，Post表单域为可选项，但我们强烈建议使用该域来限制Post请求。

policy示例

```
{ "expiration": "2014-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"]
  ]
}
```

Post policy中必须包含expiration和conditions。

- Expiration

  Expiration项指定了policy的过期时间，以ISO8601 GMT时间表示。例如`2014-12-01T12:00:00.000Z`指定了Post请求必须发生在2014年12月1日12点之前。

- Conditions

  Conditions是一个列表，可以用于指定Post请求的表单域的合法值。

  > **说明:**
  > 表单域对应的值在检查policy之后进行扩展，因此，policy中设置的表单域的合法值应当对应于扩展之前的表单域的值。

  Policy中支持的conditions项见下表：

名称	描述
content-length-range	上传Object的最小和最大允许大小。 该condition支持contion-length-range匹配方式。
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	HTTP请求Header。 该condition支持精确匹配和starts-with匹配方式。
key	上传Object的object名称。 该condition支持精确匹配和starts-with匹配方式。
success_action_redirect	上传成功后的跳转URL地址。 该condition支持精确匹配和starts-with匹配方式。

名称	描述
success_action_status	未指定success_action_redirect时，上传成功后的返回状态码。 该condition支持精确匹配和starts-with匹配方式。
x-oss-meta-*	用户指定的user meta。 该condition支持精确匹配和starts-with匹配方式。

如果Post请求中包含额外的表单域，OSS可以将这些额外的表单域加入到policy的conditions中并检查合法性。

- Conditions匹配方式

Conditions匹配方式	描述
精确匹配	表单域的值必须精确匹配conditions中声明的值。如指定key表单域的值必须为a：{“key”：“a”}同样可以写为：[“eq”，“$key”，“a”]
Starts With	表单域的值必须以指定值开始。例如指定key的值必须以user/user1开始：[“starts-with”，“$key”，“user/user1”]
指定文件大小范围	指定所允许上传的文件最大和最小范围，例如允许的文件大小为1到10字节：[“content-length-range”, 1, 10]

转义字符

在 Post policy 中 $ 表示变量，如果要描述 $，需要使用转义字符 \$。除此之外，JSON 将对一些字符进行转义。下表描述了 Post policy 的 JSON 中需要进行转义的字符。

转义字符	描述
\/	斜杠
\	反斜杠
\”	双引号
\$	美元符
\b	空格
\f	换页
\n	换行
\r	回车
\t	水平制表符
\uxxxx	Unicode 字符

Post Signature

对于验证的Post请求，HTML表单中必须包含policy和Signature信息。Policy控制请求中哪些值是允许的。

计算Signature的具体流程为：

1. 创建一个 UTF-8 编码的 policy。
2. 将 policy 进行 base64 编码，其值即为 policy 表单域填入的值，将该值作为将要签名的字符串。
3. 使用 AccessKeySecret 对要签名的字符串进行签名，签名方法与Header中签名的计算方法相同（将要签名的字符串替换为 policy 即可），请参见在Header中包含签名。

示例 Demo

Web 端表单直传 OSS 示例 Demo，请参见JavaScript客户端签名直传。

7.12 Callback

用户只需要在发送给 OSS 的请求中携带相应的 Callback 参数，即能实现回调。本文详细介绍Callback的实现原理。

> 说明：
> 目前支持 Callback 的 API 接口有：PutObject、PostObject、CompleteMultipartUpload。更多 Callback 详情请参见原理介绍。

步骤1：构造参数

· Callback 参数

Callback 参数是由一段经过 base64 编码的 Json 字符串（字段）。构建 callback 参数的关键是指定请求回调的服务器 url（callbackUrl）以及回调的内容（callbackBody）。

Json 字段如下：

字段	含义	是否必需
callbackUrl	- 文件上传成功后，OSS 向此 url 发送回调请求，请求方法为 POST，body 为 callbackBody 指定的内容。正常情况下，该 url 需要响应 `HTTP/1.1 200 OK`，body 必须为 Json 格式，响应头 Content-Length 必须为合法的值，且不超过 3 MB。 - 支持同时配置最多 5 个 url，以分号（;）分割。OSS 会依次发送请求直到第一个返回成功为止。 - 如果没有配置或者值为空则认为没有配置 callback。 - 支持 HTTPS 地址。 - 为了保证正确处理中文等情况，callbackUrl 需做 url 编码处理，比如`http://example.com/中文.php?key=value&中文名称=中文值` 需要编码成 `http://example.com/%E4%B8%AD%E6%96%87.php?key=value&%E4%B8%AD%E6%96%87%E5%90%8D%E7%A7%B0=%E4%B8%AD%E6%96%87%E5%80%BC`	是
callbackHost	- 发起回调请求时 Host 头的值，只有在设置了 callbackUrl 时才有效。 - 如果没有配置 callbckHost，则会解析 callbackUrl 中的 url 并将解析出的 host 填充到 callbackHost 中。	否

字段	含义	是否必需
callbackBody	- 发起回调时请求 body 的值，例如：key=$(key)&etag=$(etag)&my_var=$(x:my_var)。 - 支持 OSS 系统变量、自定义变量和常量，支持的系统变量如下表所示 。自定义变量的支持方式在 PutObject 和CompleteMultipart 中是通过 callback-var 来传递，在PostObject中则是将各个变量通过表单域来传递。	是
callbackBodyType	- 发起回调请求的 Content-Type，支持 application/x-www-form-urlencoded 和 application/json，默认为前者。 - callbackBodyType的取值为 application/x-www-form-urlencoded，则 callbackBody 中的变量将会被经过 url 编码的值替换掉；如果为 application/json，则会按照 json 格式替换其中的变量。	否

Json 字段示例如下：

```
{
"callbackUrl":"121.101.166.30/test.php",
"callbackHost":"oss-cn-hangzhou.aliyuncs.com",
"callbackBody":"{\"mimeType\":${mimeType},\"size\":${size}}",
"callbackBodyType":"application/json"
}
```

```
{
"callbackUrl":"121.43.113.8:23456/index.html",
"callbackBody":"bucket=${bucket}&object=${object}&etag=${etag}&size
=${size}&mimeType=${mimeType}&imageInfo.height=${imageInfo.height
}&imageInfo.width=${imageInfo.width}&imageInfo.format=${imageInfo.
format}&my_var=${x:my_var}"
}
```

callbackBody 中可以设置的系统参数如下表所示：

系统参数	含义
bucket	存储空间
object	对象（文件）
etag	文件的 ETtag，即返回给用户的 ETag 字段
size	Object 大小，CompleteMultipartUpload 时为整个 Object 的大小

系统参数	含义
mimeType	资源类型，如 jpeg 图片的资源类型为 image/jpeg
imageInfo.height	图片高度
imageInfo.width	图片宽度
imageInfo.format	图片格式，如 jpg、png 等

说明:

imageInfo 针对图片格式，如果为非图片格式，imageInfo.height、imageInfo.width、imageInfo.format 都为空。

- callback-var 自定义参数

  用户可以通过 callback-var 参数来配置自定义参数。自定义参数是一个 Key-Value 的 Map，用户可以配置自己需要的参数到该 Map。在 OSS 发起 POST 回调请求的时候，会将这些参数和上述的系统参数一起放在 POST 请求的 body 中以方便接收回调方获取。

  构造自定义参数的方法和 callBack 参数的方法是一样的，也是以 Json 格式来传递。该 Json 字符串就是一个包含所有自定义参数的 Key-Value 的 Map。

  说明:

  用户自定义参数的 Key 一定要以 x: 开头且必须为小写，否则 OSS 会返回错误。

  假定用户需要设定两个自定义的参数分别为 x:var1 和 x:var2，对应的值分别为 value1 和 value2，那么构造出来的 Json 格式如下：

```
{
"x:var1":"value1",
"x:var2":"value2"
}
```

说明:

如果传入的 callback 或者 callback-var 参数不合法，则会返回 400 错误，错误码为”InvalidArgument”，不合法的情况包括以下几类：

- PutObject() 和 CompleteMultipartUpload() 接口中 url 和 header 同时传入 callback(x-oss-callback) 或者 callback-var(x-oss-callback-var) 参数。
- callback 或者 callback-var (PostObject() 由于没有 callback-var 参数，因此没有此限制，下同)参数过长（超过 5 KB）。
- callback 或者 callback-var 参数没有经过 base64 编码或经过 base64 解码后不是合法的 Json 格式。

- callback 参数解析后 callbackUrl 字段包含的 url 超过限制（5 个），或者 url 中传入的 port 不合法，比如

```
{"callbackUrl":"10.101.166.30:test",
    "callbackBody":"test"}
```

- callback 参数解析后 callbackBody 字段为空。
- callback 参数解析后 callbackBodyType 字段的值不是 `application/x-www-form-urlencoded` 或者 `application/json`。
- callback 参数解析后 callbackBody 字段中变量的格式不合法，合法的格式为 ${var}。
- callback-var 参数解析后不是预期的 Json格式，预期的格式应该为`{"x:var1":"value1","x:var2":"value2"...}`。

步骤2：构造回调请求

构造完成上述的 callback 和 callback-var 两个参数后，需将参数附加到 OSS 的请求中。

附加方式共有三种，方式如下：

- 在 URL 中携带参数。
- 在 Header 中携带参数。
- 在 POST 请求的 body 中使用表单域来携带参数。

> 说明：
> 在使用 POST 请求上传 object 时只能使用这种方式来指定回调参数。

以上三种方式只能同时使用其中一种，否则 OSS 会返回 InvalidArgument 错误。

要将参数附加到 OSS 的请求中，首先要将上文构造的 Json 字符串使用 base64 编码，然后按照如下的方法附加到 OSS 的请求中。

- 如果在 URL 中携带参数。把`callback=[CallBack]`或者`callback-var=[CallBackVar]`作为一个 url 参数带入请求发送。计算签名 CanonicalizedResource 时，将 callback 或者 callback-var 当做一个 sub-resource 计算在内。
- 如果在 Header 中携带参数。把`x-oss-callback=[CallBack]`或者`x-oss-callback-var=[CallBackVar]`作为一个 Header 带入请求发送。在计算签名CanonicalizedOSSHeaders 时，将 x-oss-callback-var 和 x-oss-callback 计算在内。示例如下：

```
PUT /test.txt HTTP/1.1
Host: callback-test.oss-test.aliyun-inc.com
Accept-ncoding: identity
Content-Length: 5
x-oss-callback-var: eyJ4Om15X3ZhciI6ImZvci1jYWxsYmFjay10ZXN0In0=
User-Agent: aliyun-sdk-python/0.4.0 (Linux/2.6.32-220.23.2.ali1089.
el5.x86_64/x86_64;2.5.4)
```

```
x-oss-callback: eyJjYWxsYmFja1VybCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbm
RleC5odG1sIiwgICJjYWxsYmFja0JvZHkiOiJidWNrZXQ9JHtidWNrZXR9Jm
9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW
1lVHlwZT0ke21pbWVUeXBlfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbm
ZvLmhlaWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLndpZHRofS
ZpbWFnZUluZm8uZm9ybWF0PSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPS
R7eDpteV92YXJ9In0=
Host: callback-test.oss-test.aliyun-inc.com
Expect: 100-Continue
Date: Mon, 14 Sep 2015 12:37:27 GMT
Content-Type: text/plain
Authorization: OSS mlepou3zr4u7b14:5a74vhd4UXpmyuudV14Kaen5****
Test
```

- 在 POST 请求的 body 中使用表单域来携带参数。
  - 如果需要在 POST 上传 Object 时附带回调参数会稍微复杂一点，callback 参数要使用独立的表单域来附加，示例如下：

```
--9431149156168
Content-Disposition: form-data; name="callback"
eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5w
aHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tC
b2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIs
ImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVy
bGVuY29kZWQifQ==
```

  - 如果拥有自定义参数的话，不能直接将 callback-var 参数直接附加到表单域中，每个自定义的参数都需要使用独立的表单域来附加。示例如下，如果用户的自定义参数 Json 字段为

```
{
"x:var1":"value1",
"x:var2":"value2"
}
```

    那么 POST 请求的表单域为：

```
--9431149156168
Content-Disposition: form-data; name="callback"
eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5w
aHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tC
b2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIs
ImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVy
bGVuY29kZWQifQ==
--9431149156168
Content-Disposition: form-data; name="x:var1"
value1
--9431149156168
Content-Disposition: form-data; name="x:var2"
value2
```

    同时可以在 policy 中添加 callback 条件（如果不添加 callback，则不对该参数做上传验证）如：

```
{ "expiration": "2014-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith" },
```

```
    {"callback": "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6
ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2
LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJj
YWxsYmFja0JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxl
bmNvZGVkIn0="},
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

步骤3：发起回调请求

如果文件上传成功，OSS 会根据用户请求中的 callback 参数和 callback-var 自定义参数，将特定内容以 POST 方式发送给应用服务器。

```
POST /index.html HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 181
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
bucket=callback-test&object=test.txt&etag=D8E8FCA2DC0F896FD7CB
4CB0031BA249&size=5&mimeType=text%2Fplain&imageInfo.height=&imageInfo.
width=&imageInfo.format=&x:var1=for-callback-test
```

（可选）步骤4：回调签名

用户设置 callback 参数后，OSS 将按照用户设置的 callbackUrl 发送 POST 回调请求给用户的应用服务器。应用服务器收到回调请求之后，如果希望验证回调请求确实是由 OSS 发起的话，可以通过在回调中带上签名来验证 OSS 的身份。

- 生成签名

  签名发生在 OSS 端，采用 RSA 非对称方式。

  私钥加密生成签名的过程为：

  ```
  authorization = base64_encode(rsa_sign(private_key, url_decode(path)
   + query_string + '\n' + body, md5))
  ```

   说明：

其中 private_key 为私钥，只有 OSS 知晓，path 为回调请求的资源路径，query_string 为查询字符串，body 为回调的消息体。

签名过程分为以下三步：

1. 获取待签名字符串：资源路径经过 url 解码后，加上原始的查询字符串，加上一个回车符，加上回调消息体
2. RSA签名：使用秘钥对待签名字符串进行签名，签名的 hash 函数为 md5
3. 将签名后的结果做 base64 编码，得到最终的签名，签名放在回调请求的 authorization 头中

示例如下：

```
POST /index.php?id=1&index=2 HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 18
authorization: kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgb
WEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2t****
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9j
YWxsYmFja19wdWJfa2V5X3YxLnBlbQ==
bucket=yonghu-test
```

path 为`/index.php`，query_string 为`?id=1&index=2`，body 为`bucket=yonghu-test`，最终签名结果为`kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==`。

- 验证签名

验证签名的过程即为签名的逆过程，由应用服务器验证，过程如下：

```
Result = rsa_verify(public_key, md5(url_decode(path) + query_string
 + ‘\n’ + body), base64_decode(authorization))
```

字段的含义与签名过程中描述相同，其中 public_key 为公钥， authorization 为回调头中的签名，整个验证签名的过程分为以下几步：

1. 回调请求的 x-oss-pub-key-url 头保存的是公钥 url 地址的 base64 编码，因此需要对其做 base64 解码后获取到公钥，即

   ```
   public_key = urlopen(base64_decode(x-oss-pub-key-url头的值))
   ```

   说明:

了保证这个 publickey 是由 OSS 颁发的，用户需要校验x-oss-pub-key-url头的值必须以http://gosspublic.alicdn.com/或者https://gosspublic.alicdn.com/开头。

2. 获取 base64 解码后的签名

```
signature = base64_decode(authorization头的值)
```

3. 获取待签名字符串，方法与签名一致

```
sign_str = url_decode(path) + query_string + '\n' + body
```

4. 验证签名

```
result = rsa_verify(public_key, md5(sign_str), signature)
```

以上例为例：

1. 获取到公钥的 url 地址，即aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==经过base64 解码后得到http://gosspublic.alicdn.com/callback_pub_key_v1.pem。
2. 签名头kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==做 base64 解码（由于为非打印字符，无法显示出解码后的结果）。
3. 获取待签名字符串，即 url_decode(“index.php”) + “?id=1&index=2” + “\n” + “bucket=yonghu-test”，并做 MD5校验。
4. 验证签名。

· 应用服务器示例

以下为一段 Python 示例，演示了一个简单的应用服务器，主要是说明验证签名的方法，此示例需要安装 M2Crypto库。

```
import httplib
import base64
import md5
import urllib2
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
from M2Crypto import RSA
from M2Crypto import BIO
def get_local_ip():
    try:
        csock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        csock.connect(('8.8.8.8', 80))
        (addr, port) = csock.getsockname()
        csock.close()
        return addr
    except socket.error:
        return ""
class MyHTTPRequestHandler(BaseHTTPRequestHandler):
```

```
    '''
    def log_message(self, format, *args):
        return
    '''
    def do_POST(self):
        #get public key
        pub_key_url = ''
        try:
            pub_key_url_base64 = self.headers['x-oss-pub-key-url']
            pub_key_url = pub_key_url_base64.decode('base64')
            if not pub_key_url.startswith("http://gosspublic.alicdn.
com/") and not pub_key_url.startswith("https://gosspublic.alicdn.com
/"):
                self.send_response(400)
                self.end_headers()
                return
            url_reader = urllib2.urlopen(pub_key_url)
            #you can cache it
            pub_key = url_reader.read()
        except:
            print 'pub_key_url : ' + pub_key_url
            print 'Get pub key failed!'
            self.send_response(400)
            self.end_headers()
            return
        #get authorization
        authorization_base64 = self.headers['authorization']
        authorization = authorization_base64.decode('base64')
        #get callback body
        content_length = self.headers['content-length']
        callback_body = self.rfile.read(int(content_length))
        #compose authorization string
        auth_str = ''
        pos = self.path.find('?')
        if -1 == pos:
            auth_str = urllib2.unquote(self.path) + '\n' +
callback_body
        else:
            auth_str = urllib2.unquote(self.path[0:pos]) + self.path
[pos:] + '\n' + callback_body
        print auth_str
        #verify authorization
        auth_md5 = md5.new(auth_str).digest()
        bio = BIO.MemoryBuffer(pub_key)
        rsa_pub = RSA.load_pub_key_bio(bio)
        try:
            result = rsa_pub.verify(auth_md5, authorization, 'md5')
        except:
            result = False
        if not result:
            print 'Authorization verify failed!'
            print 'Public key : %s' % (pub_key)
            print 'Auth string : %s' % (auth_str)
            self.send_response(400)
            self.end_headers()
            return
        #do something accoding to callback_body
        #response to OSS
        resp_body = '{"Status":"OK"}'
        self.send_response(200)
        self.send_header('Content-Type', 'application/json')
        self.send_header('Content-Length', str(len(resp_body)))
        self.end_headers()
        self.wfile.write(resp_body)
```

```
class MyHTTPServer(HTTPServer):
    def __init__(self, host, port):
        HTTPServer.__init__(self, (host, port), MyHTTPRequestHandler
)
if '__main__' == __name__:
    server_ip = get_local_ip()
server_port = 23451
server = MyHTTPServer(server_ip, server_port)
server.serve_forever()
```

其它语言的服务端代码如下：

Java 版本：

- 下载地址：点击这里
- 运行方法：解压包运行`java -jar oss-callback-server-demo.jar 9000`（9000 就运行的端口，可以自己指定）

PHP 版本：

- 下载地址：点击这里
- 运行方法：部署到 Apache 环境下，因为 PHP 本身语言的特点，取一些数据头部会依赖于环境。所以可以参考例子根据所在环境修改。

Python 版本：

- 下载地址：点击这里
- 运行方法：解压包直接运行`python callback_app_server.py`，运行该程序需要安装 RSA 的依赖。

.NET 版本：

- 下载地址：点击这里
- 运行方法：解压后参看`README.md`。

Go 版本：

- 下载地址：点击这里
- 运行方法：解压后参看`README.md`。

Ruby 版本：

- 下载地址：点击这里
- 运行方法： ruby aliyun_oss_callback_server.rb

步骤5：返回回调结果

应用服务器返回响应给OSS。

返回的回调请求为：

```
HTTP/1.0 200 OK
Server: BaseHTTP/0.3 Python/2.7.6
Date: Mon, 14 Sep 2015 12:37:27 GMT
Content-Type: application/json
Content-Length: 9
{"a":"b"}
```

说明：

应用服务器返回 OSS 的响应必须带有 Content-Length 的 Header，Body 大小不要超过 1MB。

步骤6：返回上传结果

OSS将应用服务器返回的内容返回给用户。

返回的内容响应为：

```
HTTP/1.1 200 OK
Date: Mon, 14 Sep 2015 12:37:27 GMT
Content-Type: application/json
Content-Length: 9
Connection: keep-alive
ETag: "D8E8FCA2DC0F896FD7CB4CB0031BA249"
Server: AliyunOSS
x-oss-bucket-version: 1442231779
x-oss-request-id: 55F6BF87207FB30F2640C548
{"a":"b"}
```

说明：

- 如果类似 CompleteMultipartUpload 这样的请求，在返回请求本身 body 中存在内容（如 XML 格式的信息），使用上传回调功能后会覆盖原有的 body 中的内容如`{"a":"b"}`，希望对此处做好判断处理。
- 如果回调失败，则返回 203，错误码为 ”CallbackFailed”。文件已经成功上传到了 OSS，但回调失败。回调失败只是表示 OSS 没有收到预期的回调响应，不代表应用服务器没有收到回调请求（比如应用服务器返回的内容不是 Json 格式）。

7.13 PutSymlink

PutSymlink接口用于为OSS的TargetObject创建软链接，您可以通过该软链接访问TargetObject。

请求语法

```
PUT /ObjectName?symlink HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
```

```
Date: GMT Date
Authorization: SignatureValue
x-oss-symlink-target: TargetObjectName
```

请求头

名称	类型	是否必选	描述
x-oss-symlink-target	字符串	是	软链接指向的目标文件。 合法值：命名规范同Object 说明： · TargetObjectName同ObjectName一样，需要URL encode。 · 软链接的目标文件类型不能为软链接。
x-oss-storage-class	字符串	否	指定Object的存储类型。 取值：Standard、IA、Archive 说明： · IA与Archive类型的单个文件如不足64 KB，会按64 KB计量计费。建议您在使用Symlink接口时不要将Object的存储类型指定为IA或Archive。 · 对于任意存储类型的Bucket，若上传Object时指定此参数，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject

细节分析

· 使用PutSymlink接口创建软链接时不会做以下检查，以下检查会在GetObject等需要访问目标文件的API中进行。
  - 不检查目标文件是否存在。
  - 不检查目标文件类型是否合法。
  - 不检查目标文件是否有权限访问。
· 如果试图添加的文件已经存在，并且有访问权限，新添加的文件将覆盖原来的文件，成功将返回200 OK。

- 使用PutSymlink接口时，携带以x-oss-meta-为前缀的参数，则视为user meta，例如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的user meta总大小不能超过8 KB。

示例

- 请求示例

```
PUT /link-to-oss.jpg?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
Content-Disposition: attachment;filename=oss_download.jpg
Date: Tue, 08 Nov 2016 02:00:25 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2
****= x-oss-symlink-target: oss****
x-oss-storage-class: Standard
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 08 Nov 2016 02:00:25 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 582131B9109F4EE66CDE56A5
ETag: "0A477B89B4602AA8DECB8E19BFD4****"
```

SDK

PutSymlink接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET

错误码

错误码	HTTP 状态码	描述
InvalidArgument	400	StorageClass的值不合法。

7.14 GetSymlink

GetSymlink接口用于获取软链接。此操作需要您对该软链接有读权限。

请求语法

```
GET /ObjectName?symlink HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应头

名称	类型	描述
x-oss-symlink-target	字符串	软链接指向的目标文件。

示例

- 请求示例

```
GET /link-to-oss.jpg?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 06:38:30 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:UNQDb7GapEgJCZkcde6OhZ9J
****
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5650BD72207FB30443962F9A
x-oss-symlink-target: oss.jpg
ETag: "A797938C31D59EDD08D86188F6D5****"
```

SDK

GetSymlink接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET

错误码

错误码	HTTP 状态码	描述
NoSuchKey	404	软链接不存在。

7.15 RestoreObject

RestoreObject接口用于解冻归档类型（Archive）的文件（Object）。

说明:

- RestoreObject接口只针对归档类型的Object，不适用于标准类型和低频访问类型的Object。
- 如果针对该Object第一次调用RestoreObject接口，返回202。
- 如果已经成功调用过RestoreObject接口，且Object已完成解冻，再次调用时返回200 OK。

解冻过程说明

归档类型的Object在执行解冻前后的状态变换过程如下：

1. 归档类型的Object初始时处于冷冻状态。
2. 提交一次解冻请求后，Object处于解冻中的状态。完成解冻任务通常需要1分钟，最长等待任务完成时间为4小时。
3. 服务端完成解冻任务后，Object进入解冻状态，此时您可以读取Object。解冻状态默认持续24小时，24小时内再次调用RestoreObject接口则解冻状态会自动延长24小时。对于同份归档文件，一次解冻流程内可有效调用7次RestoreObject接口达到最长7天的解冻持续时间。
4. 解冻状态结束后，Object再次返回到冷冻状态。

计费说明

状态变换过程中产生的相关费用如下：

- 对处于冷冻状态的Object执行解冻操作，会产生数据取回费用。
- 解冻状态最多延长7天。在此期间不再重复收取数据取回费用。
- 解冻状态结束后，Object又回到冷冻状态，再次执行解冻操作会收取数据取回费用。

请求语法

```
POST /ObjectName?restore HTTP/1.1
Host: archive-bucket.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
```

```
Authorization: SignatureValue
```

示例

- **首次对处于冷冻状态的Object提交解冻请求**

  请求示例

  ```
  POST /oss.jpg?restore HTTP/1.1
  Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
  Date: Sat, 15 Apr 2017 07:45:28 GMT
  Authorization: OSS e1Unnbm1rgdnpI:y4eyu+4yje5ioRCr****
  ```

  返回示例

  ```
  HTTP/1.1 202 Accepted
  Date: Sat, 15 Apr 2017 07:45:28 GMT
  Content-Length: 0
  Connection: keep-alive
  Server: AliyunOSS
  x-oss-request-id: 5374A2880232A65C23002D74
  ```

- **对处于解冻中状态的Object提交解冻请求**

  请求示例

  ```
  POST /oss.jpg?restore HTTP/1.1
  Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
  Date: Sat, 15 Apr 2017 07:45:29 GMT
  Authorization: OSS e1Unnbm1rgdnpI:21qtGJ+ykDVmdy4eyu+N****
  ```

  返回示例

  ```
  HTTP/1.1 409 Conflict
  Date: Sat, 15 Apr 2017 07:45:29 GMT
  Content-Length: 556
  Connection: keep-alive
  Server: AliyunOSS
  x-oss-request-id: 5374A2880232A65C23002D74
  <?xml version="1.0" encoding="UTF-8"?>
  <Error>
    <Code>RestoreAlreadyInProgress</Code>
    <Message>The restore operation is in progress.</Message>
    <RequestId>58EAF141461FB42C2B000008</RequestId>
    <HostId>10.101.200.***</HostId>
  </Error>
  ```

- **对处于解冻状态的Object提交解冻请求**

  请求示例

  ```
  POST /oss.jpg?restore HTTP/1.1
  Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
  Date: Sat, 15 Apr 2017 07:45:29 GMT
  ```

```
Authorization: OSS e1Unnbm1rgdnpI:u6O6FMJnn+WuBwbByZxm1+y4eyu+N****
```

返回示例

```
HTTP/1.1 200 Ok
Date: Sat, 15 Apr 2017 07:45:30 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C
- .NET

错误码

错误码	HTTP 状态码	描述
NoSuchKey	404	目标Object不存在。
OperationNotSupported	400	目标Object不是归档类型。
RestoreAlreadyInProgress	409	您已经成功调用过RestoreObject接口，且服务端正在执行解冻操作。请不要重复提交RestoreObject。

7.16 SelectObject

SelectObject用于查询某个Object，此操作要求用户对该Object有读权限。

SelectObject

SelectObject API 用于对目标文件执行SQL语句，返回执行结果。

正确执行时，该API返回206。如果SQL语句不正确，或者和文件不匹配，则会返回400错误。

> 说明：
> 关于SelectObject的功能介绍请参见开发指南中的SelectObject。

- **请求语法**

  - **请求语法（CSV）**

```
POST /object?x-oss-process=csv/select HTTP/1.1
HOST: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: time GMT
Content-Length: ContentLength
Content-MD5: MD5Value
Authorization: Signature
<?xml  version="1.0"  encoding="UTF-8"?>
<SelectRequest>
 <Expression>base64 encode(Select * from OSSObject where ...)</
Expression>
 <InputSerialization>
  <CompressionType>None|GZIP</CompressionType>
  <CSV>
   <FileHeaderInfo>
    NONE|IGNORE|USE
   </FileHeaderInfo>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>
   <QuoteCharacter>base64 encode</QuoteCharacter>
   <CommentCharacter>base64 encode</CommentCharacter>
   <Range>line-range=start-end|split-range=start-end</Range>
  </CSV>
 </InputSerialization>
 <OutputSerialization>
                 <CSV>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>

  </CSV>
         <KeepAllColumns>false|true</KeepAllColumns>
         <OutputRawData>false|true</OutputRawData>
              <EnablePayloadCrc>true</EnablePayloadCrc>
         <OutputHeader>false</OutputHeader>
 </OutputSerialization>
 <Options>
     <SkipPartialDataRecord>false</SkipPartialDataRecord>
     <MaxSkippedRecordsAllowed>
  max allowed number of records skipped
     <MaxSkippedRecordsAllowed>
 </Options>
</SelectRequest>
```

  - **请求语法（JSON）**

```
POST /object?x-oss-process=json/select HTTP/1.1
HOST: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: time GMT
Content-Length: ContentLength
Content-MD5: MD5Value
Authorization: Signature
<?xml  version="1.0"  encoding="UTF-8"?>
<SelectRequest>
 <Expression>
  Base64 encode of sql such as (select * from ossobject)
 </Expression>
 <InputSerialization>
  <CompressionType>None|GZIP</CompressionType>
  <JSON>
```

```
    <Type>DOCUMENT|LINES</Type>
    <Range>
    line-range=start-end|split-range=start-end
    </Range>
    <ParseJsonNumberAsString> true|false
    </ParseJsonNumberAsString>
   </JSON>
  </InputSerialization>
  <OutputSerialization>
   <JSON>
    <RecordDelimiter>
     Base64 of record delimiter
    </RecordDelimiter>
   </JSON>
   <OutputRawData>false|true</OutputRawData>
               <EnablePayloadCrc>true</EnablePayloadCrc>
  </OutputSerialization>
  <Options>
   <SkipPartialDataRecord>
    false|true
   </SkipPartialDataRecord>
   <MaxSkippedRecordsAllowed>
    max allowed number of records skipped
      <MaxSkippedRecordsAllowed>
         </Options>
</SelectRequest>
```

- **请求元素**

名称	类型	描述
SelectRequest	容器	保存Select请求的容器 子节点：Expression，InputSerialization，OutputSerialization 父节点：None
Expression	字符串	以Base64 编码的SQL语句 子节点：None 父节点：SelectRequest
InputSerialization	容器	输入序列化参数（可选） 子节点：CompressionType、CSV、JSON 父节点：SelectRequest
OutputSerialization	容器	输出序列化参数（可选） 子节点：CSV、JSON、OutputRawData 父节点：SelectRequest

名称	类型	描述
CSV(InputSerialization)	容器	输入CSV的格式参数（可选） 子节点：FileHeaderInfo、RecordDelimiter、FieldDelimiter、QuoteCharacter、CommentCharacter、 Range 父节点：InputSerialization
CSV(OutputSerialization)	容器	输出CSV的格式参数（可选） 子节点： RecordDelimiter、 FieldDelimiter 父节点：OutputSerialization
JSON(InputSerialization)	容器	输入JSON的格式参数（可选） 子节点：Type、Range、ParseJsonNumberAsString
JSON(OutputSerialization)	容器	输出JSON的格式参数（可选） 子节点：RecordDelimiter
Type	枚举	指定输入JSON的类型：DOCUMENT 、LINES
OutputRawData	bool，默认false	指定输出数据为纯数据（不是下面提到的基于Frame格式）（可选） 子节点：None 父节点：OutputSerialization
CompressionType	枚举	指定文件压缩类型：None|GZIP 子节点：None 父节点：InputSerialization

名称	类型	描述
FileHeaderInfo	枚举	指定CSV文件头信息（可选） 取值： - Use：该CSV文件有头信息，可以用CSV列名作为Select中的列名。 - Ignore：该CSV文件有头信息，但不可用CSV列名作为Select中的列名。 - None：该文件没有头信息，为默认值。 子节点：None 父节点：CSV （输入）
RecordDelimiter	字符串	指定换行符，以Base64编码。默认值为\n（可选）。未编码前的值最多为两个字符，以字符的ANSI值表示，比如在Java里用\n表示换行。 子节点：None 父节点：CSV （输入、输出），JSON（输出）
FieldDelimiter	字符串	指定CSV列分隔符，以Base64编码。默认值为,（可选）。未编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用, 表示逗号。 子节点：None 父节点：CSV （输入，输出）
QuoteCharacter	字符串	指定CSV的引号字符，以Base64编码。默认值为\”（可选）。在CSV中引号内的换行符，列分隔符将被视作普通字符。未编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用\”表示引号。 子节点：None 父节点：CSV （输入）

名称	类型	描述
CommentCharacter	字符串	指定CSV的注释符，以Base464编码。默认值为空（即没有注释符）。
Range	字符串	指定查询文件的范围（可选）。支持两种格式： - 按行查询：line-range=start-end - 按Split查询：split-range=start-end 其中start和end均为inclusive。其格式和range get中的range参数一致。 仅在文档是CSV或者JSON Type为LINES时使用。 子节点：None 父节点：CSV （输入）、JSON（输入）
KeepAllColumns	bool	指定返回结果中包含CSV所有列的位置（可选，默认值为false）。但仅仅在select 语句里出现的列会有值，不出现的列则为空，返回结果中每一行的数据按照CSV列的顺序从低到高排列。比如下面语句： `select _5, _1 from ossobject.` 如果KeepAllColumn = true，假设一共有6列数据，则返回的数据如下： Value of 1st column,,,,Value of 5th column,\n 子节点：None 父节点：OutputSerialization（Csv）

名称	类型	描述
EnablePayloadCrc	bool	在每个Frame中会有一个32位的crc32校验值。客户端可以计算相应payload的Crc32值进行数据完整性校验。 子节点：None 父节点：OutputSerialization
Options	容器	额外的可选参数 类型：容器 子节点：SkipPartialDataRecord、MaxSkippedRecordsAllowed 父节点：SelectRequest
OutputHeader	bool	在返回结果开头输出CSV头信息。 类型：bool，默认 false。 子节点：None 父节点：OutputSerialization
SkipPartialDataRecord	bool	忽略缺失数据的行。当该参数为false时，OSS会忽略缺失某些列(该列值当做null)而不报错。当该参数为true时，该行数据因为不完整而被整体跳过。当跳过的行数超过指定的最大跳过行数时OSS会报错并停止处理。 类型：bool，默认 false。 子节点:None 父节点：Options

名称	类型	描述
MaxSkippedRecordsAllowed	Int	指定最大能容忍的跳过的行数。当某一行数据因为不匹配SQL中期望的类型、或者某一列或者多列数据缺失且SkipPartialDataRecord为True时，该行数据会被跳过。如果跳过的行数超过该参数的值，OSS会停止处理并报错。 说明: 如果某一行是非法CSV行，比如在一列中间连续含有奇数个quote字符，则OSS会马上停止处理并报错，因为该错误很可能会影响对整个CSV文件的解析。即该参数用来调整对非整齐数据的容忍度，但不应用于非法的CSV文件。 类型：int，默认 0。 子节点:None 父节点：Options
ParseJsonNumberAsString	bool	将Json中的数字（整数和浮点数）解析成字符串。目前Json中的浮点数解析时会损失精度，如果要完整保留原始数据，则推荐用该选项。如果需要进行数值计算，则可以在Sql中cast成需要的格式，比如int、double、decimal。 类型：bool，默认 false。 子节点:None 父节点：JSON

- 返回Body

  当请求响应中HTTP Status是4xx，表明该请求没有通过相应的SQL语法检查或者目标文件有明显的问题，此时返回错误信息的Body，和Get API返回的错误信息一致。

  当请求返回5xx，则表明服务器内部错误，返回的错误信息格式和Get API返回的错误信息一致。

  当请求成功返回206时，若header x-oss-select-output-raw值为true，则表明返回结果是对象数据（而不是Frame包装的），客户端可以完全按照Get API的方式获取数据。

  当x-oss-select-output-raw的值为false时，请求结果以一个个Frame形式返回。

  当用户在请求中指定OutputRawData值时，OSS服务端会按照请求中的要求返回数据。当用户不指定OutputRawData时，OSS服务端会自动选择一种格式返回，这是推荐做法。

  当用户显式地指定OutputRawData为True时，其副作用在于如果该SQL很长时间内没有数据返回，http请求可能因没有数据而超时。

  当Select Json文件时，如果SQL select中有重复的Key（比如 select s.key, s.key from ossobject s），则响应Header x-oss-select-output-json-dup-key的值为true。

  每个Frame的格式如下，其中checksum均为CRC32：

  Version|Frame-Type | Payload Length | Header Checksum | Payload | Payload Checksum

  <1 byte><--3 bytes--><---4 bytes----><-------4 bytes--><variable><----4bytes---------->

  Frame里所有的整数均以大端编码(big endian)。 Version目前为1。

  对于SelectObject这个API一共有三种不同的Frame Type， 列举如下：

名称	Frame-Type值	Payload格式	描述
Data Frame	8388609	offset | data <-8 bytes><---variable->	DataFrame包含Select请求返回的数据，并用offset来汇报进展，offset为当前扫描位置（从文件头开始的偏移），是8位整数。
Continuous Frame	8388612	offset <----8 bytes-->	Continuous Frame用以汇报当前进展以及维持http连接。如果该查询在5s内未返回数据则会返回一个Continuous Frame。

名称	Frame-Type值	Payload格式	描述
End Frame	8388613	offset | total scanned bytes | http status code | error message <--8bytes-><--8bytes---------><----4 bytes--------><-variable------>	End Frame用来返回最终的状态，扫描的字节数以及可能的出错信息。其中offset为扫描后最终的位置偏移，total scanned bytes为最终扫描过的数据大小。http status code为最终的处理结果，error message为错误信息，包括所有跳过的行号及其总行数。 这里返回status code的原因在于SelectObject为流式处理，因而在发送Response Header的时候仅仅处理了第一个Block。如果第一个Block数据和SQL是匹配的，则在Response Header中的Status为206，但如果下面的数据非法，我们已无法更改Header中的Status，只能在End Frame里包含最终的Status及其出错信息。因此客户端应该视其为最终状态。

- 关于Error Message

  End Frame中的Error Message格式如下：

  `ErrorCodes.DetailMessage`

  其中ErrorCodes包含一个或者多个ErrorCode，用逗号隔开。ErrorCodes和DetailMessage之间用句号（.）隔开。具体的ErrorCode列表请参见下面ErrorCode。

- 样例请求

  - 样例请求（Csv）

```
POST /oss-select/bigcsv_normal.csv?x-oss-process=csv%2Fselect HTTP
/1.1
Date: Fri, 25 May 2018 22:11:39 GMT
Content-Type:
Authorization: OSS LTAIJPLocA0fD:FC/9JRbBGRw4o2QqdaL246Px****
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.
0.30319.42000)
Content-Length: 748
Expect: 100-continue
Connection: keep-alive
Host: host name
```

```
<?xml version="1.0"?>
<SelectRequest>
 <Expression>c2VsZWN0IGNvdW50KCopIGZyb20gb3Nzb2JqZWN0IHdoZXJlIF
80ID4gNDU=
 </Expression>
 <InputSerialization>
  <Compression>None</Compression>
  <CSV>
   <FileHeaderInfo>Ignore</FileHeaderInfo>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
   <CommentCharacter>Iw==</CommentCharacter/>
  </CSV>
 </InputSerialization>
 <OutputSerialization>
  <CSV>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
  </CSV>
  <KeepAllColumns>false</KeepAllColumns>
         <OutputRawData>false</OutputRawData>
 </OutputSerialization>
</SelectRequest>
```

- **样例请求（Json）**

```
POST /oss-select/sample_json.json?x-oss-process=json%2Fselect HTTP
/1.1
Host: host name
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Darwin/16.7.0/x86_64;3.5.4)
Accept: */*
Connection: keep-alive
date: Mon, 10 Dec 2018 18:28:11 GMT
authorization: OSS AccessKeySignature
Content-Length: 317
<SelectRequest>
 <Expression>c2VsZWN0ICogZnJvbSBvc3NvYmplY3Qub2JqZWN0c1sqXSB3aG
VyZSBwYXJ0eSA9ICdEZW1vY3JhdCc=
 </Expression>
 <InputSerialization>
 <JSON>
  <Type>DOCUMENT</Type>
 </JSON>
 </InputSerialization>
 <OutputSerialization>
 <JSON>
  <RecordDelimiter>LA==</RecordDelimiter>
 </JSON>
 </OutputSerialization>
 <Options />
</SelectRequest>
```

- **SQL语句正则表达式**

`SELECT select-list from table where_opt limit_opt`

其中SELECT、OSSOBJECT以及 WHERE为关键字不得更改。

```
select_list: column name
```

```
| column index (比如_1, _2. 仅Csv文件有column index)
| json path (比如s.contacts.firstname 仅应用在Json文件)
| function(column index | column name)
| function(json_path) (仅应用在Json 文件)
| select_list AS alias
```

支持的function为AVG、SUM、MAX、MIN、COUNT、 CAST(类型转换函数)。其中COUNT后只能用*。

table: OSSOBJECT

| OSSOBJECT json_path（仅应用于Json文件）

对于CSV文件，table必须为OSSOBJECT。对于JSON文件（包括DOCUMENT或者LINES），table可以在OSSOBJECT后指定json_path。

json_path: ['string '] （string若没有空格或者*，则引号可以去掉；等价于.'string '）

| [n] （应用在数组元素中，表示第n个元素，从0开始）

| [*] （应用在数组或者对象中，表示任意子元素）

| .'string ' （string若没有空格或者*，则引号可以去掉）

| json_path jsonpath （json path可以连接，比如 [n].property1.attributes[*]）

```
Where_opt:
| WHERE expr
expr:
| literal value
| column name
| column index
| json path (仅应用于Json文件)
| expr op expr
| expr OR expr
| expr AND expr
| expr IS NULL
| expr IS NOT NULL
| (column name | column index | json path) IN (value1, value2,….)
| (column name | column index | json path) NOT in (value1, value2,…)
| (column name | column index | json path) between value1 and value2
| NOT (expr)
| expr op expr
| (expr)
```

```
| cast (column index |column name | json path | literal as INT|
DOUBLE|)
```

op：包括 >、< 、>=、<=、!=、 =、， 、LIKE、+、-、*、/、%以及字符串连接||。

cast: 对于同一个column，只能cast成一种类型。

```
limit_opt:
```

| limit 整数

聚合和Limit的混用

```
Select avg(cast(_1 as int)) from ossobject limit 100
```

对于上面的语句，其含义是指在前100行中计算第一列的AVG值。这个行为和MY SQL不同，原因是在 SelectObject中聚合永远只返回一行数据，因而对聚合来说限制其输出规模是多余的。因此SelectObject里limit 将先于聚合函数执行。

SQL语句限制

- 目前仅仅支持UTF-8编码的文本文件以及GZIP压缩过的UTF-8文本。GZIP文件不支持deflate格式。
- 仅支持单文件查询，不支持join、order by、group by、having。
- Where语句里不能包含聚合条件（e.g. where max(cast(age as int)) > 100这个是不允许的）。
- 支持的最大的列数是1000，SQL中最大的列名称为1024。
- 在LIKE语句中，支持最多5个%通配符。*和%是等价的，表示0或多个任意字符。LIKE支持Escape关键字，用于将Pattern中的%*?特殊字符Escape成普通字符串。
- 在IN语句中，最多支持1024个常量项。
- Select后的Projection可以是列名，CSV列索引(_1, _2等)，或者是聚合函数，或者是CAST函数；不支持其他表达式。 比如select _1 + _2 from ossobject是不允许的。
- 支持的CSV最大行及最大列长度都是256K。
- 支持的 from 后json path指定的JSON最大节点为512K，最大深度为10层，其数组内元素的个数最多为5000。
- JSON文件的SQL中，select 或者where条件的表达式中不能有[*]数组通配符，数组通配符只能出现在from后的json path中。（比如select s.contacts[*] from ossobject s是不允许的，但是select * from ossobject.contacts[*]是可以的）。
- SQL最大长度16K，where后面表达式个数最多20个，表达式深度最多10层，聚合操作最多100个。

- 对于数据的容错处理
  - CSV的某些行缺失了某些列

    当SkipPartialDataRecord没有被指定或者其值为False时，OSS会把缺失的列都当做null来计算SQL中的表达式。

    当SkipPartialDataRecord为True时，OSS则会忽略该行数据。这时，如果MaxSkippedRecordsAllowed未指定或者指定的值小于忽略的行数，则OSS会返回400错误（通过http status code或者End Frame返回）。

    比如：Sql为`select _1, _3 from ossobject`

    而CSV的某一行为 “张小,阿里巴巴”，则当SkipPartialDataRecord为False时，返回内容为”张小,\n”；当SkipPartialDataRecord为True时，该行被跳过。
  - JSON缺失某些Key

    和Csv一样，JSON的某些对象可能未包括SQL中指定的Key。当SkipPartialDataRecord没有被指定或者其值为False时，这时OSS会把缺失的Key当做Null来计算SQL中的表达式。

    如果SkipPartialDataRecord为true时，OSS则会忽略该Json节点的数据 。这时，如果MaxSkippedRecordsAllowed未指定或者指定的值小于忽略的行数，则OSS会返回400错误（通过http status code或者End Frame返回）。

    比如：Sql为 `select s.firstName, s.lastName , s.age from ossobject.contacts[*] s`

    某个Json节点为{ “firstName” :” 小” , ” lastName” :” 张” }，当SkipPartialDataRecord为False或未指定时，该节点的返回值为{ “firstName” :” 小” , “lastName” :” 张” }。当SkipPartialDataRecord为True时则该行被跳过。
  - CSV的某些列类型不匹配SQL

    当CSV中一些行的数据类型和SQL中指定的不匹配时，该行数据会被忽略。和上面情况类似，其行为受到MaxSkippedRecordsAllowed的影响。当忽略行数超过上限时，处理中转并返回400。

    比如：SQL为`select _1, _3 from ossobject where _3 > 5`

    当某CSV行是张小，阿里巴巴，待入职时由于第三列不是整数类型，该行被跳过。
  - Json中某些Key类型不匹配SQL

    处理方法同上。比如：SQL为select s.name from ossobject s where s.aliren_age > 5

    当某Json节点是：{"Name":"张小", "aliren_age":待入职}，则该节点被跳过。

- JSON返回数据中Key的处理

  要说明的是SelectJson的输出一定是Json LINES。输出结果中的Key值按下面规则确定。

  - 当SQL是`select * from ossobject…`时，如果*返回的内容是一个JSON对象(指的是{…})，则原样返回；如果*不是一个Json对象（比如是字符串或者数组），则会使用一个DummyKey _1来表示。

    比如：数据为{“Age”:5}，`select * from ossobject.Age s where s = 5`，这时*对应的内容是5不是一个Json对象，故返回的内容为{“_1”:5}。但如果sql为`select * from ossobject s where s.Age = 5`，这时*对应的内容是{“Age”:5}，故原样返回。

  - 当SQL没有用select *，而是指定了列时，则返回内容的格式为`{“{列1}”:值，“{列2}”: 值…..}`。

    其中{列n}的生成方法如下：

    ■ 当select中指定了该列的Alias时用Alias。

    ■ 如果该列是一个Json对象的Key时，用该Key作为输出的Key值。

    ■ 如果该列是Json数组的某个元素时或者是聚合函数，用该列在输出中的序号（从1开始）加上前缀_作为输出值的key。

    比如：数据为{” contacts”:{“Age”:35，“Children”:[“child1”，“child2”,” child3”]}，sql为`select s.contacts.Age, s.contacts.Children[0] from ossobjects`，则输出为{“Age”:35，“_2”:” child1”}（因为Age是输入Json对象的Key，但Children[0]是数组Children的第一个元素，其在输出中是第二列，故为”_2”:“child1”）。

    ■ 如果指定了Alias，select s.contacts.Age, s.contacts.Children[0] as firstChild from ossobject，则输出为{“Age”:35，“firstChild”:” child1”}。

    ■ 当SQL为 select max(cast(s.Age as int)) from ossobject.contacts s时，输出为{“_1”:35}，因为该列是聚合函数，故而用该列在输出中的序号加上前缀_1表示Key。

   说明:
  Json文件和SQL中Key的匹配是大小写敏感的，比如select s.Age 和select s.age是不同的。

CreateSelectObjectMeta

CreateSelectObjectMeta API用于获取目标文件总的行数，总的列数（对于CSV文件），以及Splits个数。如果该信息不存在，则会扫描整个文件，分析并记录下CSV文件的上述信息。重复调

用该API则会利用到上述信息而不必重新扫描整个文件。如果该API执行正确，返回200。如果目标文件不是合法CSV或者JSON LINES文件，或者指定的CSV分隔符和目标CSV不匹配，则返回400。

说明:

CreateSelectObjectMeta操作要求用户对该Object有写权限。

- 请求语法
  - 请求语法（Csv）

```
POST  /samplecsv?x-oss-process=csv/meta
<CsvMetaRequest>
 <InputSerialization>
  <CompressionType>None</CompressionType>
  <CSV>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>
   <QuoteCharacter>base64 encode</QuoteCharacter>
  </CSV>
 </InputSerialization>
 <OverwriteIfExists>false|true</OverwriteIfExists>
</CsvMetaRequest>
```

  - 请求语法（Json）

```
POST  /samplecsv?x-oss-process=json/meta
<JsonMetaRequest>
 <InputSerialization>
  <CompressionType>None</CompressionType>
  <JSON>
   <Type>LINES</Type>
  </JSON>
 </InputSerialization>
 <OverwriteIfExists>false|true</OverwriteIfExists>
</JsonMetaRequest>
```

- 请求元素

名称	类型	描述
CsvMetaRequest	容器	保存创建Select csv Meta请求的容器。 子节点：InputSerialization 父节点：None
JsonMetaRequest	容器	保存创建 Select json meta请求的容器。 子节点：InputSerialization 父节点：None

名称	类型	描述
InputSerialization	容器	输入序列化参数（可选） 子节点：CompressionType、 CSV、JSON 父节点：CsvMetaRequest、JsonMetaRequest
OverwriteIfExists	bool	重新计算SelectMeta，覆盖已有数据。（可选，默认是false，即如果Select Meta已存在则直接返回） 子节点：None 父节点：CsvMetaRequest、JsonMetaRequest
CompressionType	枚举	指定文件压缩类型（可选）。目前不支持任何压缩，故只能为None。 子节点： None 父节点：InputSerialization
RecordDelimiter	字符串	指定CSV换行符，以Base64编码。默认值为’\n’（可选）。未编码前的值最多为两个字符，以字符的ANSI值表示，比如在Java里用\n表示换行。 子节点：None 父节点：CSV
FieldDelimiter	字符串	指定CSV列分隔符，以Base64编码。默认值为,（可选） 未编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用,表示逗号。 子节点：None 父节点：CSV （输入，输出）

名称	类型	描述
QuoteCharacter	字符串	指定CSV的引号字符，以Base64编码。默认值为\”（可选）。在CSV中引号内的换行符，列分隔符将被视作普通字符。为编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用\”表示引号。 子节点：None 父节点：CSV （输入）
CSV	容器	指定CSV输入格式。 子节点：RecordDelimiter，FieldDelimiter，QuoteCharacter 父节点：InputSerialization
JSON	容器	指定JSON输入格式。 子节点：Type 父节点：InputSerialization

名称	类型	描述
Type	枚举	指定Json类型。 合法值： LINES

Response Body：和SelectObject 类似，Create Meta API也以Frame的形式返回。有两种类型的Type：Continuous Frame以及End Meta Frame。Continuous Frame和SelectObject API完全一致。

名称	Frame-Type值	Payload格式	描述
Meta End Frame（Csv）	8388614	offset | total scanned bytes | status| splits count | rows count | columns count | error message <-8 bytes><------8 bytes ------><--4bytes><--4 bytes--><--8 bytes><--4 bytes---><variable size>	offset：8位整数，扫描结束时的文件偏移。 total scanned bytes：8位整数，最终扫描过的数据大小。 status：4位整数，最终的status splits_count：4位整数，总split个数。 rows_count：8位整数，总行数。 cols_count：4位整数，总列数。 error_message：详细的错误信息，若没有错误则为空。 Meta End Frame用来汇报Create Select Meta API最终的状态。

名称	Frame-Type值	Payload格式	描述
Meta End Frame（Json）	8388615	offset | total scanned bytes | status| splits count | rows count | error message <-8 bytes><------8 bytes------><--4bytes><--4 bytes--><--8 bytes><variable size>	offset：8位整数，扫描结束时的文件偏移。 total scanned bytes：8位整数，最终扫描过的数据大小。 status：4位整数，最终的status splits_count：4位整数，总split个数。 rows_count：8位整数，总行数。 error_message：详细的错误信息，若没有错误则为空。 Meta End Frame用来汇报Create Select Meta API最终的状态。

Response Header：无专门header。

- 样例请求

  - 样例请求（Csv）

```
POST /oss-select/bigcsv_normal.csv?x-oss-process=csv%2Fmeta HTTP/1
.1
Date: Fri, 25 May 2018 23:06:41 GMT
Content-Type:
Authorization: OSS AccessKeySignature
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.
0.30319.42000)
Content-Length: 309
Expect: 100-continue
Connection: keep-alive
Host: Host
<?xml version="1.0"?>
<CsvMetaRequest>
 <InputSerialization>
  <CSV>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
  </CSV>
 </InputSerialization>
 <OverwriteIfExisting>false</OverwriteIfExisting>
```

```
</CsvMetaRequest>
```

- **样例请求（Json）**

```
POST /oss-select/sample.json?x-oss-process=json%2Fmeta HTTP/1.1
Date: Fri, 25 May 2018 23:06:41 GMT
Content-Type:
Authorization: OSS AccessKeySignature
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.
0.30319.42000)
Content-Length: 309
Expect: 100-continue
Connection: keep-alive
Host: Host
<?xml version="1.0"?>
<JsonMetaRequest>
 <InputSerialization>
  <JSON>
   <Type>LINES</Type>
  </JSON>
 </InputSerialization>
 <OverwriteIfExisting>false</OverwriteIfExisting>
</JsonMetaRequest>
```

支持的时间格式

自动识别是指用户无需指定日期格式就能把符合下面格式之一的字符串转成timestamp类型。比如cast('20121201' as timestamp)会被解析成2012年12月1日。

以下格式为自动识别的格式：

格式	说明
YYYYMMDD	年月日
YYYY/MM/DD	年/月/日
DD/MM/YYYY/	日/月/年
YYYY-MM-DD	年-月-日
DD-MM-YY	日-月-年
DD.MM.YY	日.月.年
HH:MM:SS.mss	小时:分钟:秒.毫秒
HH:MM:SS	小时:分钟:秒
HH MM SS mss	小时 分钟 秒 毫秒
HH.MM.SS.mss	小时.分钟.秒.毫秒
HHMM	小时分钟
HHMMSSmss	小时分钟秒毫秒
YYYYMMDD HH:MM:SS.mss	年月日 小时:分钟:秒.毫秒

格式	说明
YYYY/MM/DD HH:MM:SS.mss	年/月/日 小时:分钟:秒.毫秒
DD/MM/YYYY HH:MM:SS.mss	日/月/年 小时:分钟:秒.毫秒
YYYYMMDD HH:MM:SS	年月日 小时:分钟:秒
YYYY/MM/DD HH:MM:SS	年/月/日 小时:分钟:秒
DD/MM/YYYY HH:MM:SS	日/月/年 小时:分钟:秒
YYYY-MM-DD HH:MM:SS.mss	年-月-日 小时:分钟:秒.毫秒
DD-MM-YYYY HH:MM:SS.mss	日-月-年 小时:分钟:秒.毫秒
YYYY-MM-DD HH:MM:SS	年-月-日 小时:分钟:秒
YYYYMMDDTHH:MM:SS	年月日T小时:分钟:秒
YYYYMMDDTHH:MM:SS.mss	年月日T小时:分钟:秒.毫秒
DD-MM-YYYYTHH:MM:SS.mss	日-月-年T小时:分钟:秒.毫秒
DD-MM-YYYYTHH:MM:SS	日-月-年T小时:分钟:秒
YYYYMMDDTHHMM	年月日T小时分钟
YYYYMMDDTHHMMSS	年月日T小时分钟秒
YYYYMMDDTHHMMSSMSS	年月日T小时分钟秒毫秒
ISO8601-0	年-月-日T小时:分钟+小时:分钟，或者年-月-日T小时:分钟-小时:分钟 这里+表示时区在标准时间UTC前面，-表示在后面。注意这个格式可用ISO8601-0来表示
ISO8601-1	年-月-日T小时:分钟:秒+小时:分钟，或者年-月-日T小时:分钟-小时:分钟 这里+表示时区在标准时间UTC前面，-表示在后面. 这个格式可用ISO8601-1来表示
CommonLog	比如28/Feb/2017:12:30:51 +0700
RFC822	比如 Tue, 28 Feb 2017 12:30:51 GMT
?D/?M/YY	日/月/年 此处日月均可用1位或者2位表示
?D/?M/YY ?H:?M	日月年 小时:分钟, 此处日月分钟小时均可用1位或者2位表示
?D/?M/YY ?H:?M:?S	日月年 小时:分钟:秒, 此处日月秒分钟小时均可用1位或者2位表示

由于以下格式会引起歧义，故用户需要指定格式，比如cast('20121201' as timestamp format 'YYYYDDMM')会将20121201解析成2012年1月12日。

格式	说明
YYYYDDMM	年日月
YYYY/DD/MM	年日月
MM/DD/YYYY	月/日/年
YYYY-DD-MM	年-日-月
MM-DD-YYYY	月-日-年
MM.DD.YYYY	月.日.年

ErrorCode

Select有两种形式返回ErrorCode。

- 一种是和其他OSS请求一样，在http响应Headers里返回http status code，并在body里返回Error信息。这种错误一般是有明显的输入错误时发生（比如输入的SQL不正确），或者有明显数据错误。
- 另一种ErrorCode是在响应Body的End Frame里返回，这种情况一般是Select处理数据的过程中发现数据不正确或者和SQL不匹配（比如在某一行里原本是整数列现在是一个字符串，且SQL里指定了该列是整数），这个时候往往一部分数据已经处理完毕并已发送到客户端且Http status code 仍然是206。

在下表的ErrorCode中，有些ErrorCode（比如InvalidCSVLine）可能以http status code的形式返回，也可能以在End Frame中的status code返回，这取决于发生错误的CSV 行出现在文件的什么位置。

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
InvalidSql Parameter	非法的SQL参数。 请求中SQL为空或者SQL长度超过最大值或者SQL不是以base64编码。	400	无

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
InvalidInputFieldDelimiter	非法的CSV列分隔符（输入）。 该参数不是以Base64编码或者解码后的长度大于1。	400	无
InvalidInputRecordDelimiter	非法的CSV行分隔符(输入)。该参数不是以Base64编码或者解码后长度大于2。	400	无
InvalidInputQuote	非法的CSV Quote字符(输入)。该参数不是以Base64编码或者解码后长度超过1。	400	无
InvalidOutputFieldDelimiter	非法的CSV列分隔符（输出）。该参数不是以Base64编码或者解码后的长度大于1。	400	无
InvalidOutputRecordDelimiter	非法的CSV行分隔符(输出)。该参数不是以Base64编码或者解码后长度大于2。	400	无
UnsupportedCompressionFormat	非法的Compression参数。该参数值不是NONE或者GZIP（此处大小写无关）。	400	无
InvalidCommentCharacter	非法的CSV注释字符。该参数值不是以Base64编码或者解码后的值超过1。	400	无
InvalidRange	非法的Range参数。该参数不是以line-range=或者split-range=作为前缀，或者range值本身不符合http range规范。	400	无
DecompressFailure	Compression参数是GZIP且解压失败。	400	无
InvalidMaxSkippedRecordsAllowed	MaxSkippedRecordsAllowed参数不是整数。	400	无
SelectCsvMetaUnavailable	指定了Range参数，但目标对象没有CSV meta时，此时应该先调用Create Select object MetaAPI。	400	无
InvalidTextEncoding	对象不是以UTF-8编码。	400	无
InvalidOSSSelectParameters	同时指定了EnablePayloadCrc为True以及OutputRawData为True时，两个参数冲突，返回此错误。	400	无

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
InternalError	OSS系统出现错误。	500或者206	无或者500
SqlSyntaxError	Base64解码后的SQL语法错误。	400	无
SqlExceedsMaxInCount	SQL中IN语句包含的值数目超过1024。	400	无
SqlExceedsMaxColumnNameLength	SQL中列名超过最大长度时1024。	400	无
SqlInvalidColumnIndex	SQL中列Index小于1或者大于1000。	400	无
SqlAggregationOnNonNumericType	SQL中聚合函数应用在非数值列中。	400	无
SqlInvalidAggregationOnTimestamp	SQL中SUM/AVG聚合应用在日期类型列。	400	无
SqlValueTypeOfInMustBeSame	SQL中IN语句包含不同类型的值。	400	无
SqlInvalidEscapeChar	SQL LIKE语句中Escape的字符为?或%或*。	400	无
SqlOnlyOneEscapeCharIsAllowed	SQL LIKE语句中Escape字符长度大于1。	400	无
SqlNoCharAfterEscapeChar	SQL LIKE语句中Escape字符后面没有被Escape字符。	400	无
SqlInvalidLimitValue	SQL Limit语句后的数字小于1。	400	无
SqlExceedsMaxWildCardCount	SQL LIKE语句中通配符*或者%的个数超过最大值。	400	无
SqlExceedsMaxConditionCount	SQL Where语句中条件表达式个数超过最大值。	400	无

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
SqlExceeds MaxConditionDepth	SQL Where语句中条件树的深度超过最大值。	400	无
SqlOneColumnCastToDifferentTypes	SQL语句中同一列被Cast成不同类型。	400	无
SqlOperationAppliedToDifferentTypes	SQL条件语句中一个操作符应用在两个不同类型的对象。比如_col1 > 3如果其中col1是字符串时就会报这个错误。	400	无
SqlInvalidColumnName	SQL语句中使用列名且该列名并不在CSV文件Header中出现。	400	无
SqlNotSupportedTimestampFormat	SQL CAST语句中指定了时间戳格式，且该格式不支持。	400	无
SqlNotMatchTimestampFormat	SQL CAST语句中指定了时间戳格式，但该格式和提供的时间戳字符串不匹配。	400	无
SqlInvalidTimestampValue	SQL CAST语句中未指定时间戳格式，且提供的时间戳字符串无法cast成时间戳类型。	400	无
SqlInvalidLikeOperand	SQL LIKE语句中左边不是列名或者列Index或者左边的列不是字符串类型，或者右边字符串。	400	无
SqlInvalidMixOfAggregationAndColumn	SQL Select语句中同时包含聚合操作以及非聚合操作的列名、列Index。	400	无
SqlExceedsMaxAggregationCount	SQL Select语句中包含的聚合操作超过最大值。	400	无
SqlInvalidMixOfStarAndColumn	SQL Select语句中同时包含*以及列名、列Index。	400	无
SqlInvalidKeepAllColumnsWithAggregation	SQL包含聚合操作，同时KeepAllColumns参数为True。	400	无

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
SqlInvalidKeepAllColumnsWithDuplicateColumn	SQL包含重复的列名、列索引，同时KeepAllColumns参数为True。	400	无
SqlInvalidSqlAfterAnalysis	SQL过分复杂导致分析后无法支持。	400	无
InvalidArithmeticOperand	SQL中算术操作应用在非数字类型的常量或者列。	400	无
SqlInvalidAndOperand	SQL中AND操作连接的两个表达式不是bool类型。	400	无
SqlInvalidOrOperand	SQL中OR操作连接的两个表达式不是bool类型。	400	无
SqlInvalidNotOperand	SQL中NOT操作应用在非bool类型的表达式。	400	无
SqlInvalidIsNullOperand	SQL中IS NULL应用在常量上。	400	无
SqlCompareOperandTypeMismatch	SQL中比较操作应用在两个不同类型的对象。	400	无
SqlInvalidConcatOperand	SQL中字符串连接操作（||）连接两个常量。	400	无
SqlUnsupportedSql	SQL过分复杂从而生成的SQL计划超过最大值。	400	无
HeaderInfoExceedsMaxSize	SQL指定了输出Header信息，且Header信息超过最大允许的长度。	400	无
OutputExceedsMaxSize	SQL的输出有放大导致输出结果中单行超过最大长度。	400	无
InvalidCsvLine	CSV行是非法时（包括长度超过最大允许的长度），或者被跳过的行数超过MaxSkippedRecordsAllowed。	206或者400	400或者无
NegativeRowIndex	SQL中作为数组Index的值为负数。	400	无

ErrorCode	描述	HTTP Status Code	Http Status Code in End Frame
ExceedsMaxNestedColumnDepth	SQL中的Json属性嵌套层数超过最大值。	400	无
NestedColumnNotSupportInCsv	嵌套的属性（含有数组[]或者.）不支持在CSV的SQL中。	400	无
TableRootNodeOnlySupportInJson	只有Json文件支持在From ossobject后指定根节点路径。	400	无
JsonNodeExceedsMaxSize	JSON文件的SQL的根节点大小超过允许最大值。	400或者206	无或者400
InvalidJsonData	非法的Json数据（格式不正确）。	400或者206	无或者400
ExceedsMaxJsonArraySize	JSON文件的SQL的根节点中的某个数组的元素个数超过最大值。	400或者206	无或者400
WildCardNotAllowed	JSON文件的SQL中wild card *不能出现select或者where的json属性中。比如 `select s.a.b[*] from ossobject where a.c[*] > 0`是不支持的。	400	无
JsonNodeExceedsMaxDepth	Json文件的SQL根节点深度超过最大值。	400或者206	无或者400

7.17 PutObjectTagging

您可以通过PutObjectTagging接口设置或更新对象的标签（Object Tagging）。

请求语法

```
PUT /objectname?tagging
Content-Length: 114
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
<Tagging>
  <TagSet>
    <Tag>
      <Key>Key</Key>
      <Value>Value</Value>
```

```
    </Tag>
  </TagSet>
</Tagging>
```

请求元素

名称	类型	是否必需	描述
Tagging	容器	是	子节点：TagSet
TagSet	容器	是	父节点：Tagging 子节点：Tag
Tag	容器	否	父节点：TagSet 子节点：Key，Value
Key	字符串	否	父节点：Tag 子节点：无
Value	字符串	否	父节点：Tag 子节点：无

细节分析

- 请求者需要有PutObjectTagging权限。
- 更改Tagging#会更新Object Last-Modified时间。
- 标签合法字符集包括大小写字母、数字、空格和以下符号：

  +-=._:/

示#

- 请求示例

```
PUT /objectname?tagging
Content-Length: 114
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Mon, 18 Mar 2019 08:25:17 GMT
Authorization: OSS ***********:**********************
<Tagging>
  <TagSet>
    <Tag>
      <Key>a</Key>
      <Value>1</Value>
    </Tag>
    <Tag>
      <Key>b</Key>
```

```
      <Value>2</Value>
    </Tag>
  </TagSet>
</Tagging>
```

- 返回示例

```
200 (OK)
content-length: 0
server: AliyunOSS
x-oss-request-id: 5C8F55ED461FB4A64C000004
date: Mon, 18 Mar 2019 08:25:17 GMT
```

7.18 GetObjectTagging

您可以通过GetObjectTagging接口获取对象的标签信息。

请求语法

```
GET /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
Tagging	容器	子节点：TagSet
TagSet	容器	父节点：Tagging 子节点：Tag
Tag	容器	父节点：TagSet 子节点：Key，Value
Key	字符串	父节点：Tag 子节点：无
Value	字符串	父节点：Tag 子节点：无

示例

- 请求示例

```
GET /objectname?tagging
```

```
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 20 Mar 2019 02:02:36 GMT
Authorization: OSS ************:********************
```

- 响应示例

```
200 (OK)
content-length: 209
server: AliyunOSS
x-oss-request-id: 5C919F38461FB42826000002
date: Wed, 20 Mar 2019 02:02:32 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<Tagging>
  <TagSet>
    <Tag>
      <Key>a</Key>
      <Value>1</Value>
    </Tag>
    <Tag>
      <Key>b</Key>
      <Value>2</Value>
    </Tag>
  </TagSet>
</Tagging>
```

7.19 DeleteObjectTagging

您可以通过DeleteObjectTagging删除指定对象的标签。

请求语法

```
DELETE /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

- 请求示例

```
DELETE /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 09 Apr 2019 03:00:33 GMT
Authorization: OSS LTAIbsTkySSptaz****/Zr0o6BKgAl7iiBtHN2JMC****
```

- 响应示例

```
204 (No Content)
content-length: 0
server: AliyunOSS
x-oss-request-id: 5CAC0AD16D0232E2051B****
date: Tue, 09 Apr 2019 03:00:33 GMT
```

8 关于MultipartUpload的操作

8.1 简介

除了通过PUT Object接口上传文件到OSS以外，OSS还提供了另外一种上传模式——Multipart Upload。

您可以在如下的应用场景内（但不仅限于此）使用Multipart Upload上传模式：

- 需要支持断点上传。
- 上传超过100MB大小的文件。
- 网络条件较差，和OSS的服务器之间的链接经常断开。
- 上传文件之前，无法确定上传文件的大小。

8.2 InitiateMultipartUpload

使用Multipart Upload模式传输数据前，必须先调用该接口来通知OSS初始化一个Multipart Upload事件。

说明：

- 该接口会返回一个OSS服务器创建的全局唯一的Upload ID，用于标识本次Multipart Upload事件。您可以根据这个ID来发起相关的操作，如中止Multipart Upload、查询Multipart Upload等。
- 初始化Multipart Upload请求，并不会影响已存在的同名Object。
- 该操作计算认证签名时，需要加“?uploads”到CanonicalizedResource中。

请求语法

```
POST /ObjectName?uploads HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT date
Authorization: SignatureValue
```

请求参数

Initiate Multipart Upload时，可以通过encoding-type对返回结果中的Key进行编码。

名称	类型	描述
encoding-type	字符串	指定对返回的Key进行编码，目前支持url编码。Key使用UTF-8字符，但xml 1.0标准不支持解析一些控制字符，比如ascii值从0到10的字符。对于Key中包含xml 1.0标准不支持的控制字符，可以通过指定encoding-type对返回的Key进行编码。 默认值：无 可选值：url

请求头

说明:

初始化Multipart Upload请求，支持如下标准的HTTP请求头：Cache-Control、Content-Disposition、Content-Encoding、Content-Type、Expires，以及以x-oss-meta-开头的用户自定义Headers。具体含义请参见PutObject。

名称	类型	描述
Cache-Control	字符串	指定该Object被下载时的网页的缓存行为；更详细描述请参照RFC2616。 默认值：无
Content-Disposition	字符串	指定该Object被下载时的名称；更详细描述请参照RFC2616。 默认值：无
Content-Encoding	字符串	指定该Object被下载时的内容编码格式；更详细描述请参照RFC2616。 默认值：无
Expires	整数	过期时间（milliseconds）；更详细描述请参照RFC2616。 默认值：无

名称	类型	描述
x-oss-server-side-encryption	字符串	指定上传该Object每个part时使用的服务器端加密编码算法，OSS会对上传的每个part采用服务器端加密编码进行存储。 合法值：AES256 或 KMS 说明: 您需要在控制台开通KMS（密钥管理服务）后，才可使用KMS加密算法。
x-oss-server-side-encryption-key-id	字符串	表示KMS托管的用户主密钥。 该参数在x-oss-server-side-encryption为KMS时有效。
x-oss-storage-class	字符串	指定Object的存储类型。 取值： · Standard · IA · Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。 说明: · 如果StorageClass的值不合法，返回400 错误。错误码：InvalidArgumet。 · 对于任意存储类型Bucket，若上传Object时指定该值，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。
x-oss-tagging	字符串	指定Object的标签，可同时设置多个标签，例如：TagA=A&TagB=B 说明: Key和Value需要先进行URL编码，如果某项没有"="，则看作Value为空字符串。

响应元素

> **说明:**
> 服务器收到初始化Multipart Upload请求后，会返回一个XML格式的请求体。该请求体内包含Bucket、Key和UploadID三个元素。

名称	类型	描述
Bucket	字符串	初始化一个Multipart Upload事件的Bucket名称。 父节点：InitiateMultipartUploadResult
InitiateMultipartUploadResult	容器	保存Initiate Multipart Upload请求结果的容器。 子节点：Bucket, Key, UploadId 父节点：None
Key	字符串	初始化一个Multipart Upload事件的Object名称。 父节点：InitiateMultipartUploadResult
UploadId	字符串	唯一标示此次Multipart Upload事件的ID。 父节点：InitiateMultipartUploadResult **说明:** 请记录下其中的UploadID，以用于后续的Multipart相关操作。
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，那返回的结果会对Key进行编码。 父节点：容器

示例

请求示例

```
POST /multipart.data?uploads HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
x-oss-storage-class: Archive
```

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:/cluRFtRwMTZpC2hTj4F67AG
****
```

返回示例

```
HTTP/1.1 200 OK
Content-Length: 230
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 42c25703-7503-fbd8-670a-bda01eae****
Date: Wed, 22 Feb 2012 08:32:21 GMT
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns=”http://doc.oss-cn-hangzhou.
aliyuncs.com”>
    <Bucket> multipart_upload</Bucket>
    <Key>multipart.data</Key>
    <UploadId>0004B9894A22E5B1888A1E29F823****</UploadId>
</InitiateMultipartUploadResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++
- PHP
- C
- .NET
- Node.js

错误码

错误码	HTTP 状态码	描述
InvalidEncryptionAlgorithmError	400	指定OSS服务器端除AES256和KMS以外的加密方式
InvalidArgument	400	在上传每个part时继续添加x-oss-server-side-encryption 请求头
KmsServiceNotEnabled	403	使用KMS加密算法时，没有在控制台开通KMS（密钥管理服务）

8.3 UploadPart

初始化一个MultipartUpload之后，可以根据指定的Object名和Upload ID来分块（Part）上传数据。

说明:

- 调用该接口上传Part数据前，必须先调用InitiateMultipartUpload接口来获取一个OSS服务器颁发的Upload ID。对于同一个Upload ID，该号码不但唯一标识这一块数据，也标识了这块数据在整个文件内的相对位置。
- 每一个上传的Part都有一个标识它的号码（part number，范围是1-10000），单个Part大小范围100KB-5GB。MultipartUpload要求除最后一个Part以外，其他的Part大小都要大于100KB。因不确定是否为最后一个Part，UploadPart接口并不会立即校验上传Part的大小，只有当CompleteMultipartUpload的时候才会校验。
- 如果你用同一个part number上传了新的数据，那么OSS上已有的这个号码的Part数据将被覆盖。
- OSS会将服务器端收到Part数据的MD5值放在ETag头内返回给用户。
- 若调用InitiateMultipartUpload接口时，指定了x-oss-server-side-encryption请求头，则会对上传的Part进行加密编码，并在UploadPart响应头中返回x-oss-server-side-encryption头，其值表明该Part的服务器端加密算法，详情请参考InitiateMultipartUpload。

请求语法

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Content-Length: Size
Authorization: SignatureValue
```

示例

请求示例

```
PUT /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E36  HTTP/
1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMmUm
****
```

```
[6291456 bytes data]
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
ETag: 7265F4D211B56873A381D321F586****
x-oss-request-id: 3e6aba62-1eae-d246-6118-8ff42cd0****
Date: Wed, 22 Feb 2012 08:32:21 GMT
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++
- PHP
- C
- .NET
- Node.js

错误码

错误码	HTTP 状态码	描述
InvalidArgument	400	超出Part number范围（1-10000）
InvalidDigest	400	为了保证数据在网络传输过程中不出现错误，用户发送请求时可以携带Content-MD5，OSS计算上传数据的MD5与用户上传的MD5值不一致

8.4 UploadPartCopy

UploadPartCopy通过从一个已存在的Object中拷贝数据来上传一个Part。

通过在UploadPart请求的基础上增加一个Header:x-oss-copy-source来调用该接口。当拷贝一个大于1GB的文件时，必须使用UploadPartCopy的方式进行拷贝。如果想通过单个操作拷贝小于1GB的文件，可以参考CopyObject。

说明:

- 该操作不能拷贝以AppendObject方式上传的Object。

- UploadPartCopy的源Bucket地址和目标Bucket地址必须是同一个Region。
- 调用该接口上传Part数据前，必须先调用InitiateMultipartUpload接口来获取一个OSS服务器颁发的Upload ID。
- 若调用InitiateMultipartUpload接口时，指定了x-oss-server-side-encryption请求头，则会对上传的Part进行加密编码，并在UploadPart响应头中返回x-oss-server-side-encryption头，其值表明该Part的服务器端加密算法，详情请参考InitiateMultipartUpload。
- MultipartUpload要求除最后一个Part以外，其他的Part大小都要大于100KB。因不确定是否为最后一个Part，UploadPart接口并不会立即校验上传Part的大小，只有当CompleteMultipartUpload的时候才会校验。

请求语法

```
PUT /ObjectName? partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Content-Length: Size
Authorization: SignatureValue
x-oss-copy-source: /SourceBucketName/SourceObjectName
x-oss-copy-source-range:bytes=first-last
```

请求头

除了通用的请求头，UploadPartCopy请求中通过下述请求头指定拷贝的源Object地址和拷贝的范围。

名称	类型	描述
x-oss-copy-source	字符串	拷贝源地址（必须有可读权限） 默认值：无

名称	类型	描述
x-oss-copy-source-range	整型	源Object的拷贝范围。如，设定 bytes=0-9，表示传送第0到第9这10个字符。 当拷贝整个源Object时不需要该请求Header。 默认值：无 说明: 不指定x-oss-copy-source-range请求头时，表示拷贝整个源Object。当指定该请求头时，则返回消息中会包含整个文件的长度和此次拷贝的范围，例如：Content-Range: bytes 0-9/44，表示整个文件长度为44，此次拷贝的范围为0-9。当指定的范围不符合范围规范时，则拷贝整个文件，并且不在结果中提及Content-Range。

下述请求Header作用于x-oss-copy-source指定的源Object。

名称	类型	描述
x-oss-copy-source-if-match	字符串	如果源Object的ETAG值和用户提供的ETAG相等，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-none-match	字符串	如果传入的ETag值和Object的ETag不匹配，则正常传输文件，并返回200 OK；否则返回304 Not Modified 默认值：无

名称	类型	描述
x-oss-copy-source-if-unmodified-since	字符串	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误。 默认值：无
x-oss-copy-source-if-modified-since	字符串	如果指定的时间早于实际修改时间，则正常传送文件，并返回200 OK；否则返回304 not modified 默认值：无 时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT

示例

- 请求示例

```
PUT /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E36
HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMmUm
****
x-oss-copy-source: /oss-example/ src-object
x-oss-copy-source-range:bytes=100-6291756
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 3e6aba62-1eae-d246-6118-8ff42cd0****
Date: Thu, 17 Jul 2014 06:27:54 GMT'
<?xml version="1.0" encoding="UTF-8"?>
<CopyPartResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <LastModified>2014-07-17T06:27:54.000Z </LastModified>
    <ETag>"5B3C1A2E053D763E1B002CC607C5****"</ETag>
</CopyPartResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++
- PHP
- C
- .NET

错误码

错误码	HTTP 状态码	描述
OperationNotSupported	400	Bucket的类型为Archive时调用UploadPartCopy接口。

8.5 CompleteMultipartUpload

在将所有数据Part都上传完成后，必须调用`CompleteMultipartUpload`接口来完成整个文件的MultipartUpload。

在执行该操作时，用户必须提供所有有效的数据的Part列表（包括Part号码和ETag）。OSS收到用户提交的Part列表后，会逐一验证每个数据Part的有效性。当所有的数据Part验证通过后，OSS将把这些数据part组合成一个完整的Object。

说明:

- CompleteMultipartUpload时会确认除最后一块以外所有块的大小是否都大于100KB，并检查用户提交的Part列表中的每一个Part号码和Etag。所以在上传Part时，客户端除了需要记录Part号码外，还需要记录每次上传Part成功后服务器返回的ETag值。
- 由于OSS处理CompleteMultipartUpload请求时会持续一定的时间。在这段时间内，如果客户端与OSS之间连接中断，OSS仍会继续将该请求。
- 用户提交的Part列表中，Part号码可以不连续。例如第一块的Part号码是1，第二块的Part号码是5。
- OSS处理CompleteMultipartUpload请求成功后，该Upload ID就会变成无效。
- 同一个Object可以同时拥有不同的Upload ID，当Complete一个Upload ID后，该Object的其他Upload ID不受影响。
- 若调用InitiateMultipartUpload接口时，指定了x-oss-server-side-encryption请求头，则在CompleteMultipartUpload的响应头中返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。

请求语法

```
POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Content-Length: Size
Authorization: Signature
<CompleteMultipartUpload>
<Part>
<PartNumber>PartNumber</PartNumber>
<ETag>ETag</ETag>
</Part>
...
</CompleteMultipartUpload>
```

请求参数(Request Parameters)

Complete Multipart Upload时，可以通过Encoding-type对返回结果中的Key进行编码。

名称	类型	描述
Encoding-type	字符串	指定对返回的Key进行编码，目前支持url编码。Key使用UTF-8字符，但xml 1.0标准不支持解析一些控制字符，比如ascii值从0到10的字符。对于Key中包含xml 1.0标准不支持的控制字符，可以通过指定encoding-type对返回的Key进行编码。 默认值：无 可选值：url

请求元素

名称	类型	描述
CompleteMultipartUpload	容器	保存Complete Multipart Upload请求内容的容器。 子节点：一个或多个Part元素 父节点：无
ETag	字符串	Part成功上传后，OSS返回的ETag值。 父节点：Part

名称	类型	描述
Part	容器	保存已经上传Part信息的容器。 子节点：ETag, PartNumber 父节点：InitiateMultipartUploadResult
PartNumber	整数	Part数目。 父节点：Part

响应元素

名称	类型	描述
Bucket	字符串	Bucket名称。 父节点：CompleteMultipartUploadResult
CompleteMultipartUploadResult	容器	保存Complete Multipart Upload请求结果的容器。 子节点：Bucket, Key, ETag, Location 父节点：None
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。Complete Multipart Upload请求创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。 父节点：CompleteMultipartUploadResult
Location	字符串	新创建Object的URL。 父节点：CompleteMultipartUploadResult

名称	类型	描述
Key	字符串	新创建Object的名字。 父节点：CompleteMultipartUploadResult
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，那返回的结果会对Key进行编码。 父节点：容器

示例

- **请求示例**

```
POST /multipart.data? uploadId=0004B9B2D2F7815C432C9057C03134D4
HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 1056
Date: Fri, 24 Feb 2012 10:19:18 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:8VwFhFUWmVecK6jQlHlXMK/z
****
<CompleteMultipartUpload>
    <Part>
        <PartNumber>1</PartNumber>
        <ETag>"3349DC700140D7F86A0784842780****"</ETag>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <ETag>"8EFDA8BE206636A695359836FE0A****"</ETag>
    </Part>
    <Part>
        <PartNumber>8</PartNumber>
        <ETag>"8C3150651671324441774 11FDA14****"</ETag>
    </Part>
</CompleteMultipartUpload>
```

返回示例

```
HTTP/1.1 200 OK
Server: AliyunOSS
Content-Length: 329
Content-Type: Application/xml
Connection: keep-alive
x-oss-request-id: 594f0751-3b1e-168f-4501-4ac71d21****
Date: Fri, 24 Feb 2012 10:19:18 GMT
<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns=”http://doc.oss-cn-hangzhou.
aliyuncs.com”>
    <Location>http://oss-example.oss-cn-hangzhou.aliyuncs.com /
multipart.data</Location>
    <Bucket>oss-example</Bucket>
    <Key>multipart.data</Key>
    <ETag>B864DB6A936D376F9F8D3ED3BBE540****</ETag>
```

```
</CompleteMultipartUploadResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- Go
- C++
- PHP
- C
- .NET

错误码

错误码	HTTP 状态码	描述
InvalidDigest	400	为了保证数据在网络传输过程中不出现错误，用户发送请求时可以携带Content-MD5，OSS计算上传数据的MD5与用户上传的MD5值不一致

8.6 AbortMultipartUpload

AbortMultipartUpload接口用于终止MultipartUpload事件。您需要提供MultipartUpload事件相应的Upload ID。

> **说明：**
>
> - 当一个MultipartUpload事件被中止后，您无法再使用这个Upload ID做任何操作，已经上传的Part数据也会被删除。
> - 中止一个MultipartUpload事件时，如果其所属的某些Part仍然在上传，那么这次中止操作将无法删除这些Part。所以如果存在并发访问的情况，需要调用几次AbortMultipartUpload接口以彻底释放OSS上的空间。

请求语法

```
DELETE /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
```

```
Authorization: Signature
```

示例

请求示例

```
Delete /multipart.data?&uploadId=0004B9895DBBB6EC98E  HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMm
UmWjI=
```

返回示例

```
HTTP/1.1 204
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 059a22ba-6ba9-daed-5f3a-e48027df344d
Date: Wed, 22 Feb 2012 08:32:21 GMT
```

SDK

此接口所对应的各语言SDK如下：

- Java
- PHP
- Go
- C
- .NET

错误码

错误码	HTTP 状态码	描述
NoSuchUpload	404	Upload ID不存在。

8.7 ListMultipartUploads

ListMultipartUploads用来列举所有执行中的Multipart Upload事件，即已经初始化但还未Complete或者Abort的Multipart Upload事件。

如果想指定OSS返回列举结果内Multipart Upload信息的数目，可以在请求中添加max-uploads参数。另外，OSS返回列举结果中的IsTruncated元素标明是否还有其他的Multipart Upload。

说明：

- max-uploads最大值为1000，表明OSS返回的列举结果中最多包含1000个Multipart Upload信息。
- OSS的返回结果按照Object名字字典序升序排列；对于同一个Object，则按照时间序升序排列。

请求语法

```
Get /?uploads HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: Signature
```

请求参数

说明:

ListMultipartUploads请求支持prefix、marker、delimiter、upload-id-marker和max-uploads5种请求参数。通过这些参数的组合，可以设定查询Multipart Uploads事件的规则，获得期望的查询结果。

名称	类型	描述
delimiter	字符串	是一个用于对Object名字进行分组的字符。所有名字包含指定的前缀且第一次出现delimiter字符之间的object作为一组元素——CommonPrefixes。
max-uploads	字符串	限定此次返回Multipart Uploads事件的最大数目，如果不设定，默认为1000，max-uploads取值不能大于1000。

名称	类型	描述
key-marker	字符串	与upload-id-marker参数一同使用来指定返回结果的起始位置。 · 如果upload-id-marker参数未设置，查询结果中包含所有Object名字的字典序大于key-marker参数值的Multipart事件。 · 如果upload-id-marker参数被设置，查询结果中包含： - 所有Object名字的字典序大于key-marker参数值的Multipart事件。 - Object名字等于key-marker参数值，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。
prefix	字符串	限定返回的object key必须以prefix作为前缀。注意使用prefix查询时，返回的key中仍会包含prefix。 说明: 您可以灵活地使用prefix参数对bucket内的object进行分组管理（类似文件夹功能）。

名称	类型	描述
upload-id-marker	字符串	与key-marker参数一同使用来指定返回结果的起始位置。 · 如果key-marker参数未设置，则OSS忽略upload-id-marker参数。 · 如果key-marker参数被设置，查询结果中包含： - 所有Object名字的字典序大于key-marker参数值的Multipart事件。 - Object名字等于key-marker参数值，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。
encoding-type	字符串	指定对返回的内容进行编码，指定编码的类型。Delimiter、KeyMarker、Prefix、Ne 用UTF-8字符，但xml 1.0标准不支持解析一些控制字符，比如ascii值从0到10的字符。对于包含xml 1.0标准不支持的控制字符，可以通过指定encoding-type对返回的Delimiter、KeyMarker、Prefix、Next 行编码。 默认值：无

响应元素

名称	类型	描述
ListMultipartUploadsResult	容器	保存ListMultipartUpload请求结果的容器。 子节点：Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MasUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload 父节点：None
Bucket	字符串	Bucket名称。 父节点：ListMultipartUploadsResult
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，那返回的结果会对Delimiter、KeyMarker、Prefix、Next些元素进行编码。 父节点：ListMultipartUploadsResult
KeyMarker	字符串	列表的起始Object位置。 父节点：ListMultipartUploadsResult
UploadIdMarker	字符串	列表的起始UploadID位置。 父节点：ListMultipartUploadsResult

名称	类型	描述
NextKeyMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextKeyMarker元素，用于标明接下来请求的KeyMarker值。 父节点：ListMultipartUploadsResult
NextUploadMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextUploadMarker元素，用于标明接下来请求的UploadMarker值。 父节点：ListMultipartUploadsResult
MaxUploads	整数	返回的最大Upload数目。 父节点：ListMultipartUploadsResult
IsTruncated	枚举字符串	标明本次返回的MultipartUpload结果列表是否被截断。“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。 有效值：false、true 默认值：false 父节点：ListMultipartUploadsResult
Upload	容器	保存Multipart Upload事件信息的容器。 子节点：Key, UploadId, Initiated 父节点：ListMultipartUploadsResult

名称	类型	描述
Key	字符串	初始化Multipart Upload事件的Object名字。 父节点：Upload
UploadId	字符串	Multipart Upload事件的ID。 父节点：Upload
Initiated	日期	Multipart Upload事件初始化的时间。 父节点：Upload

示例

请求示例

```
Get /?uploads  HTTP/1.1
Host:oss-example. oss-cn-hangzhou.aliyuncs.com
Date: Thu, 23 Feb 2012 06:14:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:JX75CtQqsmBBz+dcivn7kwBM
****
```

返回示例

```
HTTP/1.1 200
Server: AliyunOSS
Connection: keep-alive
Content-length: 1839
Content-type: application/xml
x-oss-request-id: 58a41847-3d93-1905-20db-ba6f561c****
Date: Thu, 23 Feb 2012 06:14:27 GMT

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs
.com”>
    <Bucket>oss-example</Bucket>
    <KeyMarker></KeyMarker>
    <UploadIdMarker></UploadIdMarker>
    <NextKeyMarker>oss.avi</NextKeyMarker>
    <NextUploadIdMarker>0004B99B8E707874FC2D692FA5D77D3F</NextUpload
IdMarker>
    <Delimiter></Delimiter>
    <Prefix></Prefix>
    <MaxUploads>1000</MaxUploads>
    <IsTruncated>false</IsTruncated>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0004B999EF518A1FE585B0C9360DC4C8</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>multipart.data</Key>
```

```
        <UploadId>0004B999EF5A239BB9138C6227D6****</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>oss.avi</Key>
        <UploadId>0004B99B8E707874FC2D692FA5D7****</UploadId>
        <Initiated>2012-02-23T06:14:27.000Z</Initiated>
    </Upload>
</ListMultipartUploadsResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Go
- C++
- PHP
- .NET

8.8 ListParts

ListParts接口用于列举指定Upload ID所属的所有已经上传成功Part。

说明:

- OSS的返回结果按照Part号码升序排列。
- 由于网络传输可能出错，所以不推荐用ListParts的结果（Part Number和ETag值）来生成最后Complete Multipart的Part列表。

请求语法

```
Get  /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: Signature
```

请求参数(Request Parameters)

名称	类型	描述
uploadId	字符串	MultipartUpload事件的ID。 默认值：无

名称	类型	描述
max-parts	整数	规定在OSS响应中的最大Part数目。 默认值：1,000 最大值：1,000
part-number-marker	整数	指定List的起始位置，只有Part Number数目大于该参数的Part会被列出。 默认值：无
Encoding-type	字符串	指定对返回的内容进行编码，指定编码的类型。Key使用UTF-8字符，但xml 1.0标准不支持解析一些控制字符，比如ascii值从0到10的字符。对于Key中包含xml 1.0标准不支持的控制字符，可以通过指定Encoding-type对返回的Key进行编码。 默认值：无 可选值：url

响应元素(Response Elements)

名称	类型	描述
ListPartsResult	容器	保存List Part请求结果的容器。 子节点：Bucket, Key, UploadId, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part 父节点：无
Bucket	字符串	Bucket名称。 父节点：ListPartsResult

名称	类型	描述
EncodingType	字符串	指明对返回结果进行编码使用的类型。如果请求的参数中指定了Encoding-type，那会对返回结果中的Key进行编码。 父节点：ListPartsResult
Key	字符串	Object名称。 父节点：ListPartsResult
UploadId	字符串	Upload事件ID。 父节点：ListPartsResult
PartNumberMarker	整数	本次List结果的Part Number起始位置。 父节点：ListPartsResult
NextPartNumberMarker	整数	如果本次没有返回全部结果，响应请求中将包含NextPartNumberMarker元素，用于标明接下来请求的PartNumberMarker值。 父节点：ListPartsResult
MaxParts	整数	返回请求中最大的Part数目。 父节点：ListPartsResult
IsTruncated	枚举字符串	标明本次返回的ListParts结果列表是否被截断。“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。 有效值：true、false 父节点：ListPartsResult
Part	字符串	保存Part信息的容器 子节点：PartNumber, LastModified, ETag, Size 父节点：ListPartsResult

名称	类型	描述
PartNumber	整数	标示Part的数字。 父节点：ListPartsResult.Part
LastModified	日期	Part上传的时间。 父节点：ListPartsResult.part
ETag	字符串	已上传Part内容的ETag。 父节点：ListPartsResult.Part
Size	整数	已上传Part大小。 父节点：ListPartsResult.Part

示例

请求示例

```
Get  /multipart.data?uploadId=0004B999EF5A239BB9138C6227D69F95  HTTP/1
.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 23 Feb 2012 07:13:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:4qOnUMc9UQWqkz8wDqD3
lIsa9P8=
```

返回示例

```
HTTP/1.1 200
Server: AliyunOSS
Connection: keep-alive
Content-length: 1221
Content-type: application/xml
x-oss-request-id: 106452c8-10ff-812d-736e-c865294afc1c
Date: Thu, 23 Feb 2012 07:13:28 GMT

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <Bucket>multipart_upload</Bucket>
    <Key>multipart.data</Key>
    <UploadId>0004B999EF5A239BB9138C6227D69F95</UploadId>
    <NextPartNumberMarker>5</NextPartNumberMarker>
    <MaxParts>1000</MaxParts>
    <IsTruncated>false</IsTruncated>
    <Part>
        <PartNumber>1</PartNumber>
        <LastModified>2012-02-23T07:01:34.000Z</LastModified>
```

```
        <ETag>&quot;3349DC700140D7F86A078484278075A9&quot;</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>2</PartNumber>
        <LastModified>2012-02-23T07:01:12.000Z</LastModified>
        <ETag>&quot;3349DC700140D7F86A078484278075A9&quot;</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <LastModified>2012-02-23T07:02:03.000Z</LastModified>
        <ETag>&quot;7265F4D211B56873A381D321F586E4A9&quot;</ETag>
        <Size>1024</Size>
    </Part>
</ListPartsResult>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- PHP
- Go
- C
- .NET

9 跨域资源共享

9.1 简介

跨域资源共享（CORS）允许Web端的应用程序访问不属于本域的资源。

OSS支持CORS标准，方便您更灵活地开发Web应用程序。OSS提供了如下接口方便您控制跨域访问的各种权限：

- PutBucketCORS
- GetBucketCORS
- DeleteBucketCORS
- OptionObject

9.2 PutBucketCORS

PutBucketcors接口用于为指定的存储空间（Bucket）设定一个跨域资源共享（CORS）规则。如果Bucket已有CORS规则，使用此接口会覆盖原有规则。Bucket默认不开启CORS功能，所有跨域请求的Origin都不被允许。

说明：

- 在应用程序中使用CORS功能，需通过本接口手动上传CORS规则来开启CORS功能。例如，从www.a.com通过浏览器的XMLHttpRequest功能来访问OSS，需要通过本接口手动上传CORS规则，且CORS规则需由XML文档进行描述。
- 当OSS收到一个跨域请求或OPTIONS请求，会先读取Bucket对应的CORS规则，然后进行相应的权限检查。OSS会依次检查每一条规则，使用第一条匹配的规则来允许请求并返回对应的header。如果所有规则都匹配失败则不附加任何CORS相关的header。
- 每个Bucket的CORS设定是由多条CORS规则指定的。每个Bucket最多允许10条规则。上传的XML文档最大允许16 KB。
- CORS规则匹配成功必须满足以下三个条件：
  - 请求的Origin必须匹配一个AllowedOrigin项。
  - 请求的方法（如GET、PUT等）或者OPTIONS请求的Access-Control-Request-Method头对应的方法必须匹配一个AllowedMethod项。
  - OPTIONS请求的Access-Control-Request-Headers头包含的每个header都必须匹配一个AllowedHeader项。

请求语法

```
PUT /?cors HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
    <CORSRule>
      <AllowedOrigin>the origin you want allow CORS request from</
AllowedOrigin>
      <AllowedOrigin>…</AllowedOrigin>
      <AllowedMethod>HTTP method</AllowedMethod>
      <AllowedMethod>…</AllowedMethod>
        <AllowedHeader> headers that allowed browser to send</
AllowedHeader>
          <AllowedHeader>…</AllowedHeader>
          <ExposeHeader> headers in response that can access from
client app</ExposeHeader>
          <ExposeHeader>…</ExposeHeader>
          <MaxAgeSeconds>time to cache pre-fight response</MaxAgeSeco
nds>
    </CORSRule>
    <CORSRule>
      …
    </CORSRule>
…
</CORSConfiguration >
```

请求元素

名称	类型	是否必选	描述
CORSRule	容器	是	CORS规则的容器。 **说明:** 每个Bucket最多允许10条CORS规则。 父节点：CORSConfiguration
AllowedOrigin	字符串	是	指定允许的跨域请求的来源。 **说明:** · OSS允许使用多个元素来指定多个允许的来源。 · OSS允许使用最多一个星号（*）通配符。如果指定为星号（*），则表示允许所有的来源的跨域请求。 父节点：CORSRule

名称	类型	是否必选	描述
AllowedMethod	枚举（GET,PUT,DELETE,POST,HEAD）	是	指定允许的跨域请求方法。 父节点：CORSRule
AllowedHeader	字符串	否	控制在OPTIONS预取指令中Access-Control-Request-Headers头中指定的header是否允许。在Access-Control-Request-Headers中指定的每个header都必须在AllowedHeader中有一条对应的项。 说明: 允许最多使用一个星号（*）通配符 。 父节点：CORSRule
ExposeHeader	字符串	否	指定允许用户从应用程序中访问的响应头。例如，一个Javascript的XMLHttpRequest对象。 说明: 不允许使用星号（*）通配符。 父节点：CORSRule
MaxAgeSeconds	整型	否	指定浏览器对特定资源的预取（OPTIONS）请求返回结果的缓存时间。单位为秒。 说明: 一条CORS规则最多允许出现一个MaxAgeSeconds。 父节点：CORSRule
CORSConfiguration	容器	是	Bucket的CORS规则容器。 父节点：无

示例

请求示例

```
PUT /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrT
*****
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
    <CORSRule>
      <AllowedOrigin>*</AllowedOrigin>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedMethod>GET</AllowedMethod>
      <AllowedHeader>Authorization</AllowedHeader>
    </CORSRule>
    <CORSRule>
      <AllowedOrigin>http://www.a.com</AllowedOrigin>
      <AllowedOrigin>http://www.b.com</AllowedOrigin>
      <AllowedMethod>GET</AllowedMethod>
      <AllowedHeader> Authorization</AllowedHeader>
      <ExposeHeader>x-oss-test</ExposeHeader>
      <ExposeHeader>x-oss-test1</ExposeHeader>
      <MaxAgeSeconds>100</MaxAgeSeconds>
    </CORSRule>
</CORSConfiguration >
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 50519080C4689A033D0*****
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C++
- C
- .NET
- Ruby

错误码

错误码	HTTP 状态码	描述
InvalidDigest	400	上传了Content-MD5请求头后，OSS会计算消息体的Content-MD5并检查一致性，如果不一致会返回此错误码。

9.3 GetBucketCORS

GetBucketCORS接口用于获取指定存储空间（Bucket）当前的跨域资源共享（CORS）规则。

请求语法

```
GET /?cors HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

响应元素

名称	类型	描述
CORSRule	容器	CORS规则的容器。每个Bucket最多允许10条规则。 父节点：CORSConfiguration
AllowedOrigin	字符串	允许的跨域请求来源。 星号（*）表示允许所有的来源的跨域请求。 父节点：CORSRule
AllowedMethod	枚举（GET,PUT,DELETE,POST,HEAD）	允许的跨域请求方法。 父节点：CORSRule
AllowedHeader	字符串	控制在OPTIONS预取指令中Access-Control-Request-Headers指定的header是否允许。在Access-Control-Request-Headers中指定的每个header都在AllowedHeader中有一条对应的项。 父节点：CORSRule

名称	类型	描述
ExposeHeader	字符串	允许用户从应用程序中访问的响应头（例如一个Javascript的XMLHttpRequest对象）。 父节点：CORSRule
MaxAgeSeconds	整型	浏览器对特定资源的预取（OPTIONS）请求返回结果的缓存时间。 一个CORS规则里面最多允许出现一个MaxAgeSeconds。 单位：秒 父节点：CORSRule
CORSConfiguration	容器	Bucket的CORS规则容器。 父节点：无

示例

请求示例

```
Get /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39
****
```

返回示例

```
HTTP/1.1 200
x-oss-request-id: 50519080C4689A033D00****
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
Content-Length: 218
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
    <CORSRule>
      <AllowedOrigin>*</AllowedOrigin>
      <AllowedMethod>GET</AllowedMethod>
      <AllowedHeader>*</AllowedHeader>
      <ExposeHeader>x-oss-test</ExposeHeader>
      <MaxAgeSeconds>100</MaxAgeSeconds>
    </CORSRule>
</CORSConfiguration>
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C++
- C
- .NET
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
NoSuchCORS Configuration	404	目标CORS规则不存在。
AccessDenied	403	只有Bucket的拥有者才能获取CORS规则。

9.4 DeleteBucketCORS

DeleteBucketcors用于关闭指定存储空间（Bucket）对应的跨域资源共享（CORS）功能并清空所有规则。

请求语法

```
DELETE /?cors HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

示例

请求示例

```
DELETE /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWicOME
****
```

返回示例

```
HTTP/1.1 204 No Content
x-oss-request-id: 5051845BC4689A033D00****
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
```

```
Server: AliyunOSS
```

SDK

此接口所对应的各语言SDK如下：

- Java
- Python
- PHP
- Go
- C++
- C
- .NET
- Ruby

错误码

错误码	HTTP 状态码	描述
NoSuchBucket	404	目标Bucket不存在。
AccessDenied	403	没有删除权限。只有Bucket的拥有者才有权限删除Bucket对应的CORS规则。

9.5 OptionObject

浏览器在发送跨域请求之前会发送一个preflight请求（OPTIONS）给OSS，并带上特定的来源域、HTTP方法和header等信息，以决定是否发送真正的请求。

请求语法

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Origin:Origin
Access-Control-Request-Method:HTTP method
Access-Control-Request-Headers:Request Headers
```

请求头

名称	类型	描述
Origin	字符串	请求来源域，用来标示跨域请求。 默认值：无

名称	类型	描述
Access-Control-Request-Method	字符串	表示在实际请求中将会用到的方法。 默认值：无
Access-Control-Request-Headers	字符串	表示在实际请求中会用到的除了简单头部之外的头。 默认值：无

响应头

名称	类型	描述
Access-Control-Allow-Origin	字符串	请求中包含的Origin。如不允许该请求，则响应头不包含该头部。
Access-Control-Allow-Methods	字符串	允许请求的HTTP方法。如不允许该请求，则响应头不包含该头部。
Access-Control-Allow-Headers	字符串	允许请求携带的header的列表。如果请求中有不被允许的header，则不包含该头部，请求也将被拒绝。
Access-Control-Expose-Headers	字符串	允许在客户端JavaScript程序中访问的headers的列表。
Access-Control-Max-Age	整型	允许浏览器缓存preflight结果的时间，单位为秒。

示例

请求示例

```
OPTIONS /testobject HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Origin:http://www.example.com
Access-Control-Request-Method:PUT
```

```
Access-Control-Request-Headers:x-oss-test
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 5051845BC4689A033D00****
Date: Fri, 24 Feb 2012 05:45:34 GMT
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-oss-test
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

错误码

错误码	HTTP 状态码	描述
Forbidden	403	OSS可以通过PutBucketCORS接口来开启Bucket的CORS功能。开启CORS功能后，OSS在收到浏览器preflight请求时会根据设定的规则评估是否允许本次请求，如果不允许或者CORS功能没有开启，则返回此错误码。

10 关于LiveChannel的操作

10.1 LiveChannel简介

您可以使用RTMP协议将音视频数据上传到OSS，转储为指定格式的音视频文件。上传前需要先创建一个LiveChannel，以获取对应的推流地址。

通过RTMP协议上传音视频数据目前有以下限制：

- 只能使用RTMP推流的方式，不支持拉流。
- 必须包含视频流，且视频流格式为H264。
- 音频流是可选的，并且只支持AAC格式，其他格式的音频流会被丢弃。
- 转储只支持HLS协议。
- 一个LiveChannel同时只能有一个客户端向其推流。

10.2 RTMP推流地址及签名

本文介绍RTMP推流地址及其签名规则。

RTMP推流地址形如：rtmp://your-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-channel

其组成规则为: `rtmp://${bucket}.${host}/live/${channel}?${params}`

- live为RTMP协议的app名称，OSS固定使用live。
- params为推流的参数，格式与HTTP请求的query string相同，即形如”varA=valueA&varB=valueB“。
- BucketAcl非public-read-write时，推流地址需要签名才可以使用；签名方法类似OSS的Url签名，但有一些细节上的不同，后文会描述具体的规则。

RTMP推流支持的url参数

名称	描述
playlistName	用来指定生成的m3u8文件名称，其值覆盖LiveChannel中的配置。注意：生成的m3u8名称仍然会被添加”${channel_name}/“前缀。

推流地址的签名规则

一个带签名的推流地址形如：`rtmp://${bucket}.${host}/live/${channel}?OSSAccessKeyId=xxx&Expires=yyy&Signature=zzz&${params}`

参数名称	描述
OSSAccessKeyId	意义同OSS的HTTP签名的AccessKeyId
Expires	过期时间戳，格式采用Unit timestamp
Signature	签名字符串，后文会描述其计算方法
params	其他参数，所有的参数都需要参与签名

Signature的计算规则如下：

```
base64(hmac-sha1(AccessKeySecret,
    + Expires + "\n"
    + CanonicalizedParams
    + CanonicalizedResource))
```

名称	描述
CanonicalizedResource	格式为 “/BucketName/ChannelName”
CanonicalizedParams	按照param key字典序拼接”key:value\n”,将所有的参数拼起来，如果参数个数为0，那么这一项为空。参数中不包含SecurityToken、OSSAccessKeyId和Expire以及Signature。每一个param key只能出现一次。

10.3 PutLiveChannel

通过RTMP协议上传音视频数据前，必须先调用该接口创建一个LiveChannel。调用PutLiveChannel接口会返回RTMP推流地址，以及对应的播放地址。

说明:

您可以使用返回的地址进行推流、播放，您还可以根据该LiveChannel的名称来发起相关的操作，如查询推流状态、查询推流记录、禁止推流等。

请求语法

```
PUT /ChannelName?live HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT date
Content-Length: Size
Authorization: SignatureValue
<?xml version="1.0" encoding="UTF-8"?>
```

```
<LiveChannelConfiguration>
  <Description>ChannelDescription</Description>
  <Status>ChannelStatus</Status>
  <Target>
     <Type>HLS</Type>
     <FragDuration>FragDuration</FragDuration>
     <FragCount>FragCount</FragCount>
     <PlaylistName>PlaylistName</PlaylistName>
  </Target>
  <Snapshot>
    <RoleName>Snapshot ram role</RoleName>
    <DestBucket>Snapshot dest bucket</DestBucket>
    <NotifyTopic>Notify topic of MNS</NotifyTopic>
    <Interval>Snapshot interval in second</Interval>
  </Snapshot>
</LiveChannelConfiguration>
```

请求头

名称	类型	是否必选	描述
ChannelName	字符串	是	必须符合ObjectName的命名规范，另外，ChannelName不能包含斜杠（/）。

请求元素

名称	类型	是否必选	描述
LiveChannelConfiguration	容器	是	保存LiveChannel配置的容器。 子节点：Description、Status、Target 父节点：无

名称	类型	是否必选	描述
Description	字符串	否	LiveChannel的描述信息，最长128字节。 子节点：无 父节点：LiveChannelConfiguration
Status	枚举字符串	否	指定LiveChannel的状态。 子节点：无 父节点：LiveChannelConfiguration 有效值：enabled、disabled 默认值：enabled

名称	类型	是否必选	描述
Target	容器	是	保存转储配置的容器。 子节点：Type、FragDuration、FragCount、PlaylistName 父节点：LiveChannelConfiguration

名称	类型	是否必选	描述
Type	枚举字符串	是	指定转储的类型。 子节点：无 父节点：Target 有效值：HLS  说明： · 转储类型为HLS时，OSS会在生成每个ts文件后更新m3u8文件。m3u8文件中最多包含最近的FragCount个ts文件。 · 转储类型为HLS时，写入当前ts文件的音视频数据时长达到FragDuration指定的时长后，OSS会在收到下一个关键帧的时切换到下一个ts文件；如果max(2*FragDuration, 60s)后仍未收

名称	类型	是否必选	描述
FragDuration	字符串	否	当Type为HLS时，指定每个ts文件的时长。 单位：秒 子节点：无 父节点: Target 取值范围：[1, 100] 默认值：5  说明: FragDuration和FragCount的默认值只有在两者都未指定时才会生效；指定了其中一个，则另一个的值也必须指定。

名称	类型	是否必选	描述
FragCount	字符串	否	当Type为HLS时，指定m3u8文件中包含ts文件的个数。 子节点：无 父节点：Target 取值范围：[1, 100] 默认值：3 说明：FragDuration和FragCount的默认值只有在两者都未指定时才会生效；指定了其中一个，则另一个的值也必须指定。
PlaylistName	字符串	否	当Type为HLS时，指定生成的m3u8文件的名称。必须以”.m3u8”结尾，长度范围为[6, 128]。 子节点：无 父节点：Target 默认值：playlist.m3u8 取值范围：[6, 128]

名称	类型	是否必选	描述
Snapshot	容器	否	保存高频截图操作Snapshot选项的容器。 子节点： RoleName，DestBucket，NotifyTopic，Interval，PornRec 父节点： Snapshot
RoleName	字符串	否	用于高频截图操作的角色名称，要求有DestBucket的写权限和向NotifyTopic发消息的权限。 子节点：无 父节点： Snapshot
DestBucket	字符串	否	保存高频截图目标Bucket，要求与当前Bucket是同一个Owner。 子节点：无 父节点： Snapshot

名称	类型	是否必选	描述
NotifyTopic	字符串	否	用于通知用户高频截图操作结果的MNS的Topic。 子节点：无 父节点：Snapshot
Interval	数字	否	高频截图的间隔长度。如果该段间隔时间内没有关键帧（I 帧），那么该间隔时间不截图。 单位：秒 子节点：无 父节点：Snapshot 取值范围：[1, 100]

响应元素

名称	类型	描述
CreateLiveChannelResult	容器	保存CreateLiveChannel请求结果的容器。 子节点：PublishUrls,PlayUrls 父节点：无
PublishUrls	容器	保存推流地址的容器。 子节点：Url 父节点：CreateLiveChannelResult

名称	类型	描述
Url	字符串	推流地址。 子节点：无 父节点：PublishUrls 说明： · 推流地址是未加签名的URL，如Bucket ACL非public-read-write，则需先进行签名才可访问。 · 播放地址是未加签名的URL，如Bucket ACL为private，则需先进行签名才可访问。
PlayUrls	容器	保存播放地址的容器。 子节点：Url 父节点：CreateLiveChannelResult
Url	字符串	播放地址。 子节点：无 父节点：PlayUrls

示例

请求示例

```
PUT /test-channel?live HTTP/1.1
Date: Wed, 24 Aug 2016 11:11:28 GMT
Content-Length: 333
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:hvwOZJRh8toAj3DZvtsuPgf+a****
<?xml version="1.0" encoding="utf-8"?>
<LiveChannelConfiguration>
    <Description/>
    <Status>enabled</Status>
    <Target>
        <Type>HLS</Type>
        <FragDuration>2</FragDuration>
        <FragCount>3</FragCount>
    </Target>
    <Snapshot>
        <RoleName>role_for_snapshot</RoleName>
        <DestBucket>snapshotdest</DestBucket>
        <NotifyTopic>snapshotnotify</NotifyTopic>
        <Interval>1</Interval>
     </Snapshot>
```

```
</LiveChannelConfiguration>
```

返回示例

```
HTTP/1.1 200
content-length: 259
server: AliyunOSS
x-oss-server-time: 4
connection: close
x-oss-request-id: 57BD8419B92475920B0002F1
date: Wed, 24 Aug 2016 11:11:28 GMT
x-oss-bucket-storage-type: standard
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<CreateLiveChannelResult>
  <PublishUrls>
    <Url>rtmp://test-bucket.oss-cn-hangzhou.aliyuncs.com/live/test-
channel</Url>
  </PublishUrls>
  <PlayUrls>
    <Url>http://test-bucket.oss-cn-hangzhou.aliyuncs.com/test-channel/
playlist.m3u8</Url>
  </PlayUrls>
</CreateLiveChannelResult>
```

10.4 ListLiveChannel

ListLiveChannel接口用于罗列指定的LiveChannel。

请求语法

```
GET /?live HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
```

请求元素

名称	类型	是否必选	描述
Marker	字符串	否	设定结果从marker之后按字母排序的第一个开始返回。
Max-keys	字符串	否	限定此次返回LiveChannel的最大数。max-keys取值不能大于1000。 默认值：100

名称	类型	是否必选	描述	
Prefix	字符串	否	限定返回的LiveChannel必须以prefix作为前缀。注意使用prefix查询时，返回的key中仍会包含prefix。	

响应元素

名称	类型	描述
ListLiveChannelResult	容器	保存ListLiveChannel请求结果的容器。 子节点：Prefix、Marker、MaxKeys、IsTruncated、NextMarker、LiveChannel 父节点：无
Prefix	字符串	本次查询结果的开始前缀。 子节点：无 父节点：ListLiveChannelResult
Marker	字符串	本次ListLiveChannel的起点。 子节点：无 父节点：ListLiveChannelResult
MaxKeys	字符串	响应请求内返回结果的最大数目。 子节点：无 父节点：ListLiveChannelResult

名称	类型	描述
IsTruncated	字符串	指明是否所有的结果都已经返回，其中： · ture果。 · false表示本次已经返回了全部结果。 子节点：无 父节点：ListLiveChannelResult
NextMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextMarker元素，用于标明接下来请求的Marker值。 子节点：无 父节点：ListLiveChannelResult
LiveChannel	容器	保存返回每个LiveChannel信息的容器。 子节点：Name、Description、Status、LastModified、PublishUrls、PlayUrls 父节点：ListLiveChannelResult
Name	字符串	LiveChannel的名称。 子节点：无 父节点：LiveChannel
Description	字符串	LiveChannel的描述。 子节点：无 父节点：LiveChannel

名称	类型	描述
Status	枚举字符串	LiveChannel的状态。 子节点：无 父节点：LiveChannel 有效值：disabledenabled
LastModified	字符串	LiveChannel配置的最后修改时间。 格式：ISO8601 子节点：无 父节点：LiveChannel
PublishUrls	容器	保存LiveChannel对应的推流地址的容器。 子节点：Url 父节点：LiveChannel
Url	字符串	LiveChannel对应的推流地址。 子节点：无 父节点：PublishUrls
PlayUrls	容器	保存LiveChannel对应的播放地址的容器。 子节点：Url 父节点：LiveChannel
Url	字符串	LiveChannel对应的播放地址。 子节点：无 父节点：PlayUrls

示例

请求示例

```
GET /?live&max-keys=1 HTTP/1.1
```

```
Date: Thu, 25 Aug 2016 07:50:09 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:TaX+tlc/Xsgpz6uRuqcbmUJs****
```

返回示例

```
HTTP/1.1 200
content-length: 656
server: AliyunOSS
connection: close
x-oss-request-id: 57BEA331B92475920B00245E
date: Thu, 25 Aug 2016 07:50:09 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<ListLiveChannelResult>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>true</IsTruncated>
  <NextMarker>channel-0</NextMarker>
  <LiveChannel>
    <Name>channel-0</Name>
    <Description></Description>
    <Status>disabled</Status>
    <LastModified>2016-07-30T01:54:21.000Z</LastModified>
    <PublishUrls>
      <Url>rtmp://test-bucket.oss-cn-hangzhou.aliyuncs.com/live/
channel-0</Url>
    </PublishUrls>
    <PlayUrls>
      <Url>http://test-bucket.oss-cn-hangzhou.aliyuncs.com/channel-0/
playlist.m3u8</Url>
    </PlayUrls>
  </LiveChannel>
```

10.5 DeleteLiveChannel

DeleteLiveChannel接口用于删除指定的LiveChannel。

说明:

- 当有客户端正在向LiveChannel推流时，删除请求会失败。
- 本接口只会删除LiveChannel本身，不会删除推流生成的文件。

请求语法

```
DELETE /ChannelName?live HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
```

```
Authorization: SignatureValue
```

示例

请求示例

```
DELETE /test-channel?live HTTP/1.1
Date: Thu, 25 Aug 2016 07:32:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:ZbfvQ3XwmYEE8O9CX8kwVQY****
```

返回示例

```
HTTP/1.1 204
content-length: 0
server: AliyunOSS
connection: close
x-oss-request-id: 57BE9F0AB92475920B0*****
date: Thu, 25 Aug 2016 07:32:26 GMT
```

10.6 PutLiveChannelStatus

LiveChannel有两种状态：enabled和disabled，您可以使用本接口在两种状态之间进行切换。

说明:

- **LiveChannel处于disabled状态时，OSS会禁止您向该LiveChannel进行推流操作。如果您正在向该LiveChannel推流，那么推流的客户端会被强制断开（会有10s左右的延迟）。**
- **当没有客户端向该LiveChannel推流时，调用PutLiveChannel重新创建LiveChannel也可以达到修改Status的目的。**
- **当有客户端向该LiveChannel推流时，无法调用PutLiveChannel重新创建LiveChannel，只能通过本接口修改LiveChannel的状态为disabled。**

请求语法

```
PUT /ChannelName?live&status=NewStatus HTTP/1.1Date: GMT dateHost:
BucketName.oss-cn-hangzhou.aliyuncs.comAuthorization: SignatureValue
```

请求头

名称	类型	是否必选	描述
NewStatus	字符串	是	指定LiveChannel的目标Status。 有效值： enabled、 disabled

示例

请求示例

```
PUT /test-channel?live&status=disabled HTTP/1.1
Date: Thu, 25 Aug 2016 05:37:38 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:X/mBrSbkNoqM/JoAfRC0ytyQ****
```

返回示例

```
HTTP/1.1 200
content-length: 0
server: AliyunOSS
connection: close
x-oss-request-id: 57BE8422B92475920B002030
date: Thu, 25 Aug 2016 05:37:39 GMT
```

10.7 GetLiveChannelInfo

GetLiveChannelInfo接口用于获取指定LiveChannel的配置信息。

请求语法

```
GET /ChannelName?live HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
```

```
Authorization: SignatureValue
```

响应元素

名称	类型	描述
LiveChannelConfiguration	容器	保存GetLiveChannelInfo返回结果的容器。 子节点：Description、Status、Target 父节点：无
Description	字符串	LiveChannel的描述信息。 子节点：无 父节点：LiveChannelConfiguration
Status	枚举字符串	LiveChannel的状态信息。 子节点：无 父节点：LiveChannelConfiguration 有效值：enabled、disabled
Target	容器	保存LiveChannel转储配置的容器。 子节点： Type、FragDuration、FragCount、Play 说明： FragDuration、FragCount、PlaylistNa有当Type取值为HLS时才会返回。 父节点：LiveChannelConfiguration

名称	类型	描述
Type	枚举字符串	当Type为HLS时，指定推流时转储文件类型。 子节点：无 父节点：Target 有效值：HLS
FragDuration	字符串	当Type为HLS时，指定每个ts文件的时长。 单位：秒 子节点：无 父节点: Target
FragCount	字符串	当Type为HLS时，指定m3u8文件中包含ts文件的个数。 子节点：无 父节点：Target
PlaylistName	字符串	当Type为HLS时，指定生成的m3u8文件的名称。 子节点：无 父节点：Target

示例

请求示例

```
GET /test-channel?live HTTP/1.1
Date: Thu, 25 Aug 2016 05:52:40 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:D6bDCRXKht58hin1BL83wxyG****
```

返回示例

```
HTTP/1.1 200
content-length: 475
server: AliyunOSS
connection: close
x-oss-request-id: 57BE87A8B92475920B00****
date: Thu, 25 Aug 2016 05:52:40 GMT
content-type: application/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChannelConfiguration>
  <Description></Description>
  <Status>enabled</Status>
  <Target>
    <Type>HLS</Type>
    <FragDuration>2</FragDuration>
    <FragCount>3</FragCount>
    <PlaylistName>playlist.m3u8</PlaylistName>
  </Target>
</LiveChannelConfiguration>
```

10.8 GetLiveChannelStat

GetLiveChannelStat接口用于获取指定LiveChannel的推流状态信息。

请求语法

```
GET /ChannelName?live&comp=stat HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
```

响应元素

名称	类型	描述
LiveChannelStat	容器	保存GetLiveChannelStat返回结果的容器。 子节点：Status、ConnectedTime、Video、Audio 父节点：无
Status	枚举字符串	LiveChannel当前的推流状态描述。 子节点：无 父节点：LiveChannelStat 有效值：DisabledLiveIdle
ConnnectedTime	字符串	当Status为Live时，表示当前客户端开始推流的时间。此元素使用ISO8601格式表示。 子节点：无 父节点：LiveChannelStat

名称	类型	描述
RemoteAddr	字符串	当Status为Live时，表示当前推流客户端的IP地址。 子节点：无 父节点：LiveChannelStat
Video	容器	当Status为Live时，保存视频流信息的容器。 说明: Video、Audio容器只有在Status为Live时才会返回，但Status为Live时不一定返回这两个容器。例如，客户端已经连接到LiveChannel，但尚未发送音视频数据，这种情况不会返回这两个容器。 子节点：Width、Heigth、FrameRate、Bandwidth、Codec 父节点：LiveChannelStat
Width	字符串	当前视频流的画面宽度。 单位：像素 子节点：无 父节点：Video
Heigth	字符串	当前视频流的画面高度。 单位：像素 子节点：无 父节点：Video
FrameRate	字符串	当前视频流的帧率。 子节点：无 父节点：Video

名称	类型	描述
Bandwidth	字符串	当前视频流的码率。 单位：B/s 子节点：无 父节点：Video
Codec	枚举字符串	当前视频流的编码格式。 子节点：无 父节点：Video
Audio	容器	当Status为Live时，保存音频流信息的容器。 说明: Video、Audio容器只有在Status为Live时才会返回，但Status为Live时不一定返回这两个容器。例如，客户端已经连接到LiveChannel，但尚未发送音视频数据，这种情况不会返回这两个容器。 子节点：SampleRate、Bandwidth、Codec 父节点：LiveChannelStat
SampleRate	字符串	当前音频流的采样率。 子节点：无 父节点：Audio

名称	类型	描述
Bandwidth	字符串	当前音频流的码率。 说明: Bandwidth为音频流/视频流最近一段时间内的平均码率。LiveChannel刚切换到Live状态时，返回的Bandwidth值可能为0。 单位：B/s 子节点：无 父节点：Audio
Codec	枚举字符串	当前音频流的编码格式。 子节点：无 父节点：Audio

示例

- **请求示例 1**

```
GET /test-channel?live&comp=stat HTTP/1.1
Date: Thu, 25 Aug 2016 06:22:01 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:fOzwkAgVTVSO1VKLPIInQ0JY****
```

返回示例 1

```
HTTP/1.1 200
content-length: 100
server: AliyunOSS
connection: close
x-oss-request-id: 57BE8E89B92475920B002164
date: Thu, 25 Aug 2016 06:22:01 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<LiveChannelStat>
  <Status>Idle</Status>
</LiveChannelStat>
```

- **请求示例 2**

```
GET /test-channel?live&comp=stat HTTP/1.1
Date: Thu, 25 Aug 2016 06:25:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
```

```
Authorization: OSS YJjHKOKWDWINLKXv:WeC5joEaRzfSSS8xK0tlo7WT****
```

返回示例 2

```
HTTP/1.1 200
content-length: 469
server: AliyunOSS
connection: close
x-oss-request-id: 57BE8F56B92475920B002187
date: Thu, 25 Aug 2016 06:25:26 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<LiveChannelStat>
  <Status>Live</Status>
  <ConnectedTime>2016-08-25T06:25:15.000Z</ConnectedTime>
  <RemoteAddr>10.1.2.3:47745</RemoteAddr>
  <Video>
    <Width>1280</Width>
    <Height>536</Height>
    <FrameRate>24</FrameRate>
    <Bandwidth>0</Bandwidth>
    <Codec>H264</Codec>
  </Video>
  <Audio>
    <Bandwidth>0</Bandwidth>
    <SampleRate>44100</SampleRate>
    <Codec>ADPCM</Codec>
  </Audio>
</LiveChannelStat>
```

10.9 GetLiveChannelHistory

GetLiveChannelHistory接口用于获取指定LiveChannel的推流记录。使用GetLiveChannelHistory接口最多会返回指定LiveChannel最近的10次推流记录。

请求语法

```
GET /ChannelName?live&comp=history HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
```

响应元素

名称	类型	描述
LiveChannelHistory	容器	保存GetLiveChannelHistory返回结果的容器。 子节点：LiveRecord 父节点：无

名称	类型	描述
LiveRecord	容器	保存一次推流记录信息的容器。 子节点：StartTime、EndTime、RemoteAddr 父节点：LiveChannelHistory
StartTime	字符串	推流开始时间，使用ISO8601格式表示。 子节点：无 父节点：LiveRecord
EndTime	字符串	推流结束时间，使用ISO8601格式表示。 子节点：无 父节点：LiveRecord
RemoteAddr	字符串	推流客户端的ip地址。 子节点：无 父节点：LiveRecord

示例

请求示例

```
GET /test-channel?live&comp=history HTTP/1.1
Date: Thu, 25 Aug 2016 07:00:12 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:pqgDBP8JXTXAytBoXpvNoZfo****
```

返回示例

```
HTTP/1.1 200
content-length: 1892
server: AliyunOSS
connection: close
x-oss-request-id: 57BE977CB92475920B0022FB
date: Thu, 25 Aug 2016 07:00:12 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<LiveChannelHistory>
  <LiveRecord>
    <StartTime>2016-07-30T01:53:21.000Z</StartTime>
```

```
    <EndTime>2016-07-30T01:53:31.000Z</EndTime>
    <RemoteAddr>10.101.194.148:56861</RemoteAddr>
  </LiveRecord>
  <LiveRecord>
    <StartTime>2016-07-30T01:53:35.000Z</StartTime>
    <EndTime>2016-07-30T01:53:45.000Z</EndTime>
    <RemoteAddr>10.101.194.148:57126</RemoteAddr>
  </LiveRecord>
  <LiveRecord>
    <StartTime>2016-07-30T01:53:49.000Z</StartTime>
    <EndTime>2016-07-30T01:53:59.000Z</EndTime>
    <RemoteAddr>10.101.194.148:57577</RemoteAddr>
  </LiveRecord>
  <LiveRecord>
    <StartTime>2016-07-30T01:54:04.000Z</StartTime>
    <EndTime>2016-07-30T01:54:14.000Z</EndTime>
    <RemoteAddr>10.101.194.148:57632</RemoteAddr>
  </LiveRecord>
</LiveChannelHistory>
```

10.10 PostVodPlaylist

PostVodPlaylist接口用于为指定的LiveChannel生成一个点播用的播放列表。OSS会查询指定时间范围内由该LiveChannel推流生成的ts文件，并将其拼装为一个m3u8播放列表。

请求语法

```
POST /ChannelName/PlaylistName?vod&endTime=EndTime&startTime=StartTime
 HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
```

请求参数

名称	类型	是否必选	描述
ChannelName	字符串	是	指定LiveChannel名称，该LiveChannel必须存在。
PlaylistName	字符串	是	指定生成的点播播放列表的名称，必须以“.m3u8”结尾。
StartTime	整数	是	指定查询ts文件的起始时间，格式为Unix timestamp。

名称	类型	是否必选	描述
EndTime	整数	是	指定查询ts文件的终止时间。 格式：Unix timestamp 说明：EndTime必须大于StartTime，且时间跨度不能大于1天。

示例

请求示例

```
POST /test-channel/vod.m3u8?vod&endTime=1472020226&startTime=
1472020031 HTTP/1.1
Date: Thu, 25 Aug 2016 07:13:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:ABIigvnLtCHK+7fMHLeRlOUn****
```

返回示例

```
HTTP/1.1 200
content-length: 0
server: AliyunOSS
connection: close
etag: "9C6104DD9CF1A0C4D0CFD21F4390****"
x-oss-request-id: 57BE9A96B92475920B002359
date: Thu, 25 Aug 2016 07:13:26 GMT
```

10.11 GetVodPlaylist

GetVodPlaylist接口用于查看指定LiveChannel推流生成的、且指定时间段内的播放列表。

请求语法

```
GET /ChannelName?vod&endTime=EndTime&startTime=StartTime HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
```

```
Authorization: SignatureValue
```

请求头

名称	类型	是否必选	描述
ChannelName	字符串	是	指定LiveChannel名称。该LiveChannel必须存在。
StartTime	整数	是	指定查询ts文件的起始时间，格式为Unix timestamp。
EndTime	整数	是	指定查询ts文件的终止时间，格式为Unix timestamp。  说明：EndTime必须大于StartTime，且时间跨度不能大于1天。

示例

请求示例

```
GET /test-channel?vod&endTime=1472020226&startTime=1472020031 HTTP/1.1
Date: Thu, 25 Aug 2016 07:13:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:ABIigvnLtCHK+7fMHLeRlOUn****
```

返回示例

```
HTTP/1.1 200
content-length: 312
server: AliyunOSS
connection: close
etag: "9C6104DD9CF1A0C4D0CFD21F43905D59"
x-oss-request-id: 57BE9A96B92475920B002359
date: Thu, 25 Aug 2016 07:13:26 GMT
Content-Type: application/x-mpegURL
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:13
#EXTINF:7.120,
1543895706266.ts
#EXTINF:5.840,
1543895706323.ts
#EXTINF:6.400,
1543895706356.ts
#EXTINF:5.520,
1543895706389.ts
#EXTINF:5.240,
1543895706428.ts
```

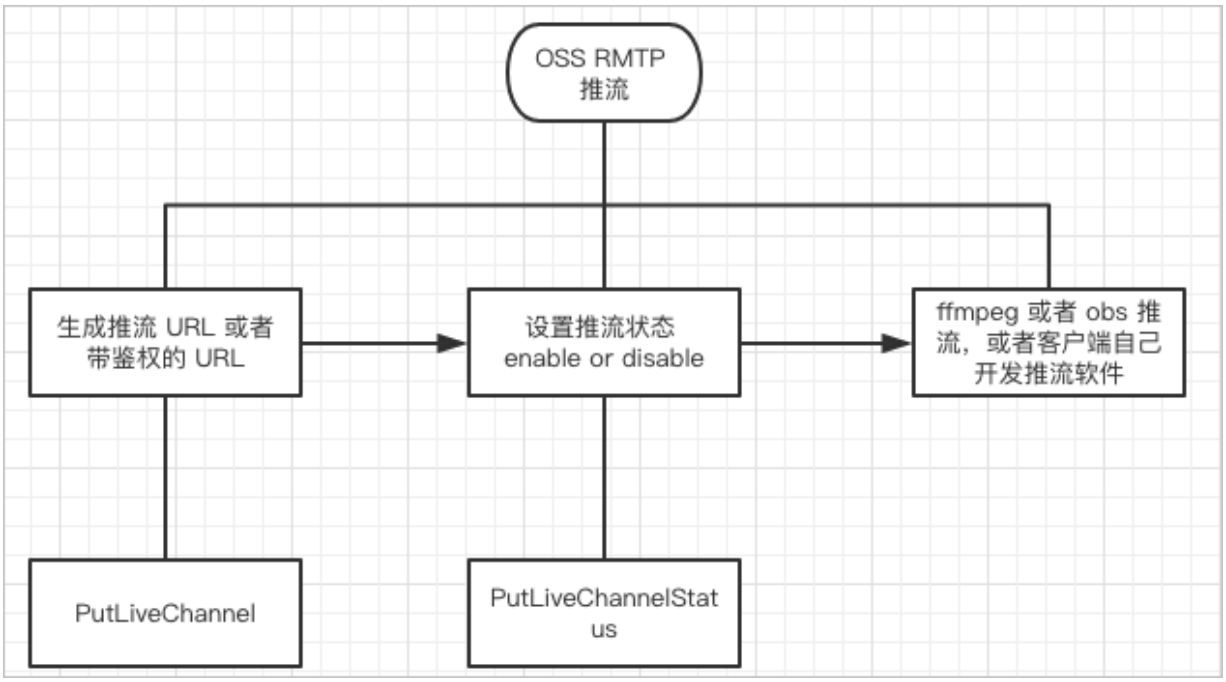
```
#EXTINF:13.320,
1543895706468.ts
#EXTINF:5.960,
1543895706538.ts
#EXTINF:6.520,
1543895706561.ts
#EXT-X-ENDLIST
```

10.12 LiveChannel常见问题

本文主要介绍 LiveChannel 中遇到的常见问题及解决方案。

OSS livechannel 推流过程

下图为 OSS livechannel 推流过程图解，了解这个过程，有助于排查 livechannel 推流过程中遇到的问题。



更多详情请参考：

- 生成推流 URL
- 设置推流状态

案例：录制 M3u8 缺失

问题分析：默认录制成品的 m3u8 只有最后 3 片，遵循的是 HLS 协议的默认规则。

解决方案：用 `PostVodPlaylist` 接口将指定时间范围内的 *ts* 文件汇聚到一个 m3u8 索引内来解决。

说明:

- `EndTime` 必须大于 `StartTime`，且时间跨度不能大于 1 天。
- OSS 会查询指定时间范围内的所有指定 LiveChannel 推流生成的 *ts* 文件，并将其拼装为一个播放列表。

案例：录制 m3u8 文件失败

问题分析：录制的 m3u8 文件成功推流到 OSS 才算录制成功。

解决方案：可以在客户端抓包查看是否有 publish succees 的标志，有这个标志才表示和 OSS 音频包交互成功。所以发现客户端推流有记录，但是就是没有录制视频的情况，可以抓包分析一下。

案例：客户端无法推流到 OSS

使用 ffmpeg 推流不成功：

```
ffmpeg -re -i 0_20180525105430445.aac -acodec aac -strict -2 -f
flv rtmp://xxx.oss-cn-beijing.aliyuncs.com/live/test_1000?Expires
=1540458859&OSSAccessKeyId=LTAlujianb6C9z&Signature=qwh31xQsan
mao6ygCFJgovNIg%3D&playlistName=playlist.m3u8
```

解决方案：

- 使用 ffmpeg 推流不成功的时候建议用最原始的命令推流，不要加一些复杂的参数。
- 推流 URL 在有 & 符号时请将整个 URL 用 双引号（""） 囊括起来。例如，ffmpeg -re -i 0_20180525105430445.aac -acodec aac -strict -2 -f flv "rtmp://xxx.oss-cn-beijing.aliyuncs.com/live/test_1000?Expires=1540458859&OSSAccessKeyId=LTAlujianb6C9z&Signature=qwh31xQsanmao6ygCFJgovNIg%3D&playlistName=playlist.m3u8"。
- 尝试更换成 obs 推流测试，确认是否因 ffmpeg 问题导致的推流失败；

案例：录制 M3u8 文件卡顿

转储类型为 HLS 时，写入当前 *ts* 文件的音视频数据时长达到`FragDuration` 指定的时长后，OSS 会在收到下一个关键帧的时候切换到下一个 *ts* 文件。如果 `max(2*FragDuration, 60s)` 后仍未收到下一个关键帧，OSS 强制切换文件，此时可能引起播放时卡顿。

案例：录制 M3u8 文件没有音频或视频

以下几种情况会出现音频或者视频没有录制的情况：

- `AVC header` 或者 `AAC header` 没有发送，可以通过抓包查看。
- `RTMP message` 长度小于2，或者 `sequence header` 非常小。
- 音频 `Message` 超过缓冲区大小。
- `codec_ ctx` 是解码上下文的关键信息，如果携带的音视频数据异常，也会导致录制失败。

案例：ffmpeg 推流到 OSS 录制没有音频

- 直接查看 ffmpeg 记录的日志，确定客户端是否有发送 `aac_header`。
- 在客户端抓个 RTMP 的包，查看是否发送了 `aac_header`。