

# 对象存储 OSS

API 参考

# API 参考

## OSS API 文档简介

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。用户可以通过本文档提供的简单的RESTful接口，在任何时间、任何地点、任何互联网设备上上传和下载数据。基于OSS，用户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。

本文档详细介绍了各个接口的请求语法、请求示例、返回示例，以及使用接口时的注意事项等，帮助您快速进行二次开发以及与第三方技术集成。

请确保在使用这些接口前，已充分了解OSS产品说明、使用协议明和收费方式。

## API概览

### 基本概念

本文中出现的术语请参考OSS 基本概念

### 关于Service操作

| API        | 描述             |
|------------|----------------|
| GetService | 得到该账户下所有Bucket |

### 关于Bucket的操作

| API            | 描述           |
|----------------|--------------|
| Put Bucket     | 创建Bucket     |
| Put Bucket ACL | 设置Bucket访问权限 |

|                         |                        |
|-------------------------|------------------------|
| Put Bucket Logging      | 开启Bucket日志             |
| Put Bucket Website      | 设置Bucket为静态网站托管模式      |
| Put Bucket Referer      | 设置Bucket的防盗链规则         |
| Put Bucket Lifecycle    | 设置Bucket中Object的生命周期规则 |
| Get Bucket Acl          | 获得Bucket访问权限           |
| Get Bucket Location     | 获得Bucket所属的数据中心位置信息    |
| Get Bucket Logging      | 查看Bucket的访问日志配置情况      |
| Get Bucket Website      | 查看Bucket的静态网站托管状态      |
| Get Bucket Referer      | 查看Bucket的防盗链规则         |
| Get Bucket Lifecycle    | 查看Bucket中Object的生命周期规则 |
| Delete Bucket           | 删除Bucket               |
| Delete Bucket Logging   | 关闭Bucket访问日志记录功能       |
| Delete Bucket Website   | 关闭Bucket的静态网站托管模式      |
| Delete Bucket Lifecycle | 删除Bucket中Object的生命周期规则 |
| Get Bucket(List Object) | 获得Bucket中所有Object的信息   |
| Get Bucket Info         | 获取Bucket信息             |

## 关于Object的操作

| API                     | 描述                    |
|-------------------------|-----------------------|
| Put Object              | 上传object              |
| Copy Object             | 拷贝一个object成另外一个object |
| Get Object              | 获取Object              |
| Delete Object           | 删除Object              |
| Delete Multiple Objects | 删除多个Object            |
| Head Object             | 获得Object的meta信息       |
| Post Object             | 使用Post上传Object        |
| Append Object           | 在Object尾追加上传数据        |
| Put Object ACL          | 设置Object ACL          |
| Get Object ACL          | 获取Object ACL信息        |
| Callback                | 上传回调                  |

## 关于Multipart Upload的操作

| API                       | 描述                            |
|---------------------------|-------------------------------|
| Initiate Multipart Upload | 初始化MultipartUpload事件          |
| Upload Part               | 分块上传文件                        |
| Upload Part Copy          | 分块复制上传文件                      |
| Complete Multipart Upload | 完成整个文件的Multipart Upload上传     |
| Abort Multipart Upload    | 取消Multipart Upload事件          |
| List Multipart Uploads    | 罗列出所有执行中的Multipart Upload事件   |
| List Parts                | 罗列出指定Upload ID所属的所有已经上传成功Part |

## 跨域资源共享(CORS)

| API                | 描述                         |
|--------------------|----------------------------|
| Put Bucket cors    | 在指定Bucket设定一个CORS的规则       |
| Get Bucket cors    | 获取指定的Bucket目前的CORS规则       |
| Delete Bucket cors | 关闭指定Bucket对应的CORS功能并清空所有规则 |
| Option Object      | 跨域访问preflight请求            |

## 相关文档

- API资源导航
- API Explorer
- API错误中心

## 访问控制

## 用户签名验证

OSS通过使用AccessKeyId/ 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，其中AccessKeySecret必须保密，只有用户和OSS知道。AccessKey 根据所属账号的类型有所区分

- 阿里云账户AccessKey：每个阿里云账号提供的AccessKey拥有对拥有的资源有完全的权限
- RAM账户AccessKey：RAM账户由阿里云账号授权生成，所拥的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，所拥的AccessKey在限定时间内拥有对特定资源限定的操作权限。过期权限收回。

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，需要首先将发送的请求按照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。

## 在Header中包含签名

用户可以在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。

## 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。详情参看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。

注意:

- 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 ( **无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
4. 如果用户请求在指定了查询字符串(QueryString，也叫Http Request Parameters)，那么将这些查询字符串及其请求值按照 **字典序**，从小到大排列，以 & 为分隔符，按参数添加到CanonicalizedResource中。此时的CanonicalizedResource如：  
/BucketName/ObjectName?acl&response-content-

type=ContentType&uploadId=UploadId。

提示：

- 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

1. 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。
2. 签名的方法用RFC 2104中定义的HMAC-SHA1方法，其中Key为 AccessKeySecret`。
3. 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。
4. 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（如上例中的x-oss-magic是需要加入签名字符串的）。
5. 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按字典序从小到大排序。
  - 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

假如AccessKeyId是“44CF9590006BF252F707”，AccessKeySecret是“OtxrxzIsfpFjA7SwPzILWY8Bw21TLhquhboDYROV”

| 请求                                                                   | 签名字符串计算公式                                                                      | 签名字符串                                                   |
|----------------------------------------------------------------------|--------------------------------------------------------------------------------|---------------------------------------------------------|
| PUT /nelson HTTP/1.0<br>Content-MD5:<br>eB5eJF1ptWaXm4bijSPyxw=<br>= | Signature = base64(hmac-sha1(AccessKeySecret, VERB + “\n” + Content-MD5 + “\n” | “PUT\n<br>eB5eJF1ptWaXm4bijSPyxw=<br>=\n<br>text/html\n |

|                                                                                                                                                                                      |                                                                                                  |                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Content-Type: text/html<br>Date: Thu, 17 Nov 2005<br>18:49:58 GMT<br>Host: oss-example.oss-cn-hangzhou.aliyuncs.com<br>X-OSS-Meta-Author:<br>foo@bar.com<br>X-OSS-Magic: abracadabra | + Content-Type + "\n"<br>+ Date + "\n"<br>+ CanonicalizedOSSHeaders<br>+ CanonicalizedResource)) | Thu, 17 Nov 2005 18:49:58<br>GMT\n<br>x-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nelson" |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|

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("OtxrxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
"PUT\nODBGOERFMDMzQTczRUY3NUE3NzA5QzdFNUIYzMDQxNEM=\ntext/html\nThu, 17 Nov 2005 18:49:58\nGMT\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 26NBxoKdsyly4EDv6inkoDft/yA=，因为Authorization = "OSS " + AccessKeyId + ":" + Signature所以最后Authorization为 "OSS 44CF9590006BF252F707:26NBxoKdsyly4EDv6inkoDft/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 44CF9590006BF252F707:26NBxoKdsyly4EDv6inkoDft/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

1. 如果传入的AccessKeyId不存在或inactive，返回403 Forbidden。错误码：InvalidAccessKeyId。
2. 若用户请求头中Authorization值的格式不对，返回400 Bad Request。错误码：InvalidArgument。
3. 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。
4. 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。
5. 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。

6. 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>
1E446260FF9B10C2
</RequestId>
<HostId>
oss-cn-hangzhou.aliyuncs.com
</HostId>
<SignatureProvided>
y5H7yzPsA/tP4+0tH1HHvPEwUv8=
</SignatureProvided>
<StringToSign>
GET
Wed, 11 May 2011 07:59:25 GMT
/oss-example?acl
</StringToSign>
<OSSAccessKeyId>
AKIAIVAKMSMOY7VOMRWQ
</OSSAccessKeyId>
</Error>
```

提示：

- 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，请使用 [可视化签名工具](#) 确认签名并排除错误。

## 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^$j\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())
'NzgxZTVIMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

# 在URL中包含签名

除了使用Authorization Head，用户还可以在URL中加入签名信息，这样用户就可以把该URL转给第三方实现授权访问。

## 实现方式

URL签名示例:

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

**URL签名，必须至少包含Signature，Expires，OSSAccessKeyId三个参数。**

- Expires 这个参数的值是一个UNIX时间（自UTC时间1970年1月1号开始的秒数，详见wiki），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。
- 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)))
```

其中，与header中包含签名相比主要区别如下：

1. 通过URL包含签名时，之前的Date参数换成Expires参数。
  2. 不支持同时在URL和Head中包含签名。
  3. 如果传入的Signature，Expires，OSSAccessKeyId出现不止一次，以第一次为准。
  4. 请求先验证请求时间是否晚于Expires时间，然后再验证签名。
  5. 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
```

## 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("OtxrxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
"GET\n\n1141889120\n/oss-example/oss-api.pdf",
sha)
urllib.quote (base64.encodestring(h.digest()).strip())
```

提示：

- 上面为python的示例代码
- OSS SDK中提供了提供URL签名的方法，使用方法请参看SDK文件中的 授权访问章节 章节
- 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   |

## 细节分析

1. 使用在URL中签名的方式，会将你授权的数据在过期时间以内暴露在互联网上，请预先评估使用风险。
2. PUT和GET请求都支持在URL中签名。
3. 在URL中添加签名时，Signature，Expires，OSSAccessKeyId顺序可以交换，但是如果Signature，Expires，OSSAccessKeyId缺少其中的一个或者多个，返回403 Forbidden。错误码：AccessDenied。

4. 如果访问的当前时间晚于请求中设定的Expires时间，返回403 Forbidden。错误码：AccessDenied。
5. 如果Expires时间格式错误，返回403 Forbidden。错误码：AccessDenied。
6. 如果URL中包含参数Signature，Expires，OSSAccessKeyId中的一个或者多个，并且Head中也包含签名消息，返回消息400 Bad Request。错误码：InvalidArgument。
7. 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。（请求中虽然仍然有Date这个请求头，但不需要将Date加入签名字符串中）

## 临时授权访问

### STS介绍

OSS可以通过阿里云STS服务，临时进行授权访问。阿里云STS (Security Token Service) 是为云计算用户提供临时访问令牌的Web服务。通过STS，您可以为第三方应用或联邦用户（用户身份由您自己管理）颁发一个自定义时效和权限的访问凭证。第三方应用或联邦用户可以使用该访问凭证直接调用阿里云产品API，或者使用阿里云产品提供的SDK来访问云产品API。

您不需要透露您的长期密钥（AccessKey）给第三方应用，只需要生成一个访问令牌并将令牌交给第三方应用即可。

这个令牌的访问权限及有效期限都可以由您自定义。

您不需要关心权限撤销问题，访问令牌过期后就自动失效。

以app应用为例，交互流程如下图：



方案的详细描述如下：

App用户登录。

客户自己管理App用户，自定义身份管理系统，也可以使用外部Web账号或OpenID。对于每个有效的App用户来说，AppServer是可以确切地定义出每个App用户的最小访问权限。

AppServer请求STS服务获取一个安全令牌（ SecurityToken ）。

在调用STS之前，AppServer需要确定App用户的最小访问权限（用Policy语法描述）以及授权的过期时间。通过调用STS的AssumeRole（扮演角色）接口来获取安全令牌。角色管理与使用相关内容请参考RAM使用指南中的角色管理。

STS返回给AppServer一个有效的访问凭证，包括一个安全令牌(SecurityToken)、临时访问密钥（ AccessKeyId、 AccessKeySecret ）以及过期时间。

AppServer将访问凭证返回给ClientApp。

ClientApp可以缓存这个凭证。当凭证失效时，ClientApp需要向AppServer申请新的有效访问凭证。比如，访问凭证有效期为1小时，那么ClientApp可以每30分钟向AppServer请求更新访问凭证。

ClientApp使用本地缓存的访问凭证去请求Alibaba Cloud Service API。云服务会感知STS访问凭证，并会依赖STS服务来验证访问凭证，并正确响应用户请求。

STS安全令牌详情，请参考RAM使用指南中的角色管理。关键是调用STS服务接口AssumeRole来获取有效访问凭证即可。也可以直接使用STS SDK来调用该方法单击查看。

## 使用STS凭证构造签名请求

用户的客户端拿到STS临时凭证后，通过其中安全令牌（ SecurityToken ）以及临时访问密钥（ AccessKeyId、 AccessKeySecret ）构建签名。授权访问签名的构建方式与直接使用根账号的AccessKey在Header中包含签名基本一致，关键注意两点：

用户使用的签名密钥为STS提供的临时访问密钥（ AccessKeyId、 AccessKeySecret ）。

用户需要将安全令牌（ SecurityToken ）携带在请求header中或者以请求参数的形式放入URI中。这两种形式只能选择其一，如果都选择，OSS会返回InvalidArgument错误。

在header中包含头部x-oss-security-token : SecurityToken。计算签名CanonicalizedOSSHeaders时，将x-oss-security-token计算在内。

在URL中携带参数security-token=SecurityToken。计算签名CanonicalizedResource时，将security-token当做一个sub-resource计算在内。

# Bucket权限控制

OSS提供ACL ( Access Control List ) 权限控制方法，OSS ACL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    | <ul style="list-style-type: none"> <li>- 任何人（包括匿名访问）都可以对该Bucket中的Object进行读/写/删除操作。</li> <li>- 所有这些操作产生的费用由该Bucket的Owner承担，<b>请慎用该权限。</b></li> </ul> |
| public-read       | 公共读，私有写 | <ul style="list-style-type: none"> <li>- 只有该Bucket的Owner或者授权对象可以对存放在其中的Object进行写/删除操作。</li> <li>- 任何人（包括匿名访问）可以对Object进行读操作。</li> </ul>             |
| private           | 私有读写    | <ul style="list-style-type: none"> <li>- 只有该Bucket的Owner或者授权对象可以对存放在其中的Object进行读/写/删除操作。</li> <li>- 其他人在未经授权的情况下无法访问该Bucket内的Object。</li> </ul>     |

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。

## 公共HTTP头定义

### 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 | 描述 |
|----|----|
|----|----|

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| Authorization  | 用于验证请求合法性的认证信息。<br>类型：字符串<br>默认值：无<br>使用场景：非匿名请求                          |
| Content-Length | RFC2616中定义的HTTP请求内容长度。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求           |
| Content-Type   | RFC2616中定义的HTTP请求内容类型。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求           |
| Date           | HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT<br>类型：字符串<br>默认值：无  |
| Host           | 访问Host值，格式为：<bucketname>.oss-cn-hangzhou.aliyuncs.com。<br>类型：字符串<br>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             | 描述                                                                                                                                                                      |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Content-Length | RFC2616中定义的HTTP请求内容长度。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求                                                                                                         |
| Connection     | 标明客户端和OSS服务器之间的链接状态。<br>类型：枚举 有效值：open、close<br>默认值：无                                                                                                                   |
| Date           | HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT<br>类型：字符串<br>默认值：无                                                                                                |
| ETag           | 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。<br>类型：字符串<br>默认值：无 |
| Server         | 生成Response的服务器。<br>类型：字符串<br>默认值：AliyunOSS                                                                                                                              |



x-oss-request-id

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。  
 类型：字符串  
 默认值：无

## 关于Service操作

## GetService (ListBuckets)

## GetService (ListBuckets)

对于服务地址作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.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

### 请求参数

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

| 名称     | 是否必须 | 描述                                                                   |
|--------|------|----------------------------------------------------------------------|
| prefix | 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<br>数据类型：字符串<br>默认值：无 |
| marker | 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<br>数据类型：字符串<br>默认值：无     |

|          |   |                                                                         |
|----------|---|-------------------------------------------------------------------------|
| max-keys | 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<br>数据类型：字符串<br>默认值：100 |
|----------|---|-------------------------------------------------------------------------|

## 响应元素(Response Elements)

| 名称                     | 描述                                                                                                                                     |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| ListAllMyBucketsResult | 保存Get Service请求结果的容器。<br>类型：容器<br>子节点: Owner, Buckets<br>父节点：None                                                                      |
| Prefix                 | 本次查询结果的前缀，当bucket未全部返回时才有此节点<br>类型：字符串<br>父节点：ListAllMyBucketsResult                                                                   |
| Marker                 | 标明这次GetService(ListBuckets)的起点，当bucket未全部返回时才有此节点<br>类型：字符串<br>父节点：ListAllMyBucketsResult                                              |
| MaxKeys                | 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<br>类型：字符串<br>父节点：ListAllMyBucketsResult                                                              |
| IsTruncated            | 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。<br>类型：枚举字符串<br>有效值：true、false<br>父节点：ListAllMyBucketsResult |
| NextMarker             | 表示下一次GetService(ListBuckets)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。<br>类型：字符串<br>父节点：ListAllMyBucketsResult                          |
| Owner                  | 用于存放Bucket拥有者信息的容器。<br>类型：容器<br>父节点：ListAllMyBucketsResult                                                                             |
| ID                     | Bucket拥有者的用户ID。<br>类型：字符串<br>父节点：ListAllMyBucketsResult.Owner                                                                          |
| DisplayName            | Bucket拥有者的名称 (目前和ID一致)。<br>类型：字符串<br>父节点：ListAllMyBucketsResult.Owner                                                                  |
| Buckets                | 保存多个Bucket信息的容器。                                                                                                                       |

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

## 细节分析

1. GetService这个API只对验证通过的用户有效。
2. 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。
3. 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 I

```
GET / HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
Authorization: OSS nxj7dtl1c24jwhcyl5hpnhi: COS3OQkfQPnKmyZTEHYv2qUI5jI=
```

### 返回示例 I

```
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: 5374A2880232A65C23002D74

<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Owner>
    <ID>51264</ID>
    <DisplayName>51264</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <CreationDate>2015-12-17T18:12:43.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-shanghai.aliyuncs.com</ExtranetEndpoint>
      <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</ExtranetEndpoint>
      <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 II

```
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 nxj7dtl1c24jwhcyl5hpnhi: COS3OQkfQPnKmyZTEHYv2qUI5jI=
```

### 返回示例 II

```
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: 5374A2880232A65C23002D75

<?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</ExtranetEndpoint>
<IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</IntranetEndpoint>
<Location>oss-cn-hangzhou</Location>
<Name>xz02tphky6fjfiuc0</Name>
<StorageClass>Standard</StorageClass>
</Bucket>
</Buckets>
</ListAllMyBucketsResult>
```

## 关于Bucket的操作

### PutBucket

PutBucket用于创建Bucket（不支持匿名访问）。创建的Bucket所在的Region和发送请求的Endpoint所对应的Region一致。Bucket所在的数据中心确定后，该Bucket下的所有Object将一直存放在对应的地区。更多内容参见 [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>
```

## 细节分析

1. 可以Put请求中的 x-oss-acl 头来设置Bucket访问权限。目前Bucket有三种访问权限：public-read-write，public-read和private。
2. 如果请求的Bucket已经存在，返回409 Conflict。错误码：BucketAlreadyExists。
3. 如果想创建的Bucket不符合命名规范，返回400 Bad Request消息。错误码：InvalidBucketName。
4. 如果用户发起PUT Bucket请求的时候，没有传入用户验证信息，返回403 Forbidden消息。错误码：AccessDenied。
5. 如果PutBucket的时候发现已经超过bucket最大创建数时，默认30个，返回400 Bad Request消息。错误码：TooManyBuckets。
6. 创建的Bucket，如果没有指定访问权限，则默认使用 Private 权限。
7. 创建的Bucket，可以指定Bucket的存储类型，可选值为Standard和IA、Archive。

## 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/KyRt8dOPfo8=

<?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: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2017 03:15:40 GMT
Location: /oss-example
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

# PutBucketACL

Put Bucket ACL接口用于修改Bucket访问权限。目前Bucket有三种访问权限：public-read-write，public-read和private。Put Bucket ACL操作通过Put请求中的“x-oss-acl”头来设置。这个操作只有该Bucket的创建者有权限执行。如果操作成功，则返回200；否则返回相应的错误码和提示信息。

## 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
```

## 细节分析

1. 如果bucket存在，发送时带的权限和已有权限不一样，并且请求发送者是bucket拥有者时。该请求不会改变bucket内容，但是会更新权限。
2. 如果用户发起Put Bucket请求的时候，没有传入用户验证信息，返回403 Forbidden消息。错误码：AccessDenied。
3. 如果请求中没有，“x-oss-acl”头，并且该bucket已存在，并属于该请求发起者，则维持原bucket权限不变。

## 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:KU5h8YMUC78M30dXqf3JxrTZHiA=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 03:21:12 GMT
Content-Length: 0
```

```
Connection: keep-alive
Server: AliyunOSS
```

如果该设置的权限不存在，示例400 Bad Request消息：

错误返回示例：

```
HTTP/1.1 400 Bad Request
x-oss-request-id: 56594298207FB304438516F9
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>56594298207FB304438516F9</RequestId>
  <HostId>leo.oss-test.aliyun-inc.com</HostId>
  <ArgumentName>x-oss-acl</ArgumentName>
  <ArgumentValue>error-acl</ArgumentValue>
</Error>
```

## PutBucketLogging

Bucket的拥有者可以为bucket开启访问日志记录功能。这个功能开启后，OSS将自动记录访问这个bucket请求的详细信息，并按照用户指定的规则，以小时为单位，将访问日志作为一个Object写入用户指定的bucket。OSS提供Bucket访问日志的目的是为了方便bucket的拥有者理解和分析bucket的访问行为。OSS提供的Bucket访问日志不保证记录下每一条访问记录。

**说明：** OSS提供Bucket访问日志的目的是为了方便bucket的拥有者理解和分析bucket的访问行为。OSS提供的Bucket访问日志不保证记录下每一条访问记录。

## 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(Request Elements)

| 名称                  | 描述                                                                                                    | 是否必需         |
|---------------------|-------------------------------------------------------------------------------------------------------|--------------|
| BucketLoggingStatus | 访问日志状态信息的容器<br>类型: 容器<br>子元素: LoggingEnabled<br>父元素: 无                                                | 是            |
| LoggingEnabled      | 访问日志信息的容器。这个元素在开启时需要，关闭时不需要。<br>类型: 容器<br>子元素: TargetBucket, TargetPrefix<br>父元素: BucketLoggingStatus | 否            |
| TargetBucket        | 指定存放访问日志的Bucket。<br>类型: 字符<br>子元素: 无<br>父元素: BucketLoggingStatus.LoggingEnabled                       | 在开启访问日志的时候必需 |
| TargetPrefix        | 指定最终被保存的访问日志文件前缀。<br>类型: 字符<br>子元素: None<br>父元素: BucketLoggingStatus.LoggingEnabled                   | 否            |

## 存储访问日志记录的object命名规则

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

命名规则中，TargetPrefix由用户指定；YYYY, mm, DD, HH, MM和SS分别是该Object被创建时的阿拉伯数字的年，月，日，小时，分钟和秒（注意位数）；UniqueString为OSS系统生成的字符串。一个实际的用于存储OSS访问日志的Object名称例子如下：

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

上例中，“MyLog-” 是用户指定的Object前缀；“oss-example” 是源bucket的名称；“2012-09-10-04-00-00” 是该Object被创建时的北京时间；“0000” 是OSS系统生成的字符串。

## LOG文件格式

| 名称                    | 例子                                       | 含义                               |
|-----------------------|------------------------------------------|----------------------------------|
| Remote IP             | 119.140.142.11                           | 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-URI           | "GET /aliyun-logo.png HTTP/1.1 "         | 用户请求的URI(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头                 |
| HostName              | oss-example.oss-cn-hangzhou.aliyuncs.com | 请求访问域名                           |
| Request ID            | 505B01695037C2AF032593A4                 | 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   | 1657136103983691                         | 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名字                    |
| 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                | 1657136103983691                         | 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 “-” |
| Reserved     | - | 保留字段                |

## 细节分析

- 源Bucket和目标Bucket必须属于同一个用户。
- 上面所示的请求语法中，“BucketName”表示要开启访问日志记录的bucket；“TargetBucket”表示访问日志记录要存入的bucket；“TargetPrefix”表示存储访问日志记录的object名字前缀，可以为空。
- 源bucket和目标bucket可以是同一个Bucket，也可以是不同的Bucket；用户也可以将多个的源bucket的LOG都保存在同一个目标bucket内（建议指定不同的TargetPrefix）。
- 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，具体方法可以参考下面的请求示例。
- 所有PUT Bucket Logging请求必须带签名，即不支持匿名访问。
- 如果PUT Bucket Logging请求发起者不是源bucket（请求示例中的BucketName）的拥有者，OSS返回403错误码；
- 如果源bucket不存在，OSS返回错误码：NoSuchBucket。
- 如果PUT Bucket Logging请求发起者不是目标bucket（请求示例中的TargetBucket）的拥有者，OSS返回403；如果目标bucket不存在，OSS返回错误码：InvalidTargetBucketForLogging。
- 源Bucket和目标Bucket必须属于同一个数据中心，否则返回400错误，错误码为：InvalidTargetBucketForLogging。
- PUT Bucket Logging请求中的XML不合法，返回错误码：MalformedXML。
- 源bucket和目标bucket可以是同一个Bucket；用户也可以将不同的源bucket的LOG都保存在同一个目标bucket内（注意要指定不同的TargetPrefix）。
- 源Bucket被删除时，对应的Logging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。
- OSS生成的Log文件命名规则中的“UniqueString”仅仅是OSS为其生成的UUID，用于唯一标识该文件。
- 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，用户可以按照普通的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。如：<http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png>，<http://oss-example.oss-cn-hangzhou.aliyuncs.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中，你可以通过搜索“x-user=admin”，很方便地定位出经过标记的这个请求。
- 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，请开发者开发Log处理工具时考虑兼容性的

问题。

- 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

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:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?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: 534B371674E88A4D8906008B
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：

```
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:KU5h8YMUC78M30dXqf3JxrTZHiA=

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

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 04:21:12 GMT
```

```
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

## PutBucketWebsite

Put Bucket Website操作可以将一个bucket设置成静态网站托管模式。

### 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            | 描述                                                                                                         | 是否必须 |
|---------------|------------------------------------------------------------------------------------------------------------|------|
| ErrorDocument | 子元素Key的父元素<br>类型: 容器<br>父元素: WebsiteConfiguration                                                          | 否    |
| IndexDocument | 子元素Suffix的父元素.<br>类型: 容器<br>父元素: WebsiteConfiguration                                                      | 是    |
| Key           | 返回404错误时使用的文件名<br>类型: 字符串 父元素:<br>WebsiteConfiguration.ErrorD<br>ocument<br>有条件: 当ErrorDocument设<br>置时, 必需 | 有条件  |

|                      |                                                                                                                                                                                                                        |   |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|
| Suffix               | 返回目录URL时添加的索引文件名，不要为空，也不要包含"/"。例如索引文件设置为index.html，则访问:oss-cn-hangzhou.aliyuncs.com/mybucket/mydir/这样请求的时候默认都相当于访问oss-cn-hangzhou.aliyuncs.com/mybucket/index.html<br>类型:字符串<br>父元素:WebsiteConfiguration.IndexDocument | 是 |
| WebsiteConfiguration | 请求的容器<br>类型：容器<br>父元素: 无                                                                                                                                                                                               | 是 |

## 细节分析

1. 所谓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。
2. 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。具体配置方法见3.4节：自定义域名绑定。
3. 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。
4. 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。
5. 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。
6. 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 qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?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
```

## PutBucketReferer

Put Bucket Referer操作可以设置一个bucket的referer访问白名单和是否允许referer字段为空的请求访问。Bucket Referer防盗链具体见OSS防盗链。

### 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 ( Request Elements )

| 名称                   | 描述                               | 是否必需 |
|----------------------|----------------------------------|------|
| RefererConfiguration | 保存Referer配置内容的容器<br>类型：容器<br>子节点 | 是    |

|                   |                                                                                            |    |
|-------------------|--------------------------------------------------------------------------------------------|----|
|                   | : AllowEmptyReferer节点、RefererList节点<br>父节点：无                                               |    |
| AllowEmptyReferer | 指定是否允许referer字段为空的请求访问。<br>类型：枚举字符串<br>有效值：true或false 默认值：true<br>父节点：RefererConfiguration | 是  |
| RefererList       | 保存referer访问白名单的容器。<br>类型：容器<br>父节点：RefererConfiguration<br>子节点：Referer                     | 是  |
| Referer           | 指定一条referer访问白名单。<br>类型：字符串<br>父节点：RefererList                                             | 可选 |

## 细节分析

1. 只有Bucket的拥有者才能发起Put Bucket Referer请求，否则返回403 Forbidden消息。错误码：AccessDenied。
2. AllowEmptyReferer中指定的配置将替换之前的AllowEmptyReferer配置，该字段为必填项，系统中默认的AllowEmptyReferer配置为true。
3. 此操作将用RefererList中的白名单列表覆盖之前配置在白名单列表，当用户上传的RefererList为空时（不包含Referer请求元素），此操作会覆盖已配置在白名单列表，即删除之前配置的RefererList。
4. 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：

不包含Referer的请求示例：

```
PUT /?referer HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTzHiA=

<?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: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?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: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

## PutBucketLifecycle

## Put Bucket Lifecycle

Bucket的拥有者可以通过Put Bucket Lifecycle来设置Bucket的Lifecycle配置。Lifecycle开启后，OSS将按照配置，定期自动删除或转储与Lifecycle规则相匹配的Object。

### 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(Request Elements)

| 名称                   | 描述                                                                                                                               | 是否必需                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| CreatedBeforeDate    | 指定规则何时之前生效。日期必需服从ISO8601的格式，并且总是UTC的零点。例如：2002-10-11T00:00:00.000Z<br>类型：字符串<br>父节点：Expiration或者Transition或者AbortMultipartUpload | Days和CreatedBeforeDate二选一 |
| Days                 | 指定规则在对象最后修改时间过后多少天生效。<br>类型：正整数<br>父节点：Expiration或者Transition                                                                    | Days和CreatedBeforeDate二选一 |
| Expiration           | 指定Object规则的过期属性。<br>类型：容器<br>子节点：Days或CreatedBeforeDate<br>父节点：Rule                                                              | 否                         |
| AbortMultipartUpload | 指定未完成的Part规则的过期属性。<br>类型：容器<br>子节点：Days或CreatedBeforeDate<br>父节点：Rule                                                            | 否                         |
| ID                   | 规则唯一的ID。最多由255字节组成。当用户没有指定，或者该值为空时，OSS会为用户生成一                                                                                    | 否                         |

|                        |                                                                                                     |                      |
|------------------------|-----------------------------------------------------------------------------------------------------|----------------------|
|                        | 个唯一值。<br>类型：字符串<br>子节点：无<br>父节点：Rule                                                                |                      |
| LifecycleConfiguration | Lifecycle配置的容器，最多可容纳1000条规则。<br>类型：容器<br>子节点：Rule<br>父节点：无                                          | 是                    |
| Prefix                 | 指定规则所适用的前缀。只有匹配前缀的对象才可能被该规则所影响。不可重叠。<br>类型：字符串<br>子节点：无<br>父节点：Rule                                 | 是                    |
| Rule                   | 表述一条规则<br>类型：容器<br>子节点<br>： ID , Prefix , Status , Expiration<br>父节点<br>： LifecycleConfiguration    | 是                    |
| Status                 | 如果其值为Enabled，那么OSS会定期执行该规则；如果是Disabled，那么OSS会忽略该规则。<br>类型：字符串<br>父节点：Rule<br>有效值：Enabled , Disabled | 是                    |
| StorageClass           | 指定对象转储到OSS的目标存储类型<br>类型：字符串<br>子节点：无<br>父节点：Transition<br>有效值：IA或者Archive                           | 是，如果父节点Transition设置了 |
| Transition             | 指定Object在有效生命周期中，OSS何时将对象转储到IA或者Archive存储类型<br>类型：容器<br>子节点：Days或CreatedBeforeDate<br>父节点：Rule      | 否                    |

## 细节分析

1. 只有Bucket的拥有者才能发起Put Bucket Lifecycle请求，否则返回403 Forbidden消息。错误码：AccessDenied。
2. 如果此前没有设置过Lifecycle，此操作会创建一个新的Lifecycle配置；否则，就覆写先前的配置。
3. 可以对Object设置过期时间，也可以对Part设置过期时间。这里的Part指的是以分片上传方式上传

- ，但最后未提交的分片。
4. 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，同时配置转储IA和Archive类型情况下，转储Archive的时间必须比转储IA的时间长。例如Transition IA设置Days为30，Transition Archive设置Days必须大于IA。否则，报InvalidArgument错。
  5. Object设置过期时间必须大于转为IA或者Archive的时间。否则，报InvalidArgument错。
  6. 支持IA Bucket中Objects转储为Archive存储类型。
  7. 不支持Archvie Bucket创建转储规则。
  8. 不支持IA类型Object转储为Standard。
  9. 不支持Archive类型Object转储为IA或Standard。

## 示例

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 qn6qrrqx02oawuk53otfjbyc:PYbzsdWSMrAIWAlMW8luWekJ8=
```

```
<?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>
</LifecycleConfiguration>
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu , 8 Jun 2017 13:08:38 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

## GetBucket

## Get Bucket (List Object)

Get Bucket操作可用来list Bucket中所有Object的信息。

### 请求语法

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

## 请求参数(Request Parameters)

GetBucket ( ListObject ) 时, 可以通过prefix, marker, delimiter和max-keys对list做限定, 返回部分结果。另外, 可以通过encoding-type对返回结果中的Delimiter、Marker、Prefix、NextMarker和Key这些元素进行编码。

| 名称            | 描述                                                                                                                                                                                                                                     |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| delimiter     | 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。<br>数据类型：字符串<br>默认值：无                                                                                                                                |
| marker        | 设定结果从marker之后按字母排序的第一个开始返回。<br>数据类型：字符串<br>默认值：无                                                                                                                                                                                       |
| max-keys      | 限定此次返回object的最大数, 如果不设定, 默认为100, max-keys取值不能大于1000。<br>数据类型：字符串<br>默认值：100                                                                                                                                                            |
| prefix        | 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。<br>数据类型：字符串<br>默认值：无                                                                                                                                                    |
| encoding-type | 指定对返回的内容进行编码, 指定编码的类型。Delimiter、Marker、Prefix、NextMarker和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、Marker、Prefix、NextMarker和Key进行编码。<br>数据类型：字符串<br>默认值：无, 可选值：url |

## 响应元素(Response Elements)

| 名称             | 描述                                                                                                                            |
|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| Contents       | 保存每个返回Object meta的容器。<br>类型：容器<br>父节点：ListBucketResult                                                                        |
| CommonPrefixes | 如果请求中指定了delimiter参数, 则在OSS返回的响应中包含CommonPrefixes元素。该元素标明那些以delimiter结尾, 并有共同前缀的object名称的集合。<br>类型：字符串<br>父节点：ListBucketResult |

|                  |                                                                                                                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Delimiter        | <p>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。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>                                                                                                              |
| EncodingType     | <p>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，那会对返回结果中的Delimiter、Marker、Prefix、NextMarker和Key这些元素进行编码。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>                                                                                                |
| DisplayName      | <p>Object 拥有者的名字。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents.Owner</p>                                                                                                                                                                   |
| ETag             | <p>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。不建议用户使用ETag来作为Object内容的MD5校验数据完整性。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents</p> |
| ID               | <p>Bucket拥有者的用户ID。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents.Owner</p>                                                                                                                                                                  |
| IsTruncated      | <p>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。</p> <p>类型：枚举字符串</p> <p>有效值：true、false</p> <p>父节点：ListBucketResult</p>                                                                                                                 |
| Key              | <p>Object的Key。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents</p>                                                                                                                                                                            |
| LastModified     | <p>Object最后被修改的时间。</p> <p>类型：时间</p> <p>父节点：ListBucketResult.Contents</p>                                                                                                                                                                         |
| ListBucketResult | <p>保存Get Bucket请求结果的容器。</p> <p>类型：容器 子节点：Name, Prefix, Marker, MaxKeys, Delimiter, IsTruncated, Nextmarker, Contents</p> <p>父节点：None</p>                                                                                                         |
| Marker           | <p>标明这次Get Bucket ( List Object ) 的起点。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>                                                                                                                                                             |
| MaxKeys          | <p>响应请求内返回结果的最大数目。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>                                                                                                                                                                                 |

|              |                                                                                                                          |
|--------------|--------------------------------------------------------------------------------------------------------------------------|
| Name         | Bucket名字<br>类型：字符串<br>父节点：ListBucketResult                                                                               |
| Owner        | 保存Bucket拥有者信息的容器。<br>类型：容器 子节点：DisplayName, ID<br>父节点：ListBucketResult                                                   |
| Prefix       | 本次查询结果的开始前缀。<br>类型：字符串<br>父节点：ListBucketResult                                                                           |
| Size         | Object的字节数。<br>类型：字符串<br>父节点：ListBucketResult.Contents                                                                   |
| StorageClass | Object的存储类型，支持 “Standard” 、<br>“IA” 、 “Archive” 。（目前只有部分区域支持<br>“Archive” 类型）<br>类型：字符串<br>父节点：ListBucketResult.Contents |

## 细节分析

1. Object中用户自定义的meta，在GetBucket请求时不会返回。
2. 如果访问的Bucket不存在，包括试图访问因为命名不规范无法创建的Bucket，返回404 Not Found错误，错误码：NoSuchBucket。
3. 如果没有访问该Bucket的权限，返回403 Forbidden错误，错误码：AccessDenied。
4. 如果因为max-keys的设定无法一次完成listing，返回结果会附加一个<NextMarker>，提示继续listing可以以此为marker。NextMarker中的值仍在list结果之中。
5. 在做条件查询时，即使marker实际在列表中不存在，返回也从符合marker字母排序的下一个开始打印。如果max-keys小于0或者大于1000，将返回400 Bad Request错误。错误码：InvalidArgument。
6. 若prefix，marker，delimiter参数不符合长度要求，返回400 Bad Request。错误码：InvalidArgument。
7. prefix，marker用来实现分页显示效果，参数的长度必须小于1024字节。
8. 如果把prefix设为某个文件夹名，就可以罗列以此prefix开头的文件，即该文件夹下递归的所有的文件和子文件夹。如果再把delimiter设置为 / 时，返回值就只罗列该文件夹下的文件，该文件夹下的子文件名返回在CommonPrefixes部分，子文件夹下递归的文件和文件夹不被显示。如一个bucket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 “/”，则返回文件 “fun/test.jpg” 和前缀 “fun/movie/”；即实现了文件夹的逻辑。

## 举例场景

在bucket “my\_oss” 内有4个object，名字分别为：

- oss.jpg



- fun/test.jpg
- fun/movie/001.avi
- fun/movie/007.avi

## 示例

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 qn6qrrqx02oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0ykboO4M=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
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>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
    <Type>Normal</Type>
    <Size>344606</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>00220120222</ID>
      <DisplayName>user-example</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>fun/movie/007.avi</Key>
    <LastModified>2012-02-24T08:43:27.000Z</LastModified>
    <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
    <Type>Normal</Type>
    <Size>344606</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>00220120222</ID>
```

```

<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/test.jpg</Key>
<LastModified>2012-02-24T08:42:32.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>oss.jpg</Key>
<LastModified>2012-02-24T06:07:48.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
</ListBucketResult>

```

#### 请求示例(含Prefix参数)：

```

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/ZghT7cGa0ykboO4M=

```

#### 返回示例：

```

HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
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>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</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&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/test.jpg</Key>
<LastModified>2012-02-24T08:42:32.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
</Contents>
</ListBucketResult>

```

#### 请求示例(含prefix和delimiter参数)：

```

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 qn6qqrxo2oawuk53otfjbyc:DNrn7xHk3sgysx7I8U9I9Y1vY=

```

#### 返回示例：

```

HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
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>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
    <Type>Normal</Type>
    <Size>344606</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>00220120222</ID>
      <DisplayName>user_example</DisplayName>
    </Owner>
  </Contents>
  <CommonPrefixes>
    <Prefix>fun/movie/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

# GetBucketAcl

Get Bucket ACL用来获取某个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(Response Elements)

| 名称                  | 描述                                             |
|---------------------|------------------------------------------------|
| AccessControlList   | 存储ACL信息的容器<br>类型：容器<br>父节点：AccessControlPolicy |
| AccessControlPolicy | 保存Get Bucket ACL结果的容器                          |

|             |                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------|
|             | 类型：容器<br>父节点：None                                                                                                   |
| DisplayName | Bucket拥有者的名称。(目前和ID一致)<br>类型：字符串<br>父节点：AccessControlPolicy.Owner                                                   |
| Grant       | Bucket的ACL权限。<br>类型：枚举字符串<br>有效值：private、public-read、public-read-write<br>父节点：AccessControlPolicy.AccessControlList |
| ID          | Bucket拥有者的用户ID<br>类型：字符串<br>父节点：AccessControlPolicy.Owner                                                           |
| Owner       | 保存Bucket拥有者信息的容器。<br>类型：容器<br>父节点：AccessControlPolicy                                                               |

## 细节分析

1. 只有Bucket的拥有者才能使用Get Bucket ACL这个接口。

## 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+WwIfNm0FmgbrQ0=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 04:11:23 GMT
Content-Length: 253
Content-Tupe: application/xml
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" ?>
<AccessControlPolicy>
  <Owner>
    <ID>00220120222</ID>
    <DisplayName>user_example</DisplayName>
  </Owner>
  <AccessControlList>
```

```
<Grant>public-read</Grant>
</AccessControlList>
</AccessControlPolicy>
```

# GetBucketLocation

Get Bucket Location用于查看Bucket所属的数据中心位置信息。

## 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(Response Elements)

| 名称                 | 描述                                                                                                                             |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------|
| LocationConstraint | Bucket所在的区域<br>类型：字符串<br>Values: oss-cn-hangzhou、oss-cn-qingdao、oss-cn-beijing、oss-cn-hongkong、oss-cn-shenzhen、oss-cn-shanghai |

## 细节分析

- 1. 只有Bucket的拥有者才能查看Bucket的Location信息，否则返回403 Forbidden错误,错误码：AccessDenied。
- 2. 目前LocationConstraint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；分别对应杭州数据中心，青岛数据中心，北京数据中心、香港数据中心、深圳数据中心、上海数据中心、美国硅谷数据中心、美国弗吉尼亚数据中心和亚太（新加坡）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:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

已设置LOG规则的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
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
>
```

# GetBucketInfo

Get Bucket Info操作用于查看bucket的相关信息。 包括如下内容：

- 创建时间
- 外网访问Endpoint
- 内网访问Endpoint
- bucket的拥有者信息
- bucket的ACL ( AccessControlList )

## 请求语法

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

## 响应元素(Response Elements)

| 名称         | 描述                                                |  |
|------------|---------------------------------------------------|--|
| BucketInfo | 保存Bucket信息内容的容器<br>类型：容器<br>子节点：Bucket节点<br>父节点：无 |  |

|                   |                                                                                                                       |  |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|--|
| Bucket            | 保存Bucket具体信息的容器<br>类型：容器<br>父节点：BucketInfo节点                                                                          |  |
| CreationDate      | Bucket创建时间。时间格式<br>2013-07-31T10:56:21.000Z<br>类型：时间<br>父节点：BucketInfo.Bucket                                         |  |
| ExtranetEndpoint  | Bucket访问的外网域名<br>类型：字符串<br>父节点：BucketInfo.Bucket                                                                      |  |
| IntranetEndpoint  | 同区域ECS访问Bucket的内网域名<br>类型：字符串<br>父节点：BucketInfo.Bucket                                                                |  |
| Location          | Bucket所在数据中心的区域<br>类型：字符串<br>父节点：BucketInfo.Bucket                                                                    |  |
| Name              | Bucket名字<br>类型：字符串<br>父节点：BucketInfo.Bucket                                                                           |  |
| Owner             | 用于存放Bucket拥有者信息的容器。<br>类型：容器<br>父节点：BucketInfo.Bucket                                                                 |  |
| ID                | Bucket拥有者的用户ID。<br>类型：字符串<br>父节点<br>：BucketInfo.Bucket.Owner                                                          |  |
| DisplayName       | Bucket拥有者的名称 (目前和ID一致)。<br>类型：字符串<br>父节点<br>：BucketInfo.Bucket.Owner                                                  |  |
| AccessControlList | 存储ACL信息的容器<br>类型：容器<br>父节点：BucketInfo.Bucket                                                                          |  |
| Grant             | Bucket的ACL权限。<br>类型：枚举字符串<br>有效值：private、public-read、public-read-write<br>父节点<br>：BucketInfo.Bucket.AccessControlList |  |

## 细节分析

1. 如果Bucket不存在，返回404错误。错误码：NoSuchBucket。



2. 只有Bucket的拥有者才能查看Bucket的信息，否则返回403 Forbidden错误,错误码：AccessDenied。
3. 请求可以从任何一个OSS的Endpoint发起。

## 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+zNhH1AcF51NNHD39zXw=
```

成功获取Bucket信息的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
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>271834739143143</ID>
    </Owner>
    <AccessControlList>
      <Grant>private</Grant>
    </AccessControlList>
  </Bucket>
</BucketInfo>
```

获取不存在的Bucket信息的返回示例：

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906009B
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>568D547F31243C673BA14274</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: 534B371674E88A4D8906008C
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>568D5566F2D0F89F5C0EB66E</RequestId>
  <HostId>test.oss.aliyuncs.com</HostId>
</Error>
```

## GetBucketLogging

Get Bucket Logging用于查看Bucket的访问日志配置情况。

### 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(Response Elements)

| 名称                  | 描述                                                     |
|---------------------|--------------------------------------------------------|
| BucketLoggingStatus | 访问日志状态信息的容器<br>类型: 容器<br>子元素: LoggingEnabled<br>父元素: 无 |

|                |                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------|
| LoggingEnabled | 访问日志信息的容器。这个元素在开启时需要，关闭时不需要。<br>类型: 容器<br>子元素: TargetBucket, TargetPrefix<br>父元素: BucketLoggingStatus |
| TargetBucket   | 指定存放访问日志的Bucket。<br>类型: 字符<br>子元素: 无<br>父元素: BucketLoggingStatus.LoggingEnabled                       |
| TargetPrefix   | 指定最终被保存的访问日志文件前缀。<br>类型: 字符<br>子元素: None<br>父元素: BucketLoggingStatus.LoggingEnabled                   |

## 细节分析

1. 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能查看Bucket的访问日志配置情况，否则返回403 Forbidden错误,错误码：AccessDenied。
3. 如果源Bucket未设置Logging规则，OSS仍然返回一个XML消息体，但其中的BucketLoggingStatus元素为空。

## 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:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

已设置LOG规则的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
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LOG规则的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
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>
```

## GetBucketWebsite

Get Bucket Website操作用于查看bucket的静态网站托管状态。

### 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(Response Elements)

| 名称            | 描述                                                                                                                        |
|---------------|---------------------------------------------------------------------------------------------------------------------------|
| ErrorDocument | 子元素Key的父元素<br>类型: 容器<br>父元素: WebsiteConfiguration                                                                         |
| IndexDocument | 子元素Suffix的父元素.<br>类型: 容器<br>父元素: WebsiteConfiguration                                                                     |
| Key           | 返回404错误时使用的文件名<br>类型:字符串 父元素:<br>WebsiteConfiguration.ErrorDocument<br>有条件：当ErrorDocument设置时，必需                           |
| Suffix        | 返回目录URL时添加的索引文件名，不要为空，也不要包含"/"。例如索引文件设置为index.html，则访问:oss-cn-hangzhou.aliyuncs.com/mybucket/mydir/这样请求的时候默认都相当于访问oss-cn- |

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
|                      | hangzhou.aliyuncs.com/mybucket/index.html<br>类型:字符串<br>父元素: WebsiteConfiguration.IndexDocument |
| WebsiteConfiguration | 请求的容器<br>类型: 容器<br>父元素: 无                                                                      |

## 细节分析

1. 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能查看Bucket的静态网站托管状态，否则返回403 Forbidden错误,错误码：AccessDenied。
3. 如果源Bucket未设置静态网站托管功能，OSS会返回404错误，错误码为：NoSuchWebsiteConfiguration。

## 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 qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39zXw=
```

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 configuration.</Message>
  <BucketName>oss-example</BucketName>
  <RequestId>505191BEC4689A033D00236F</RequestId>
  <HostId>oss-example.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>

```

## GetBucketReferer

Get Bucket Referer操作用于查看bucket的Referer相关配置。Bucket Referer防盗链具体见OSS防盗链。

### 请求语法

```

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

```

### 响应元素(Response Elements)

| 名称                   | 描述                                                                                        |
|----------------------|-------------------------------------------------------------------------------------------|
| RefererConfiguration | 保存Referer配置内容的容器<br>类型：容器<br>子节点：AllowEmptyReferer节点、RefererList节点<br>父节点：无               |
| AllowEmptyReferer    | 指定是否允许referer字段为空的请求访问。<br>类型：枚举字符串 有效值：true 或 false`默认值：true<br>父节点：RefererConfiguration |
| RefererList          | 保存referer访问白名单的容器。<br>类型：容器<br>父节点：RefererConfiguration<br>子节点：Referer                    |
| Referer              | 指定一条referer访问白名单。<br>类型：字符串                                                               |

|  |                 |
|--|-----------------|
|  | 父节点：RefererList |
|--|-----------------|

## 细节分析

1. 如果Bucket不存在，返回404错误。错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能查看Bucket的Referer配置信息，否则返回403 Forbidden错误,错误码：AccessDenied。
3. 如果Bucket未进行Referer相关配置，OSS会返回默认的AllowEmptyReferer值和空的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+zNhH1AcF51NNHD39zXw=
```

已设置Referer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"?>
<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：

```
HTTP/1.1 200
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"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

# GetBucketLifecycle

Get Bucket Lifecycle用于查看Bucket的Lifecycle配置。

## 请求语法

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

## 细节分析

1. 只有Bucket的拥有者才能查看Bucket的Lifecycle配置，否则返回403 Forbidden错误,错误码：AccessDenied。
2. 如果Bucket或Lifecycle不存在，返回404 Not Found错误，错误码：NoSuchBucket或NoSuchLifecycle。

## 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:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

已设置Lifecycle的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
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Lifecycle的返回示例：

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906008B
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>534B372974E88A4D89060099</RequestId>
  <HostId> oss-example.oss.aliyuncs.com</HostId>
</Error>
```

## DeleteBucket

Delete Bucket用于删除某个Bucket。

### 请求语法

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

## 细节分析

1. 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
2. 为了防止误删除的发生，OSS不允许用户删除一个非空的Bucket。
3. 如果试图删除一个不为空的Bucket，返回409 Conflict错误，错误码：BucketNotEmpty。
4. 只有Bucket的拥有者才能删除这个Bucket。如果试图删除一个没有对应权限的Bucket，返回403 Forbidden错误。错误码：AccessDenied。

## 示例

请求示例：

```
DELETE / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:31:04 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

返回示例：

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

# DeleteBucketLogging

Delete Bucket Logging操作用于关闭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
```

## 细节分析

1. 如果Bucket不存在，返回404 no content错误，错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能关闭Bucket访问日志记录功能。如果试图操作一个不属于你的Bucket，OSS返回403 Forbidden错误，错误码：AccessDenied。
3. 如果目标Bucket并没有开启Logging功能，仍然返回HTTP状态码 204。

## 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 qn6qrrxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/CnMZU=
```

### 返回示例

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:35:24 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

## DeleteBucketWebsite

Delete Bucket Website操作用于关闭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
```

## 细节分析

1. 如果Bucket不存在，返回404 no content错误，错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能关闭Bucket的静态网站托管模式。如果试图操作一个不属于你的Bucket，OSS返回403 Forbidden错误，错误码：AccessDenied。

## 示例

请求示例：

```
DELETE /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWicOMEkVg=
```

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
```

## DeleteBucketLifecycle

## Delete Bucket Lifecycle

通过Delete Bucket Lifecycle来删除指定Bucket的生命周期配置。

### 请求语法

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

### 细节分析

1. 本操作会删除指定Bucket的所有的生命周期规则。此后，该Bucket中不会有Object被自动删除。
2. 只有Bucket的拥有者才能删除Bucket的Lifecycle配置。如果试图操作一个不属于你的Bucket，OSS返回403 Forbidden错误，错误码：AccessDenied。

## 示例

请求示例：

```
DELETE /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:35 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxC1yxRydPQs/CnMZU=
```

返回示例：

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Mon, 14 Apr 2014 01:17:35 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

## 关于Object操作

### PutObject

### Put Object

Put Object用于上传文件。

### 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Header

| 名称                           | 描述                                                                                                                                                                  |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cache-Control                | 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。<br>类型：字符串<br>默认值：无                                                                                                           |
| Content-Disposition          | 指定该Object被下载时的名称；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                                                                                                                |
| Content-Encoding             | 指定该Object被下载时的内容编码格式；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                                                                                                            |
| Content-MD5                  | 根据协议RFC 1864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编码为一个消息的Content-MD5值。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）。虽然该请求头是可选项，OSS建议用户使用该请求头进行端到端检查。<br>类型：字符串<br>默认值：无<br>限制：无 |
| Expires                      | 过期时间；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无<br><b>注意：</b> OSS不会对这个值进行限制和验证                                                                                             |
| x-oss-server-side-encryption | 指定oss创建object时的服务器端加密编码算法。<br>类型：字符串<br>合法值：AES256                                                                                                                  |
| x-oss-object-acl             | 指定oss创建object时的访问权限。<br>类型：字符串<br>合法值：public-read, private, public-read-write                                                                                       |

## 细节分析

1. 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。
2. 如果请求头中的“Content-Length”值小于实际请求体（body）中传输的数据长度，OSS仍将成功创建文件；但Object大小只等于“Content-Length”中定义的大小，其他数据将被丢弃。
3. 如果试图添加的Object的同名文件已经存在，并且有访问权限。新添加的文件将覆盖原来的文件，成功返回200 OK。
4. 如果在PutObject的时候，携带以x-oss-meta-为前缀的参数，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8k。

5. 如果Head中没有加入Content length参数，会返回411 Length Required错误。错误码：MissingContentLength。
6. 如果设定了长度，但是没有发送消息Body，或者发送的body大小小于给定大小，服务器会一直等待，直到time out，返回400 Bad Request消息。错误码：RequestTimeout。
7. 如果试图添加的Object所在的Bucket不存在，返回404 Not Found错误。错误码：NoSuchBucket。
8. 如果试图添加的Object所在的Bucket没有访问权限，返回403 Forbidden错误。错误码：AccessDenied。
9. 如果添加文件长度超过5G，返回错误消息400 Bad Request。错误码：InvalidArgument。
10. 如果传入的Object key长度大于1023字节，返回400 Bad Request。错误码：InvalidObjectName。
11. PUT一个Object的时候，OSS支持5个 HTTP RFC2616协议规定的Header 字段：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type。如果上传Object时设置了这些Header，则这个Object被下载时，相应的Header值会被自动设置成上传时的值。
12. 如果上传Object时指定了x-oss-server-side-encryption Header，则必须设置其值为AES256，否则会返回400和相应错误提示：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，响应头中会包含x-oss-server-side-encryption，值被设置成该Object的加密算法。

## 常见问题

### Content-MD5计算方式错误

以上传的内容为 0123456789 为例，计算这个字符串的Content-MD5。

#### 1. 正确的计算方式

标准中定义的算法简单点说就是：

1. 先计算出上传内容的MD5，其值是一个128位（128 bit）的二进制数组。
2. 再对这个二进制数组进行base64编码。

以Python语言为例，正确计算的代码为：

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bjSPyxw=='
```

需要注意的是，这里要用 hash.digest() 的结果作为base64编码的输入。

```
>>> hash.digest()
'x\x1e^\$ji\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```

## 2. 错误的计算方式

常见的一个错误是对计算出的32字节 ( 32 bytes ) 的 MD5字符串表示 进行base64编码。例如，错误的对 hash.hexdigest() 进行base64编码：

```
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
>>>
>>> # 下面是错误的计算结果
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVIMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

## 3. 调查方法

可以用下述方法进行调查：

1. 最终计算出来的Content-MD5是一个24字节的可见字符串
2. 尝试用 0123456789 作为内容，确认Content-MD5是 eB5eJF1ptWaXm4bijSPyxw==

## 示例

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/jpeg
Content-Length: 344606
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=

[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-Length: 0
Connection: keep-alive
x-oss-request-id: 5650BD72207FB30443962F9A
x-oss-bucket-version: 1418321259
ETag: "A797938C31D59EDD08D86188F6D5B872"
```



# CopyObject

## Copy Object

拷贝一个在OSS上已经存在的object成另外一个object，可以发送一个PUT请求给OSS，并在PUT请求头中添加元素x-oss-copy-source来指定拷贝源。OSS会自动判断出这是一个Copy操作，并直接在服务器端执行该操作。如果拷贝成功，则返回新的object信息给用户。

该操作适用于拷贝小于1GB的文件，当拷贝一个大于1GB的文件时，必须使用Multipart Upload操作，具体见Upload Part Copy。

Copy操作的源Bucket和目标Bucket必须是同一个Region，不同Region的Bucket不能执行Copy Object操作。

### 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Header

| 名称                                    | 描述                                                                                     |
|---------------------------------------|----------------------------------------------------------------------------------------|
| x-oss-copy-source                     | 复制源地址（必须有可读权限）<br>类型：字符串<br>默认值：无                                                      |
| x-oss-copy-source-if-match            | 如果源Object的ETag值和用户提供的ETag相等，则执行拷贝操作，并返回200；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无  |
| x-oss-copy-source-if-none-match       | 如果源Object的ETag值和用户提供的ETag不相等，则执行拷贝操作，并返回200；否则返回304 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-unmodified-since | 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。<br>类型：字符串    |

|                                     |                                                                                                                                                                                                                                    |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     | 默认值：无                                                                                                                                                                                                                              |
| x-oss-copy-source-if-modified-since | 如果源Object自从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 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无                                                                                                                                                        |
| x-oss-metadata-directive            | 有效值为COPY和REPLACE。如果该值设为COPY，则新的Object的meta都从源Object复制过来；如果设为REPLACE，则忽视所有源Object的meta值，而采用用户这次请求中指定的meta值；其他值则返回400 HTTP错误码。注意该值为COPY时，源Object的x-oss-server-side-encryption的meta值不会进行拷贝。<br>类型：字符串<br>默认值：COPY<br>有效值：COPY、REPLACE |
| x-oss-server-side-encryption        | 指定oss创建目标object时的服务器端熵编码加密算法<br>类型：字符串<br>有效值：AES256                                                                                                                                                                               |
| x-oss-object-acl                    | 指定oss创建object时的访问权限。<br>类型：字符串<br>合法值：public-read，private，public-read-write                                                                                                                                                        |

## 响应元素(Response Elements)

| 名称               | 描述                                               |
|------------------|--------------------------------------------------|
| CopyObjectResult | Copy Object结果<br>类型：字符串<br>默认值：无                 |
| ETag             | 新Object的ETag值。<br>类型：字符串<br>父元素：CopyObjectResult |
| LastModified     | 新Object最后更新时间。<br>类型：字符串<br>父元素：CopyObjectResult |

## 细节分析

1. 可以通过拷贝操作来实现修改已有Object的meta信息。
2. 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meta信息。
3. OSS支持拷贝操作的四个预判断Header任意个同时出现，相应逻辑参见Get Object操作的细节分析。

4. 拷贝操作需要请求者对源Object有读权限。
5. 源Object和目标Object必须属于同一个数据中心，否则返回403 AccessDenied，错误信息为：Target object does not reside in the same data center as source object。
6. 拷贝操作的计费统计会对源Object所在的Bucket增加一次Get请求次数，并对目标Object所在的Bucket增加一次Put请求次数，以及相应的新增存储空间。
7. 拷贝操作涉及到的请求头，都是以“x-oss-”开头的，所以要加入签名字符串中。
8. 若在拷贝操作中指定了x-oss-server-side-encryption请求头，并且请求值合法（为AES256），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，值被设置成目标Object的加密算法。在这个目标Object被下载时，响应头中也会包含x-oss-server-side-encryption，值被设置成该Object的加密算法；若拷贝操作中未指定x-oss-server-side-encryption请求头，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都是未进行过服务器端加密编码加密的数据。
9. 拷贝操作中x-oss-metadata-directive请求头为COPY（默认值）时，并不拷贝源Object的x-oss-server-side-encryption值，即目标Object是否进行服务器端加密编码只根据COPY操作是否指定了x-oss-server-side-encryption请求头来决定。
10. 若在拷贝操作中指定了x-oss-server-side-encryption请求头，并且请求值非AES256，则返回400和相应的错误提示：InvalidEncryptionAlgorithmError。
11. 如果拷贝的文件大小大于1GB，会返回400和错误提示：EntityTooLarge。
12. 该操作不能拷贝通过Append追加上传方式产生的object。
13. 如果文件类型为符号链接，只拷贝符号链接。
14. 如果Bucket的类型为Archive，则不能调用该接口，否则返回400错误，错误码为OperationNotSupported。

## 示例

请求示例：

```
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-copy-source: /oss-example/oss.jpg
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:gmnwPKuu20LQEjd+iPkL259A+n0=
```

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>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
</CopyObjectResult>
```

# GetObject

# Get Object

用于获取某个Object，此操作要求用户对该Object有读权限。

## 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)
```

## 请求参数(Request Parameters)

OSS支持用户在发送GET请求时，自定义OSS返回响应中的一些Header。匿名用户请求仅可以对content-type进行自定义返回，其他Header则要求用户发送的GET请求必须携带签名。这些Header包括：

| 名称                           | 描述                                                    |
|------------------------------|-------------------------------------------------------|
| response-content-type        | 设置OSS返回请求的content-type头，匿名访问也可以设置。<br>类型：字符串<br>默认值：无 |
| response-content-language    | 设置OSS返回请求的content-language头<br>类型：字符串<br>默认值：无        |
| response-expires             | 设置OSS返回请求的expires头<br>类型：字符串<br>默认值：无                 |
| response-cache-control       | 设置OSS返回请求的cache-control头<br>类型：字符串<br>默认值：无           |
| response-content-disposition | 设置OSS返回请求的content-disposition头<br>类型：字符串              |

|                           |                                                |
|---------------------------|------------------------------------------------|
|                           | 默认值：无                                          |
| response-content-encoding | 设置OSS返回请求的content-encoding头<br>类型：字符串<br>默认值：无 |

## 请求Header

| 名称                  | 描述                                                                                                                                        |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Range               | 指定文件传输的范围。如，设定 bytes=0-9，表示传送第0到第9这10个字符。<br>类型：字符串<br>默认值：无                                                                              |
| If-Modified-Since   | 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<br>类型：字符串<br>默认值：无<br>时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT                   |
| If-Unmodified-Since | 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<br>类型：字符串<br>默认值：无<br>时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT |
| If-Match            | 如果传入期望的ETag和object的 ETag匹配，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误<br>类型：字符串<br>默认值：无                                             |
| If-None-Match       | 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<br>类型：字符串<br>默认值：无                                                       |

## 细节分析

1. GetObject通过range参数可以支持断点续传，对于比较大的Object建议使用该功能。
2. 如果在请求头中使用Range参数；则返回消息中会包含整个文件的长度和此次返回的范围，例如：Content-Range: bytes 0-9/44，表示整个文件长度为44，此次返回的范围为0-9。如果不符合范围规范，则传送整个文件，并且不在结果中提及Content-Range。
3. 如果 “If-Modified-Since” 元素中设定的时间不符合规范，直接返回文件，并返回200 OK。
4. If-Modified-Since和If-Unmodified-Since可以同时存在，If-Match和If-None-Match也可以同时存在。

5. 如果包含If-Unmodified-Since并且不符合或者包含If-Match并且不符合，返回412 precondition failed
6. 如果包含If-Modified-Since并且不符合或者包含If-None-Match并且不符合，返回304 Not Modified
7. 如果文件不存在，返回404 Not Found错误。错误码：NoSuchKey。
8. OSS不支持在匿名访问的GET请求中，通过请求参数来自定义返回请求的header。
9. 在自定义OSS返回请求中的一些Header时，只有请求处理成功（即返回码为200时），OSS才会将请求的header设置成用户GET请求参数中指定的值。
10. 若该Object为进行服务器端熵编码加密存储的，则在GET请求时会自动解密返回给用户，并且在响应头中，会返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
11. 需要将返回内容进行 GZIP压缩传输的用户，需要在请求的Header中显示方式加入 Accept-Encoding: gzip，OSS会根据文件的Content-Type和文件大小，判断是否返回给用户经过GZIP 压缩的数据。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，文件大小需不小于1k。
12. 如果文件类型为[符号链接](#)，返回目标文件的内容。并且，响应头中Content-Length、ETag、Content-Md5 均为目标文件的元信息；Last-Modified是目标文件和符号链接的最大值；其他均为符号链接的元信息。
13. 如果文件类型为[符号链接](#)，并且目标文件不存在，返回404 Not Found错误。错误码：SymlinkTargetNotExist。
14. 如果文件类型为[符号链接](#)，并且目标文件类型是符号链接，返回400 Bad request错误。错误码：InvalidTargetType。
15. 对于Archive归档类型，Object下载需要提交Restore请求，并等待Restore完成；只有在Object的Restore操作完成且超时前，Object才能被下载：
  - a. 如果没有提交Restore请求，或者上一次提交Restore已经超时，则返回403错，错误码为：InvalidObjectState。
  - b. 或者已经提交Restore请求，但数据的Restore操作还没有完成，则返回403错，错误码为：InvalidObjectState。
  - c. 只有Restore完成，且没有超时，数据才能直接下载。

## 示例

### 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 qn6qrrqx02oawuk53otfjbyc:UNQDb7GapEgJCZkcde6OhZ9Jfe8=
```

### 返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 3a89276f-2e2d-7965-3ff9-51c875b99c41
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: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
```

[344606 bytes of object data]

#### Range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 qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM=
```

#### 返回示例：

```
HTTP/1.1 206 Partial Content
x-oss-request-id: 28f6508f-15ea-8224-234e-c0ce40734b89
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: "5B3C1A2E053D763E1B002CC607C5A0FE "
Accept-Ranges: bytes
Content-Range: bytes 100-900/344606
Content-Type: image/jpg
Content-Length: 801
Server: AliyunOSS

[801 bytes of object data]
```

#### 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%25B%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: 559CC9BDC755F95A64485981
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: "5B3C1A2E053D763E1B002CC607C5A0FE "
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]

#### 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 qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM=
```

#### 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: 582143E6D3436A212ADCC87D
Accept-Ranges: bytes
ETag: "8086265EFC0211ED1F9A2F09BF462227"
Last-Modified: Tue, 08 Nov 2016 03:17:58 GMT
x-oss-object-type: Symlink
Content-MD5: gIYmXvwCEe0fmi8Jv0YiJw==
```

#### Archive类型Object的Restore操作已经完成时的请求示例：

```
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 qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+NIUs=
```

#### 返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 58F723894529F18D7F000053
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: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpeg
Content-Length: 344606
```



```
Server: AliyunOSS

[354606 bytes of object data]
```

# AppendObject

# Append Object

Append Object以追加写的方式上传文件。通过Append Object操作创建的Object类型为Appendable Object，而通过Put Object上传的Object是Normal Object。

## 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Header

| 名称                  | 描述                                                                                                                        |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|
| Cache-Control       | 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。<br>类型：字符串<br>默认值：无                                                                 |
| Content-Disposition | 指定该Object被下载时的名称；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                                                                      |
| Content-Encoding    | 指定该Object被下载时的内容编码格式；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                                                                  |
| Content-MD5         | 根据协议RFC 1864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编码为一个消息的Content-MD5值。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）。虽然该请求头是可选项，OSS建议用 |

|                              |                                                                                 |
|------------------------------|---------------------------------------------------------------------------------|
|                              | 户使用该请求头进行端到端检查。<br>类型：字符串<br>默认值：无<br>限制：无                                      |
| Expires                      | 过期时间；更详细描述请参照RFC2616。<br>类型：整数<br>默认值：无                                         |
| x-oss-server-side-encryption | 指定oss创建object时的服务器端加密编码算法。<br>类型：字符串<br>合法值：AES256                              |
| x-oss-object-acl             | 指定oss创建object时的访问权限。<br>类型：字符串<br>合法值：public-read , private , public-read-write |

## 响应Header

| 名称                         | 描述                                                                                                                |
|----------------------------|-------------------------------------------------------------------------------------------------------------------|
| x-oss-next-append-position | 指明下一次请求应当提供的position。实际上就是当前Object长度。当Append Object成功返回，或是因position和Object长度不匹配而引起的409错误时，会包含此header。<br>类型：64位整型 |
| x-oss-hash-crc64ecma       | 表明Object的64位CRC值。该64位CRC根据ECMA-182标准计算得出。<br>类型：64位整型                                                             |

## 和其他操作的关系

1. 不能对一个非Appendable Object进行Append Object操作。例如，已经存在一个同名Normal Object时，Append Object调用返回409，错误码ObjectNotAppendable。
2. 对一个已经存在的Appendable Object进行Put Object操作，那么该Appendable Object会被新的Object覆盖，类型变为Normal Object。
3. Head Object操作会返回x-oss-object-type，用于表明Object的类型。对于Appendable Object来说，该值为Appendable。对Appendable Object，Head Object也会返回上述的x-oss-next-append-position和x-oss-hash-crc64ecma。
4. Get Bucket ( List Objects ) 请求的响应XML中，会把Appendable Object的Type设为Appendable
5. 可以使用Copy Object来改变它的服务器端加密的属性或是用户自定义元信息。也可以把Appendable Object拷贝成另外一个文件，新的文件也是Appendable Object。

## 细节分析：

1. URL参数append和position均为CanonicalizedResource，需要包含在签名中。

2. URL的参数必须包含append，用来指定这是一个Append Object操作。
3. URL查询参数还必须包含position，其值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的当前长度。例如，第一次Append Object请求指定position值为0，content-length是65536；那么，第二次Append Object需要指定position为65536。每次操作成功后，响应头部x-oss-next-append-position也会标明下一次追加的position。
4. 如果position的值和当前Object的长度不一致，OSS会返回409错误，错误码为PositionNotEqualToLength。发生上述错误时，用户可以通过响应头部x-oss-next-append-position来得到下一次position，并再次进行请求。
5. 当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。
6. 当Position值为0，且没有同名Object存在，那么Append Object可以和Put Object请求一样，设置诸如x-oss-server-side-encryption之类的请求Header。这一点和Initiate Multipart Upload是一样的。如果在Position为0的请求时，加入了正确的x-oss-server-side-encryption头，那么后续的Append Object响应头部也会包含x-oss-server-side-encryption头，其值表明加密算法。后续如果需要更改meta，可以使用Copy Object请求。
7. 由于并发的关系，即使用户把position的值设为了x-oss-next-append-position，该请求依然可能因为PositionNotEqualToLength而失败。
8. Append Object产生的Object长度限制和Put Object一样。
9. 每次Append Object都会更新该Object的最后修改时间。
10. 在position值正确的情况下，对已存在的Appendable Object追加一个长度为0的内容，该操作不会改变Object的状态。
11. 如果Bucket的类型为Archive，则不能调用该接口，否则返回400错误，错误码为OperationNotSupported。

## CRC64的计算方式

Appendable Object的CRC采用ECMA-182标准，和XZ的计算方式一样。用Boost CRC模块的方式来定义则有：

```
typedef boost::crc_optimal<64, 0x42F0E1EBA9EA3693ULL, 0xffffffffffffffffULL, 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();
}
```

或是用Python crcmod的方式为：

```
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
Content-Disposition: attachment;filename=oss_download.jpg
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpeg
Content-Length: 1717
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRk=

[1717 bytes of object data]
```

返回示例：

```
HTTP/1.1 200 OK
Date: Wed, 08 Jul 2015 06:57:01 GMT
ETag: "0F7230CAA4BE94CCBDC99C5500000000"
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
x-oss-hash-crc64ecma: 14741617095266562575
x-oss-next-append-position: 1717
x-oss-request-id: 559CC9BDC755F95A64485981
```

## DeleteObject

DeleteObject用于删除某个Object。

## 请求语法

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

## 细节分析

1. DeleteObject要求对该Object要有写权限。
2. 如果要删除的Object不存在，OSS也返回状态码204（No Content）。
3. 如果Bucket不存在，返回404 Not Found。
4. 如果文件类型为**符号链接**，只删除符号链接自身。

## 示例

请求示例：

```
DELETE /copy_oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 07:45:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+NIUs=
```

返回示例：

```
HTTP/1.1 204 NoContent
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Fri, 24 Feb 2012 07:45:28 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

## DeleteMultipleObjects

## Delete Multiple Objects

Delete Multiple Objects操作支持用户通过一个HTTP请求删除同一个Bucket中的多个Object。Delete Multiple Objects操作支持一次请求内最多删除1000个Object，并提供两种返回模式：详细(verbose)模式和简单(quiet)模式：

- 详细模式：OSS返回的消息体中会包含每一个删除Object的结果。
- 简单模式：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的话，则没有消息体。

## 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>
```

## 请求参数(Request Parameters)

Delete Multiple Objects时，可以通过encoding-type对返回结果中的Key进行编码。

| 名称            | 描述                                                                                                                                                                    |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| encoding-type | 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。<br>数据类型：字符串<br>默认值：无<br>可选值：url |

## 请求元素(Request Elements)

| 名称     | 描述                                                                                   |
|--------|--------------------------------------------------------------------------------------|
| Delete | 保存Delete Multiple Object请求的容器。<br>类型：容器<br>子节点：一个或多个Object元素，可选的Quiet元素<br>父节点：None. |
| Key    | 被删除Object的名字。<br>类型：字符串<br>父节点：Object                                                |
| Object | 保存一个Object信息的容器。<br>类型：容器<br>子节点：key<br>父节点：Delete                                   |
| Quiet  | 打开“简单”响应模式的开关。<br>类型：枚举字符串<br>有效值：true、false<br>默认值：false<br>父节点：Delete              |

## 响应元素(Response Elements)

| 名称           | 描述                                                                           |
|--------------|------------------------------------------------------------------------------|
| Deleted      | 保存被成功删除的Object的容器。<br>类型：容器<br>子节点：Key<br>父节点：DeleteResult                   |
| DeleteResult | 保存Delete Multiple Object请求结果的容器。<br>类型：容器<br>子节点：Deleted<br>父节点：None         |
| Key          | OSS执行删除操作的Object名字。<br>类型：字符串<br>父节点：Deleted                                 |
| EncodingType | 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。<br>类型：字符串<br>父节点：容器 |

## 细节分析

1. Delete Multiple Objects请求必须填Content-Length和Content-MD5字段。OSS会根据这些字段验证收到的消息体是正确的，之后才会执行删除操作。
2. 生成Content-MD5字段内容方法：首先将Delete Multiple Object请求内容经过MD5加密后得到一个128位字节数组；再将该字节数组用base64算法编码；最后得到的字符串即是Content-MD5字段内容。
3. Delete Multiple Objects请求默认是详细(verbose)模式。
4. 在Delete Multiple Objects请求中删除一个不存在的Object，仍然认为是成功的。
5. Delete Multiple Objects的消息体最大允许2MB的内容，超过2MB会返回MalformedXML错误码。
6. Delete Multiple Objects请求最多允许一次删除1000个Object；超过1000个Object会返回MalformedXML错误码。
7. 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：

```
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/jEfbfu8=
```

```
<?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>
```

#### 请求示例II：

```
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:WuV0Jks8RyGSNQrBca64kEExJDs=

<?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

```

## HeadObject

## Head Object

Head Object只返回某个Object的meta信息，不返回文件内容。

## 请求语法

```

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

```

## 请求Header

| 名称                | 描述                                                                            |
|-------------------|-------------------------------------------------------------------------------|
| If-Modified-Since | 如果指定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<br>类型：字符串<br>默认值：无 |

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| If-Unmodified-Since | 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错误<br>类型：字符串<br>默认值：无   |
| If-Match            | 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错误<br>类型：字符串<br>默认值：无 |
| If-None-Match       | 如果传入的ETag值和Object的ETag不匹配，则返回200 OK和Object Meta；否则返回304 Not Modified<br>类型：字符串<br>默认值：无           |

## 细节分析

1. 不论正常返回200 OK还是非正常返回，Head Object都不返回消息体。
2. HeadObject支持在头中设定If-Modified-Since, If-Unmodified-Since, If-Match , If-None-Match的查询条件。具体规则请参见GetObject中对应的选项。如果没有修改，返回304 Not Modified。
3. 如果用户在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。
4. 如果文件不存在返回404 Not Found错误。
5. 若该Object为进行服务器端熵编码加密存储的，则在Head请求响应头中，会返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
6. 如果文件类型为**符号链接**，响应头中Content-Length、ETag、Content-Md5 均为目标文件的元信息；Last-Modified是目标文件和符号链接的最大值；其他均为符号链接元信息。
7. 如果文件类型为**符号链接**，并且目标文件不存在，返回404 Not Found错误。错误码：SymlinkTargetNotExist。
8. 如果文件类型为**符号链接**，并且目标文件类型是符号链接，返回400 Bad request错误。错误码：InvalidTargetType。
9. x-oss-storage-class展示Object的存储类型：Standard，IA，Archive
10. 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：
  - a. 如果没有提交Restore或者Restore已经超时，则不返回该字段。
  - b. 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" 。
  - c. 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" 。

## 示例

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:JbzF2LxZUtanIJ5dLA092wpDC/E=
```

#### 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: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
Content-Type: image/jpeg
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 e1Unnbm1rgdnpl:KKxkdNrUBu2t1kqIDh0MLbDb99I=
```

#### 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: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
Content-Type: image/jpeg
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 e1Unnbm1rgdnpl:21qtGJ+ykDVmdu6O6FMJnn+WuBw=
```

#### 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: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
```

## GetObjectMeta

Get Object Meta用来获取某个Bucket下的某个Object的基本meta信息，包括该Object的ETag、Size（文件大小）、LastModified，并不返回其内容。

### 请求语法

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

### 细节分析

细节分析：

1. 无论正常返回还是非正常返回，Get Object Meta均不返回消息体。
2. Get Object Meta需包含objectMeta请求参数，否则表示Get Object请求。
3. 如果文件不存在返回404 Not Found错误。
4. Get Object Meta相比Head Object更轻量，仅返回指定Object的少量基本meta信息，包括该Object的ETag、Size（文件大小）、LastModified，其中Size由响应头Content-Length的数值表示。
5. 如果文件类型为[符号链接](#)，返回符号链接自身信息。

### 示例

请求示例：

```
GET /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+WwIfNm0FmgbrQ0=
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE"
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 344606
Connection: keep-alive
Server: AliyunOSS
```

# PutObjectACL

## Put Object ACL

Put Object ACL接口用于修改Object的访问权限。目前Object有四种访问权限：default，private，public-read，public-read-write。Put Object ACL操作通过Put请求中的“x-oss-object-acl”头来设置，这个操作只有Bucket Owner有权限执行。如果操作成功，则返回200；否则返回相应的错误码和提示信息。

### 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
```

### Object ACL释义

| 名称          | 描述                                                                    |
|-------------|-----------------------------------------------------------------------|
| private     | 该ACL表明某个Object是私有资源，即只有该Object的Owner拥有该Object的读写权限，其他的用户没有权限操作该Object |
| public-read | 该ACL表明某个Object是公共读资源，即非Object Owner只有该Object的读权限，而Object              |

|                   |                                                          |
|-------------------|----------------------------------------------------------|
|                   | Owner拥有该Object的读写权限                                      |
| public-read-write | 该ACL表明某个Object是公共读写资源，即所有用户拥有对该Object的读写权限               |
| default           | 该ACL表明某个Object是遵循Bucket读写权限的资源，即Bucket是什么权限，Object就是什么权限 |

## 细节分析：

- Object的读操作包括：GetObject，HeadObject，CopyObject和UploadPartCopy中的对source object的读；Object的写操作包括：  
：PutObject，PostObject，AppendObject，DeleteObject，DeleteMultipleObjects，CompleteMultipartUpload以及CopyObject对新的Object的写。
- x-oss-object-acl中权限的值必须在上述4种权限中。如果有不属于上述4种的权限，OSS返回400 Bad Request消息，错误码：InvalidArgument。
- 用户不仅可以通过PutObjectACL接口来设置Object ACL，还可以在Object的写操作时，在请求头上带上x-oss-object-acl来设置Object ACL，效果与PutObjectA ACL等同。例如PutObject时在请求头上带上x-oss-object-acl可以在上传一个Object的同时设置某个Object的ACL。
- 对某个Object没有读权限的用户读取某个Object时，OSS返回 403 Forbidden消息，错误码：AccessDenied，提示：You do not have read permission on this object.
- 对某个Object没有写权限的用户写某个Object时，OSS返回 403 Forbidden消息，错误码：AccessDenied，提示：You do not have write permission on this object.
- 只有Bucket Owner才有权调用PutObjectACL来修改该Bucket下某个Object的ACL。非Bucket Owner调用PutObjectACL时，OSS返回 403 Forbidden消息，错误码：AccessDenied，提示：You do not have write acl permission on this object.
- Object ACL优先级高于Bucket ACL。例如Bucket ACL是private的，而Object ACL是public-read-write的，则访问这个Object时，先判断Object的ACL，所以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。

## 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:KU5h8YMUC78M30dXqf3JxrTZHiA=
```

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
```

## GetObjectACL

Get Object ACL用来获取某个Bucket下的某个Object的访问权限。

### 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(Response Elements)

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

## 细节分析

1. 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，非Bucket Owner调用该接口时，返回403 Forbidden消息。错误码：AccessDenied，提示You do not have read acl permission on this object.
2. 如果从来没有对某个Object设置过ACL，则调用GetObjectACL时，OSS返回的ObjectACL会是default，表明该Object ACL遵循Bucket ACL。即：如果Bucket是private的，则该object也是private的；如果该object是public-read-write的，则该object也是public-read-write的。

## 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+WwIfNm0FmgbrQ0=
```

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-Type: 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>
```

## PostObject



# Post Object

Post Object使用HTML表单上传文件到指定bucket。Post作为Put的替代品，使得基于浏览器上传文件到bucket成为可能。Post Object的消息实体通过多重表单格式（multipart/form-data）编码，在Put Object操作中参数通过HTTP请求头传递，在Post操作中参数则作为消息实体中的表单域传递。

## Post object

### 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--

```

## 表单域

| 名称                                                                          | 描述                                                                                                                                                                                                 | 必须   |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| OSSAccessKeyId                                                              | Bucket 拥有者的Access Key Id。<br>类型：字符串<br>默认值：无<br>限制：当bucket非public-read-write或者提供了policy（或Signature）表单域时，必须提供该表单域。                                                                                  | 有条件的 |
| policy                                                                      | policy规定了请求的表单域的合法性。不包含policy表单域的请求被认为是匿名请求，并且只能访问public-read-write的bucket。更详细描述请参考下文 Post Policy。<br>类型：字符串<br>默认值：无<br>限制：当bucket非public-read-write或者提供了OSSAccessKeyId（或Signature）表单域时，必须提供该表单域。 | 有条件的 |
| Signature                                                                   | 根据Access Key Secret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。<br>类型：字符串<br>默认值：无<br>限制：当bucket非public-read-write或者提供了OSSAccessKeyId（或policy）表单域时，必须提供该表单域。                | 有条件的 |
| Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires | REST请求头，更多的信息见Put Object。<br>类型：字符串<br>默认值：无                                                                                                                                                       | 可选   |
| file                                                                        | 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，会覆盖用户的设置。                                                                                                                                         | 必须   |

|                              |                                                                                                                                                                                                                                                                    |    |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|                              | OSS一次只能上传一个文件。<br>类型：字符串<br>默认值：无                                                                                                                                                                                                                                  |    |
| key                          | 上传文件的object名称。如果需要使用用户上传的文件名称作为object名，使用\${filename}变量。例如：如果用户上传了文件b.jpg，而key域的值设置为/user/a/\${filename}，最终的object名为/user/a/b.jpg。如果文件名包含路径，则去除文件名中的路径，例如用户上传了文件a/b/c/b.jpg，则取文件名为b.jpg，若key域的值设置为/user/a/\${filename}，最终的object名为/user/a/b.jpg<br>类型：字符串<br>默认值：无 | 必须 |
| success_action_redirect      | 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。<br>类型：字符串<br>默认值：无                                                                                                                                                               | 可选 |
| success_action_status        | 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。接受值为200, 201, 204（默认）。如果该域的值设置为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。如果其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。<br>类型：字符串<br>默认值：无                                                           | 可选 |
| x-oss-meta-*                 | 用户指定的user meta值。OSS不会检查或者使用该值。<br>类型：字符串<br>默认值：无                                                                                                                                                                                                                  | 可选 |
| x-oss-server-side-encryption | 指定OSS创建object时的服务器端加密编码算法。<br>类型：字符串<br>合法值：AES256                                                                                                                                                                                                                 | 可选 |
| x-oss-object-acl             | 指定oss创建object时的访问权限。<br>类型：字符串<br>合法值：public-                                                                                                                                                                                                                      | 可选 |

|                      |                                                                                                                                   |    |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------|----|
|                      | read , private , public-read-write                                                                                                |    |
| x-oss-security-token | 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。<br>类型：字符串<br>默认值：无 | 可选 |

## 响应Header

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

## 响应元素(Response Elements)

| 名称           | 描述                                                                                                                            |
|--------------|-------------------------------------------------------------------------------------------------------------------------------|
| PostResponse | 保持Post请求结果的容器。<br>类型：容器<br>子节点：Bucket, ETag, Key, Location                                                                    |
| Bucket       | Bucket名称。<br>类型：字符串<br>父节点：PostResponse                                                                                       |
| ETag         | 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。<br>类型：字符串<br>父节点：PostResponse |
| Location     | 新创建Object的URL。<br>类型：字符串<br>父节点：PostResponse                                                                                  |

## 细节分析

1. 进行Post操作要求对bucket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。
2. 无论bucket是否为public-read-write，一旦上传OSSAccessKeyId, policy, Signature表单域中的任何一个，则另两个表单域为必选项，缺失时OSS会返回错误码：InvalidArgument。

3. 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。
4. 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
5. policy规定了该次Post请求中表单域的合法值，OSS会根据policy判断请求的合法性，如果不合法会返回错误码：AccessDenied。在检查policy合法性时，policy中不涉及的表单域不进行检查。
6. 表单和policy必须使用UTF-8编码，policy为经过UTF-8编码和base64编码的JSON。
7. Post请求中可以包含额外的表单域，OSS会根据policy对这些表单域检查合法性。
8. 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。
9. 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。
10. 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8k。
11. Post请求的body总长度不允许超过5G。若文件长度过大，会返回错误码：EntityTooLarge。
12. 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，否则会返回400和错误码：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，响应头中会包含x-oss-server-side-encryption，值被设置成该Object的加密算法。
13. 表单域为大小写不敏感的，但是表单域的值的大小写敏感的。

## 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/${filename}
--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
```

**返回示例：**

## Post Policy

## policy示例

93

Post policy中必须包含expiration和condtions。

## 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。注意：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。例如，如果设置key表单域为user/user1/\${filename}，用户的文件名为a.txt，则Post policy应当设置成[ “eq” , “\$key” , “user/user1/\${filename}” ]，而不是[ “eq” , “\$key” , “\$key” , “user/user1/a.txt” ]。Policy中支持的conditions项见下表：

| 名称                                                                          | 描述                                                                        |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------|
| content-length-range                                                        | 上传文件的最小和最大允许大小。该condition支持content-length-range匹配方式。                      |
| Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires | HTTP请求头。该condition支持精确匹配和starts-with匹配方式。                                 |
| key                                                                         | 上传文件的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。                           |
| 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请求中包含其他的表单域，可以将这些额外的表单域加入到policy的conditions中,conditions不涉及的表单域将不会进行合法性检查。

## 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         | 指定所允许上传的文件最大大小和最小大小，例如允许的文件大小为1到10字节：[ “content-                                           |

length-range" , 1, 10]

## 转义字符

于在 Post policy 中 \$ 表示变量，所以如果要描述 \$，需要使用转义字符\ \$。除此之外，JSON 将对一些字符进行转义。下图描述了 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控制请求中那些值是允许的。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 表单域该填入的值，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Head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：[点击这里](#)

## Callback



# 上传回调

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。现在支持CallBack的API 接口有：PutObject、PostObject、CompleteMultipartUpload。

## 构造CallBack参数

CallBack参数是由一段经过base64编码的Json字符串，用户关键需要指定请求回调的服务器URL ( callbackUrl ) 以及回调的内容 ( callbackBody )。详细的Json字段如下：

| 字段           | 含义                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| callbackUrl  | <ul style="list-style-type: none"> <li>- 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3MB。</li> <li>- 支持同时配置最多5个url，以“;”分割。OSS会依次发送请求直到第一个返回成功为止。</li> <li>- 如果没有配置或者值为空则认为没有配置callback。</li> <li>- 支持HTTPS地址。</li> <li>- 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&amp;中文名称=中文值 需要编码成http://example.com/%E4%B8%AD%E6%96%87.php?key=value&amp;%E4%B8%AD%E6%96%87%E5%90%8D%E7%A7%B0=%E4%B8%AD%E6%96%87%E5%80%BC</li> </ul> | 必选项 |
| callbackHost | <ul style="list-style-type: none"> <li>- 发起回调请求时Host头的值，只有在设置了callbackUrl时才有效。</li> <li>- 如果没有配置callbackHost，则会解析callbackUrl中的url并将解析出的host填充到callbackHost中</li> </ul>                                                                                                                                                                                                                                                                                                                                                                        | 可选项 |
| callbackBody | <ul style="list-style-type: none"> <li>- 发起回调时请求body的值，例如：<br/>key=\$(key)&amp;etag=\$(etag)&amp;my_var=\$(x:my_var)。</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                              | 必选项 |

|                  |                                                                                                                                                                                                     |     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                  | - 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。                                                                            |     |
| 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。<br>- 如果为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。 | 可选项 |

示例json串如下

```
{
  "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}"
}
```

其中callbackBody中可以设置的系统变量有，其中imageInfo针对于图片格式，如果为非图片格式都为空：

| 系统变量             | 含义                                            |
|------------------|-----------------------------------------------|
| bucket           | bucket                                        |
| object           | object                                        |
| etag             | 文件的etag，即返回给用户的etag字段                         |
| size             | object大小，CompleteMultipartUpload时为整个object的大小 |
| 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 |
|------------------|----------------|

## 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这个Map。在OSS发起POST回调请求的时候，会将这些参数和上一节所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。**这里有个特别需要注意的地方就是，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"
}
```

## 构造回调请求

构造完成上述的callback和callback-var两个参数之后，一共有三种方式传给OSS。其中callback为必填参数，callback-var为可选参数，如果没有自定义参数的话可以不用添加callback-var字段。这三种方式如下：

- 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，**在使用POST请求上传Object的时候只能使用这种方式来指定回调参数。**

这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head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。如下示例：

```
PUT /test.txt HTTP/1.1
Host: callback-test.oss-test.aliyun-inc.com
Accept-ncoding: identity
Content-Length: 5
```

```
x-oss-callback-var: eyJ4Om15X3Zhci6ImZvci1jYWxsYmFjay10ZXN0In0=
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:
eyJjYWxsYmFja1VyYbCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZlZlHkiOiJidWNRZX
Q9JHtidWNRZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pb
WVUeXBIfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlYWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1
hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV
92YXJ9In0=
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 mlepu3zr4u7b14:5a74vhd4UXpmyudV14Kaen5cY4=

Test
```

- 如果需要在POST上传Object的时候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，如下面的示例：

```
--9431149156168
Content-Disposition: form-data; name="callback"

eyJjYWxsYmFja1VyYbCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZlZlHkiOiJidWNRZX
Q9JHtidWNRZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pb
WVUeXBIfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlYWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1
hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV
92YXJ9In0=
```

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，举个例子，如果用户的自定义参数的json为

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

那么POST请求的表单域应该如下：

```
--9431149156168
Content-Disposition: form-data; name="callback"

eyJjYWxsYmFja1VyYbCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZlZlHkiOiJidWNRZX
Q9JHtidWNRZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pb
WVUeXBIfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlYWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1
hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV
92YXJ9In0=

--9431149156168
Content-Disposition: form-data; name="x:var1"

value1

--9431149156168
Content-Disposition: form-data; name="x:var2"
```

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：

100

再给客户端的内容为：

```
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"}
```

需要注意的是，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的内容如{"a":"b"}，希望对此处做好判断处理。

## 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，那么可以通过在回调中带上签名来验证OSS的身份。

## 生成签名

签名在OSS端发生，采用RSA非对称方式签名，私钥加密的过程为：

```
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，所以签名过程由以下几步组成：

- 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
- 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
- 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

如下例：

```
POST /index.php?id=1&index=2 HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 18
authorization:
kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzI2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
```

```
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==
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+xYMSrmhMZjzI2/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：

- 回调请求的x-oss-pub-key-url头保存的是公钥的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))
```

这里需要注意，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，目的是为了保证这个publickey是由OSS颁发的。

- 获取base64解码后的签名

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

- 获取待签名字符串，方法与签名一致

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

- 验证签名

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

以上例为例：

- 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
- 签名头

- kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQ  
DjC3uG33esH2txA==做base64解码（由于为非打印字符，无法显示出解码后的结果）
- 获取待签名字符串，即url\_decode( "index.php" ) + "?id=1&index=2" + "\n" + "bucket=yonghu-test" ，并做md5
  - 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 according 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()
```

其它语言实现的应用服务器如下：

#### 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。

#### C#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`

## 特别须知

如果传入的callback或者callback-var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5KB）
- callback或者callback-var没有经过base64编码
- callback或者callback-var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"...}

如果回调失败，则返回203，错误码为" CallbackFailed"，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），另外，此时文件已经成功上传到了OSS

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。

## PutSymlink

## Put Symlink

Put Symlink可用于针对OSS上的TargetObject创建符号链接，用户可以通过该符号链接访问TargetObject。具体使用说明，请参考下文。

### 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Header

| 名称                   | 描述                                         |
|----------------------|--------------------------------------------|
| x-oss-symlink-target | 符号链接指向的目标文件。<br>类型：字符串<br>合法值：命名规范同Object。 |

## 细节分析

1. TargetObjectName同ObjectName一样，需要URL encode。
2. 符号链接的目标文件类型不能为符号链接。
3. 创建符号链接时，
  - 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以上检查，都推迟到GetObject等需要访问目标文件的API。
4. 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，成功返回200 OK。
5. 如果在PutSymlink的时候，携带以x-oss-meta-为前缀的参数，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8k。
6. 如果Bucket的类型为Archive，则不能调用该接口，否则返回400错误，错误码为OperationNotSupported。

## 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 qn6qrrxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=
x-oss-symlink-target: oss.jpg
```

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: "0A477B89B4602AA8DECB8E19BFD447B6"
```

## GetSymlink

用于获取符号链接，此操作要求用户对该符号链接有读权限。

## 请求语法

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

## 响应Header

| 名称                   | 描述                     |
|----------------------|------------------------|
| x-oss-symlink-target | 符号链接指向的目标文件。<br>类型：字符串 |

## 细节分析

1. 如果符号链接不存在返回404 Not Found错误。错误码：NoSuchKey

## 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:UNQDb7GapEgJCZkcde6OhZ9Jfe8=
```

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: "A797938C31D59EDD08D86188F6D5B872"
```

# RestoreObject

## Restore Object

针对Archive类型的Object读取，需要使用Restore操作让服务端执行解冻任务。如果一个Object是Standard或者IA类型，不要调用该接口。归档Object在执行Restore前后的状态变换过程如下：

- 一个Archive类型的Object初始时处于冷冻状态；
- 提交一次Restore操作后，Object将处于解冻中，服务端执行解冻；
- 待服务端执行完成解冻任务后，Object就能够进入解冻状态，此时用户可以读取Object；
- 解冻状态默认持续1天，用户可以延长到最多7天，之后，Object又回到初始时的冷冻状态。

对一个处于冷冻状态的归档Object执行Restore操作，会产生数据取回费用；如果一个归档Object处于解冻中或者解冻完成状态时，再次提交Restore，则不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
```

### 细节分析

- 如果是针对该Object第一次调用restore接口，则返回202。
- 如果已经成功调用过restore接口，且服务端仍处于解冻中，再次调用时返回409, 错误码为：RestoreAlreadyInProgress；服务端返回该错误，代表服务端正在执行restore操作，用户只需要等待作业完成，最长等待时间4小时。
- 如果已经成功调用过restore接口，且服务端解冻已经完成，再次调用时返回200，且会将object的可下载时间延长一天，最多延长7天。
- 如果object不存在，则返回404。
- 如果针对非Archive类型的Object提交restore，则返回400错，错误码为：OperationNotSupported。

### 示例

#### 首次提交restore的请求示例

```
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 e1Unnbm1rgdnpl:y4eyu+4yje5ioRCr5PB=
```

#### 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
```

#### 再次调用，且restore没有完成时：

```
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 e1Unnbm1rgdnpl:21qtGJ+ykDVmdy4eyu+NIUs=
```

#### 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>10.101.200.203</HostId>
</Error>
```

#### 再次调用，且restore已经完成时：

```
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 e1Unnbm1rgdnpl:u6O6FMJnn+WuBwbByZxm1+y4eyu+NIUs=
```

#### 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
```

# 关于MultipartUpload的操作

## 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。用户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，如：

- 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。

## 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。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。用户可以根据这个ID来发起相关的操作，如中止Multipart Upload、查询Multipart Upload等。

## 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(Request Parameters)

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

| 名称 | 描述 |
|----|----|
|----|----|



|               |                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| encoding-type | 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。<br>数据类型：字符串<br>默认值：无,可选值：url |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 请求Header

| 名称                           | 描述                                                                                    |
|------------------------------|---------------------------------------------------------------------------------------|
| Cache-Control                | 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。<br>类型：字符串<br>默认值：无                             |
| Content-Disposition          | 指定该Object被下载时的名称；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                                  |
| Content-Encoding             | 指定该Object被下载时的内容编码格式；更详细描述请参照RFC2616。<br>类型：字符串<br>默认值：无                              |
| Expires                      | 过期时间（ milliseconds ）；更详细描述请参照RFC2616。<br>类型：整数<br>默认值：无                               |
| x-oss-server-side-encryption | 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。<br>类型：字符串<br>合法值：AES256 |

## 响应元素(Response Elements)

| 名称                            | 描述                                                                                    |
|-------------------------------|---------------------------------------------------------------------------------------|
| Bucket                        | 初始化一个Multipart Upload事件的Bucket名称。<br>类型：字符串<br>父节点：InitiateMultipartUploadResult      |
| InitiateMultipartUploadResult | 保存Initiate Multipart Upload请求结果的容器。<br>类型：容器<br>子节点：Bucket, Key, UploadId<br>父节点：None |
| Key                           | 初始化一个Multipart Upload事件的Object名称。<br>类型：字符串                                           |

|              |                                                                              |
|--------------|------------------------------------------------------------------------------|
|              | 父节点：InitiateMultipartUploadResult                                            |
| UploadId     | 唯一标示此次Multipart Upload事件的ID。<br>类型：字符串<br>父节点：InitiateMultipartUploadResult  |
| EncodingType | 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。<br>类型：字符串<br>父节点：容器 |

## 细节分析

1. 该操作计算认证签名的时候，需要加 “?uploads” 到CanonicalizedResource中。
2. 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PUT Object接口。
3. 初始化Multipart Upload请求，并不会影响已经存在的同名object。
4. 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。该请求体内有三个元素：Bucket，Key和UploadID。请记录下其中的UploadID，以用于后续的Multipart相关操作。
5. 初始化Multipart Upload请求时，若设置了x-oss-server-side-encryption Header，则在响应头中会返回该Header，并且在上传的每个part时，服务端会自动对每个part进行熵编码加密存储，目前OSS服务器端只支持256位高级加密标准（AES256），指定其他值会返回400和相应的错误提示：InvalidEncryptionAlgorithmError；在上传每个part时不必再添加x-oss-server-side-encryption 请求头，若指定该请求头则OSS会返回400和相应的错误提示：InvalidArgument。

## 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
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:/cluRfRwMTZpC2hTj4F67AGdM4=
```

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-bda01eaec618
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>0004B9894A22E5B1888A1E29F8236E2D</UploadId>
</InitiateMultipartUploadResult>
```

## UploadPart

初始化一个Multipart Upload之后，可以根据指定的Object名和Upload ID来分块（Part）上传数据。每一个上传的Part都有一个标识它的号码（part number，范围是1~10,000）。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。如果你用同一个part号码，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。除了最后一块Part以外，其他的part最小为100KB；最后一块Part没有大小限制。

### 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
```

### 细节分析

1. 调用该接口上传Part数据前，必须调用Initiate Multipart Upload接口，获取一个OSS服务器颁发的Upload ID。
2. Multipart Upload要求除最后一个Part以外，其他的Part大小都要大于100KB。但是Upload Part接口并不会立即校验上传Part的大小（因为不知道是否为最后一块）；只有当Complete Multipart Upload的时候才会校验。
3. 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。
4. Part号码的范围是1~10000。如果超出这个范围，OSS将返回InvalidArgument的错误码。
5. 若调用Initiate Multipart Upload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Upload Part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，具体见Initiate Multipart Upload接口。
6. 为了保证数据在网络传输过程中不出现错误，用户发送请求时携带Content-MD5，OSS会计算上传数据的MD5与用户上传的MD5值比较，如果不一致返回InvalidDigest错误码。

### 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/ICfXEvPmmSW86bBAfMmUmWjI=

[6291456 bytes data]
```

**返回示例:**

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
ETag: 7265F4D211B56873A381D321F586E4A9
x-oss-request-id: 3e6aba62-1eae-d246-6118-8ff42cd0c21a
Date: Wed, 22 Feb 2012 08:32:21 GMT
```

## UploadPartCopy

## Upload Part Copy

Upload Part Copy通过从一个已存在的Object中拷贝数据来上传一个Part。通过在Upload Part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Upload Part Copy的方式进行拷贝。Upload Part Copy 的源Bucket地址和目标Bucket地址必须是同一个Region。如果想通过单个操作拷贝小于1GB的文件，可以参考Copy Object。

### 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Header

除了通用的请求Header，Upload Part Copy请求中通过下述Header指定拷贝的源Object地址和拷贝的范围。

| 名称                      | 描述                                                                                       |
|-------------------------|------------------------------------------------------------------------------------------|
| x-oss-copy-source       | 复制源地址（必须有可读权限）<br>类型：字符串<br>默认值：无                                                        |
| x-oss-copy-source-range | 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。<br>类型：整型<br>默认值：无 |

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

| 名称                                    | 描述                                                                                           |
|---------------------------------------|----------------------------------------------------------------------------------------------|
| x-oss-copy-source-if-match            | 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）。<br>类型：字符串<br>默认值：无               |
| x-oss-copy-source-if-none-match       | 如果源Object自从用户指定的时间以后就没有被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无               |
| x-oss-copy-source-if-unmodified-since | 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。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-modified-since   | 如果源Object自从用户指定的时间以后被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无                  |

## 响应元素(Response Elements)

| 名称                              | 描述                                                                             |
|---------------------------------|--------------------------------------------------------------------------------|
| x-oss-copy-source-if-match      | 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）。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-none-match | 如果源Object自从用户指定的时间以后就没有被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无 |

|                                       |                                                                                              |
|---------------------------------------|----------------------------------------------------------------------------------------------|
| x-oss-copy-source-if-unmodified-since | 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。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-modified-since   | 如果源Object自从用户指定的时间以后被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无                  |

## 细节分析

- 调用该接口上传Part数据前，必须调用Initiate Multipart Upload接口，获取一个OSS服务器颁发的Upload ID。
- Multipart Upload要求除最后一个Part以外，其他的Part大小都要大于100KB。但是Upload Part接口并不会立即校验上传Part的大小（因为不知道是否为最后一块）；只有当Complete Multipart Upload的时候才会校验。
- 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。
- 若调用Initiate Multipart Upload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Upload Part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，具体见Initiate Multipart Upload接口。
- 该操作不能拷贝通过Append追加上传方式产生的object。
- 如果Bucket的类型为Archive，则不能调用该接口，否则返回400错误，错误码为OperationNotSupported。

## 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/ICfXEvPmmSW86bBAfMmUmWjI=
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-d246-6118-8ff42cd0c21a
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>"5B3C1A2E053D763E1B002CC607C5A0FE" </ETag>
</CopyPartResult>
```

## CompleteMultipartUpload

在将所有数据Part都上传完成后，必须调用Complete Multipart Upload API来完成整个文件的Multipart Upload。在执行该操作时，用户必须提供所有有效的数据Part的列表（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。

### 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            | 描述                                                                                                                                                |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| encoding-type | 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。<br>数据类型：字符串 |

|  |               |
|--|---------------|
|  | 默认值：无,可选值：url |
|--|---------------|

## 请求元素(Request Elements)

| 名称                      | 描述                                                                                     |
|-------------------------|----------------------------------------------------------------------------------------|
| CompleteMultipartUpload | 保存Complete Multipart Upload请求内容的容器。<br>类型：容器<br>子节点：一个或多个Part元素<br>父节点：无               |
| ETag                    | Part成功上传后，OSS返回的ETag值。<br>类型：字符串<br>父节点：Part                                           |
| Part                    | 保存已经上传Part信息的容器。<br>类型：容器<br>子节点：ETag, PartNumber<br>父节点：InitiateMultipartUploadResult |
| PartNumber              | Part数目。<br>类型：整数<br>父节点：Part                                                           |

## 响应元素(Response Elements)

| 名称                            | 描述                                                                                                                                                                                |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bucket                        | Bucket名称。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult                                                                                                                          |
| CompleteMultipartUploadResult | 保存Complete Multipart Upload请求结果的容器。<br>类型：容器<br>子节点：Bucket, Key, ETag, Location<br>父节点：None                                                                                       |
| ETag                          | 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。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult |
| Location                      | 新创建Object的URL。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult                                                                                                                     |
| Key                           | 新创建Object的名字。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult                                                                                                                      |



|              |                                                                              |
|--------------|------------------------------------------------------------------------------|
| EncodingType | 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。<br>类型：字符串<br>父节点：容器 |
|--------------|------------------------------------------------------------------------------|

## 细节分析

1. Complete Multipart Upload时，会确认除最后一块以外所有块的大小都大于100KB，并检查用户提交的Partlist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。
2. OSS处理Complete Multipart Upload请求时，会持续一定的时间。在这段时间内，如果客户端和OSS之间的链接断掉，OSS仍会继续将请求做完。
3. 用户提交的Part List中,Part号码可以是不连续的。例如第一块的Part号码是1；第二块的Part号码是5。
4. OSS处理Complete Multipart Upload请求成功后，该Upload ID就会变成无效。
5. 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。
6. 若调用Initiate Multipart Upload接口时，指定了x-oss-server-side-encryption请求头，则在Complete Multipart Upload的响应头中，会返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
7. 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:

```
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 qn6qrrqx02oawuk53otfjbyc:8VwFhFUWmVecK6jQIHIXMK/zMT0=

<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
  </Part>
  <Part>
    <PartNumber>5</PartNumber>
    <ETag>"8EFDA8BE206636A695359836FE0A0E0A"</ETag>
  </Part>
  <Part>
    <PartNumber>8</PartNumber>
    <ETag>"8C315065167132444177411FDA149B92"</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-4501-4ac71d217d6e
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>B864DB6A936D376F9F8D3ED3BBE540DD-3</ETag>
</CompleteMultipartUploadResult>
```

## AbortMultipartUpload

该接口可以根据用户提供的Upload ID中止其对应的Multipart Upload事件。当一个Multipart Upload事件被中止后，就不能再使用这个Upload ID做任何操作，已经上传的Part数据也会被删除。

### 请求语法

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

### 细节分析

1. 中止一个Multipart Upload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，为了彻底释放OSS上的空间，需要调用几次Abort Multipart Upload接口。
2. 如果输入的Upload Id不存在，OSS会返回404错误，错误码为：NoSuchUpload。

### 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/ICfXEvPmmSW86bBAfMmUmWji=
```

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
```

# ListMultipartUploads

List Multipart Uploads可以罗列出所有执行中的Multipart Upload事件，即已经被初始化的Multipart Upload但是未被Complete或者Abort的Multipart Upload事件。OSS返回的罗列结果中最多会包含1000个Multipart Upload信息。如果想指定OSS返回罗列结果内Multipart Upload信息的数目，可以在请求中添加max-uploads参数。另外，OSS返回罗列结果中的IsTruncated元素标明是否还有其他的Multipart Upload。

## 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(Request Parameters)

| 名称          | 描述                                                                                           |
|-------------|----------------------------------------------------------------------------------------------|
| delimiter   | 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。<br>类型：字符串 |
| max-uploads | 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。<br>类型：字符串                |

|                  |                                                                                                                                                                                                                                                                                      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| key-marker       | 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事件和Object名字等于key-marker参数值的，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。<br>类型：字符串 |
| prefix           | 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。<br>类型：字符串                                                                                                                                                                                                              |
| upload-id-marker | 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事件和Object名字等于key-marker参数值的，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。<br>类型：字符串                                            |
| encoding-type    | 指定对返回的内容进行编码，指定编码的类型。Delimiter、KeyMarker、Prefix、NextKeyMarker和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、NextKeyMarker和Key进行编码。<br>数据类型：字符串<br>默认值：无                                                |

## 响应元素(Response Elements)

| 名称                         | 描述                                                                                                                                                                                                   |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ListMultipartUploadsResult | 保存List Multipart Upload请求结果的容器。<br>类型：容器<br>子节点：Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MasUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload<br>父节点：None |
| Bucket                     | Bucket名称。<br>类型：字符串<br>父节点：ListMultipartUploadsResult                                                                                                                                                |
| EncodingType               | 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、NextKeyMarker和Key这些元素进行编码。                                                                                                |

|                  |                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | 类型：字符串<br>父节点：ListMultipartUploadsResult                                                                                                            |
| KeyMarker        | 列表的起始Object位置。<br>类型：字符串<br>父节点：ListMultipartUploadsResult                                                                                          |
| UploadIdMarker   | 列表的起始UploadID位置。<br>类型：字符串<br>父节点：ListMultipartUploadsResult                                                                                        |
| NextKeyMarker    | 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。<br>类型：字符串<br>父节点：ListMultipartUploadsResult                                              |
| NextUploadMarker | 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。<br>类型：字符串<br>父节点：ListMultipartUploadsResult                                        |
| MaxUploads       | 返回的最大Upload数目。<br>类型：整数<br>父节点：ListMultipartUploadsResult                                                                                           |
| IsTruncated      | 标明是否本次返回的Multipart Upload结果列表被截断。“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。<br>类型：枚举字符串 有效值：false、true<br>默认值：false<br>父节点：ListMultipartUploadsResult |
| Upload           | 保存Multipart Upload事件信息的容器。<br>类型：容器<br>子节点：Key, UploadId, Initiated<br>父节点：ListMultipartUploadsResult                                               |
| Key              | 初始化Multipart Upload事件的Object名字。<br>类型：字符串<br>父节点：Upload                                                                                             |
| UploadId         | Multipart Upload事件的ID。<br>类型：字符串<br>父节点：Upload                                                                                                      |
| Initiated        | Multipart Upload事件初始化的时间。<br>类型：日期<br>父节点：Upload                                                                                                    |

## 细节分析

1. “max-uploads” 参数最大值为1000。
2. 在OSS的返回结果首先按照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。

3. 可以灵活地使用prefix参数对bucket内的object进行分组管理（类似与文件夹的功能）。
4. List Multipart Uploads请求支持5种请求参数：prefix，marker，delimiter，upload-id-marker和max-uploads。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。

## 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+dcivn7kwBMvOY=
```

### 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-ba6f561ce67a
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</NextUploadIdMarker>
  <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>0004B999EF5A239BB9138C6227D69F95</UploadId>
    <Initiated>2012-02-23T04:18:23.000Z</Initiated>
  </Upload>
  <Upload>
    <Key>oss.avi</Key>
    <UploadId>0004B99B8E707874FC2D692FA5D77D3F</UploadId>
    <Initiated>2012-02-23T06:14:27.000Z</Initiated>
  </Upload>
</ListMultipartUploadsResult>
```

# ListParts

List Parts命令可以罗列出指定Upload ID所属的所有已经上传成功Part。

## 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                 | 描述                                                                                                                                                              |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uploadId           | Multipart Upload事件的ID。<br>类型：字符串<br>默认值：无                                                                                                                       |
| max-parts          | 规定在OSS响应中的最大Part数目。<br>类型：整数<br>默认值：1,000                                                                                                                       |
| part-number-marker | 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。<br>类型：整数<br>默认值：无                                                                                                    |
| encoding-type      | 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。<br>数据类型：字符串<br>默认值：无，可选值：url |

## 响应元素(Response Elements)

| 名称              | 描述                                                                                                                             |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------|
| ListPartsResult | 保存List Part请求结果的容器。<br>类型：容器<br>子节点：Bucket, Key, UploadId, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part |

|                      |                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------|
|                      | 父节点：无                                                                                                                    |
| Bucket               | Bucket名称。<br>类型：字符串<br>父节点：ListPartsResult                                                                               |
| EncodingType         | 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。<br>类型：字符串<br>父节点：ListPartsResult                             |
| Key                  | Object名称。<br>类型：字符串<br>父节点：ListPartsResult                                                                               |
| UploadId             | Upload事件ID。<br>类型：字符串<br>父节点：ListPartsResult                                                                             |
| PartNumberMarker     | 本次List结果的Part Number起始位置。<br>类型：整数<br>父节点：ListPartsResult                                                                |
| NextPartNumberMarker | 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。<br>类型：整数<br>父节点：ListPartsResult                 |
| MaxParts             | 返回请求中最大的Part数目。<br>类型：整数<br>父节点：ListPartsResult                                                                          |
| IsTruncated          | 标明是否本次返回的List Part结果列表被截断。<br>“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。<br>类型：枚举字符串 有效值：true、false<br>父节点：ListPartsResult |
| Part                 | 保存Part信息的容器。<br>类型：字符串<br>子节点：PartNumber, LastModified, ETag, Size<br>父节点：ListPartsResult                                |
| PartNumber           | 标示Part的数字。<br>类型：整数<br>父节点：ListPartsResult.Part                                                                          |
| LastModified         | Part上传的时间。<br>类型：日期<br>父节点：ListPartsResult.part                                                                          |
| ETag                 | 已上传Part内容的ETag。<br>类型：字符串<br>父节点：ListPartsResult.Part                                                                    |
| Size                 | 已上传Part大小。<br>类型：整数<br>父节点：ListPartsResult.Part                                                                          |



## 细节分析

1. List Parts支持max-parts和part-number-marker两种请求参数。
2. max-parts参数最大值为1000；默认值也为1000。
3. 在OSS的返回结果按照Part号码升序排列。
4. 由于网络传输可能出错，所以不推荐用List Part出来的结果（Part Number和ETag值）来生成最后Complete Multipart的Part列表。

## 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:4qOnUMc9UQWqkz8wDqD3lIsa9P8=
```

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>"3349DC700140D7F86A078484278075A9"</ETag>
    <Size>6291456</Size>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2012-02-23T07:01:12.000Z</LastModified>
    <ETag>"3349DC700140D7F86A078484278075A9"</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>
```

# 跨域资源共享

## 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提供了CORS支持以方便利用OSS开发更灵活的WEB应用程序。OSS提供接口方便开发者控制跨域访问的各种权限。

## PutBucketcors

Put Bucket cors操作将在指定的bucket上设定一个跨域资源共享(CORS)的规则，如果原规则存在则覆盖原规则。

## 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-flight response</MaxAgeSeconds>
</CORSRule>
<CORSRule>
...
</CORSRule>
...
</CORSConfiguration >

```

## 请求元素 ( Request Elements )

| 名称            | 描述                                                                                                                                                                             | 是否必须 |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| CORSRule      | CORS规则的容器，每个 bucket最多允许10条规则<br>类型：容器<br>父节点：CORSConfiguration                                                                                                                 | 是    |
| AllowedOrigin | 指定允许的跨域请求的来源，允许使用多个元素来指定多个允许的来源。 允许使用最多一个 "*" 通配符。如果指定为 "*" 则表示允许所有的来源的跨域请求。<br>类型：字符串<br>父节点：CORSRule                                                                         | 是    |
| AllowedMethod | 指定允许的跨域请求方法。<br>类型：枚举<br>( GET,PUT,DELETE,POST,HEAD )<br>父节点：CORSRule                                                                                                          | 是    |
| AllowedHeader | 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。允许使用最多一个 "*" 通配符<br>类型：字符串<br>父节点：CORSRule | 否    |
| ExposeHeader  | 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。）不允许使用 "*" 通配符。<br>类型：字符串<br>父节点：CORSRule                                                                                 | 否    |
| MaxAgeSeconds | 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。 一个 CORSRule里面最多允许出现一                                                                                                                 | 否    |

|                   |                                   |   |
|-------------------|-----------------------------------|---|
|                   | 个。<br>类型：整型<br>父节点：CORSRule       |   |
| CORSConfiguration | Bucket的CORS规则容器<br>类型：容器<br>父节点：无 | 是 |

## 细节分析

1. 默认bucket是不开启CORS功能，所有的跨域请求的origin都不被允许。
2. 为了在应用程序中使用CORS功能，比如从一个www.a.com的网址通过浏览器的XMLHttpRequest功能来访问OSS，需要通过本接口手动上传CORS规则来开启。该规则由XML文档来描述。
3. 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文档最多允许16KB大小。
4. 当OSS收到一个跨域请求（或者OPTIONS请求），会读取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。
5. CORS规则匹配成功必须满足三个条件，首先，请求的Origin必须匹配一项AllowedOrigin项，其次，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项AllowedMethod项，最后，OPTIONS请求的Access-Control-Request-Headers头包含的每个header都必须匹配一项AllowedHeader项。
6. 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

添加bucket跨域访问请求规则示例：

```
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:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?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: 50519080C4689A033D00235F
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

# GetBucketcors

Get Bucket cors操作用于获取指定的Bucket目前的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(Response Elements)

| 名称            | 描述                                                                                                     |
|---------------|--------------------------------------------------------------------------------------------------------|
| CORSRule      | CORS规则的容器，每个bucket最多允许10条规则<br>类型：容器<br>父节点：CORSConfiguration                                          |
| AllowedOrigin | 指定允许的跨域请求的来源，允许使用多个元素来指定多个允许的来源。 允许使用最多一个 “*” 通配符。如果指定为 “*” 则表示允许所有的来源的跨域请求。<br>类型：字符串<br>父节点：CORSRule |
| AllowedMethod | 指定允许的跨域请求方法。<br>类型：枚举 ( GET,PUT,DELETE,POST,HEAD )                                                     |

|                   |                                                                                                                                                                              |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   | 父节点：CORSRule                                                                                                                                                                 |
| AllowedHeader     | 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。允许使用最多一个 "*" 通配符<br>类型：字符串<br>父节点：CORSRule |
| ExposeHeader      | 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。不允许使用 "*" 通配符。<br>类型：字符串<br>父节点：CORSRule                                                                                  |
| MaxAgeSeconds     | 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。一个CORSRule里面最多允许出现一个。<br>类型：整型<br>父节点：CORSRule                                                                                      |
| CORSConfiguration | Bucket的CORS规则容器<br>类型：容器<br>父节点：无                                                                                                                                            |

## 细节分析

1. 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能获取CORS规则，否则返回403 Forbidden错误,错误码：AccessDenied。
3. 如果CORS规则不存在，返回404 Not Found错误，错误码NoSuchCORSConfiguration。

## 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+zNhH1AcF51NNHD39zXw=
```

已设置CORS规则的返回示例：

```
HTTP/1.1 200
x-oss-request-id: 50519080C4689A033D00235F
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>
```

## DeleteBucketcors

Delete Bucket cors用于关闭指定Bucket对应的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
```

### 细节分析

1. 如果Bucket不存在，返回404 no content错误，错误码：NoSuchBucket。
2. 只有Bucket的拥有者才能删除Bucket对应的CORS规则。如果试图操作一个不属于你的Bucket，OSS返回403 Forbidden错误，错误码：AccessDenied。

### 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 qn6qrrqx02oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWicOMEkVg=
```

返回示例：

```
HTTP/1.1 204 No Content
x-oss-request-id: 5051845BC4689A033D0022BC
```

```
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

# OptionObject

浏览器在发送跨域请求之前会发送一个preflight请求（OPTIONS）并带上特定的来源域，HTTP方法和header信息等给OSS以决定是否发送真正的请求。OSS可以通过Put Bucket cors接口来开启Bucket的CORS支持，开启CORS功能之后，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，返回403 Forbidden。

## 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Header

| 名称                             | 描述                                               |
|--------------------------------|--------------------------------------------------|
| Origin                         | 请求来源域，用来标示跨域请求。<br>类型：字符串<br>默认值：无               |
| Access-Control-Request-Method  | 表示在实际请求中将会用到的方法。<br>类型：字符串<br>默认值：无              |
| Access-Control-Request-Headers | 表示在实际请求中会用到的除了简单头部之外的headers。<br>类型：字符串<br>默认值：无 |

## 响应Header

| 名称                          | 描述                           |
|-----------------------------|------------------------------|
| Access-Control-Allow-Origin | 请求中包含的Origin，如果不允许的话将不包含该头部。 |



|                               |                                                               |
|-------------------------------|---------------------------------------------------------------|
|                               | 类型：字符串                                                        |
| Access-Control-Allow-Methods  | 允许请求的HTTP方法，如果不允许该请求，则不包含该头部。<br>类型：字符串                       |
| Access-Control-Allow-Headers  | 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。<br>类型：字符串 |
| Access-Control-Expose-Headers | 允许在客户端JavaScript程序中访问的headers的列表。<br>类型：字符串                   |
| Access-Control-Max-Age        | 允许浏览器缓存preflight结果的时间，单位为秒<br>类型：整型                           |

## 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: 5051845BC4689A033D0022BC
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
```

## 关于LiveChannel的操作

# LiveChannel简介

用户可以使用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，用户首先需要创建一个LiveChannel，以获取对应的推流地址，更详细的信息请参考对应的API。

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。

## RTMP推流地址及签名

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。<br>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。<br>参数中不包含SecurityToken、OSSAccessKeyId和Expirel以及Signature。<br>每一个param key只能出现一次。 |

## PutLiveChannelStatus

LiveChannel有两种Status：enabled和disabled，用户可以使用本接口在两种Status之间进行切换。处于disabled状态时，OSS会禁止用户向该LiveChannel进行推流操作；如果有用户正在向该LiveChannel推流，那么推流的客户端会被强制断开（可能会有10s左右的延迟）。

### 请求语法

PUT /ChannelName?live&status=NewStatus HTTP/1.1Date: GMT dateHost: BucketName.oss-cn-hangzhou.aliyuncs.comAuthorization: SignatureValue

### 请求参数

| 名称        | 描述                      | 是否必需 |
|-----------|-------------------------|------|
| NewStatus | 指定LiveChannel的目标Status。 | 是    |

|  |                      |  |
|--|----------------------|--|
|  | 有效值：enabled、disabled |  |
|--|----------------------|--|

## 细节分析

1. 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。
2. 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。

## 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/JoAfRC0ytyQ5pY=
```

### 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
```

# PutLiveChannel

通过RTMP协议上传音视频数据前，必须先调用该接口来创建一个LiveChannel。该接口会返回RTMP推流地址，以及对应的播放地址。用户可以使用返回的地址进行推流、播放。另外，用户可以根据该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>
```

## 请求元素(Request Elements)

| 名称                       | 描述                                                                                                  | 是否必需             |
|--------------------------|-----------------------------------------------------------------------------------------------------|------------------|
| LiveChannelConfiguration | 保存LiveChannel配置的容器。<br>类型：容器<br>子节点：Description、Status、Target<br>父节点：无                              | 是                |
| Description              | LiveChannel的描述信息，最长128字节。<br>类型：字符串<br>子节点：无<br>父节点：LiveChannelConfiguration                        | 否                |
| Status                   | 指定LiveChannel的状态。<br>类型：枚举字符串<br>子节点：无<br>父节点：LiveChannelConfiguration<br>有效值：enabled、disabled      | 否<br>默认值：enabled |
| Target                   | 保存转储配置的容器。<br>类型：容器<br>子节点：Type、FragDuration、FragCount、PlaylistName<br>父节点：LiveChannelConfiguration | 是                |
| Type                     | 指定转储的类型。<br>类型：枚举字符串                                                                                | 是                |

|              |                                                                                                               |                        |
|--------------|---------------------------------------------------------------------------------------------------------------|------------------------|
|              | 子节点：无<br>父节点：Target<br>有效值：HLS                                                                                |                        |
| FragDuration | 当Type为HLS时，指定每个ts文件的时长（单位：秒），取值范围为[1, 100]的整数。<br>类型：字符串<br>子节点：无<br>父节点：Target                               | 否<br>默认值：5             |
| FragCount    | 当Type为HLS时，指定m3u8文件中包含ts文件的个数，取值范围为[1, 100]的整数。<br>类型：字符串<br>子节点：无<br>父节点：Target                              | 否<br>默认值：3             |
| PlaylistName | 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]。<br>类型：字符串<br>子节点：无<br>父节点：Target                        | 否<br>默认值：playlist.m3u8 |
| Snapshot     | 保存高频截图操作Snapshot 选项的容器。<br>类型：容器<br>子节点：<br>：RoleName，DestBucket，NotifyTopic，Interval，PornRec<br>父节点：Snapshot | 否                      |
| RoleName     | 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。<br>类型：字符串<br>子节点：无<br>父节点：Snapshot                        | 否                      |
| DestBucket   | 保存高频截图目标Bucket，要求与当前Bucket是同一个Owner。<br>类型：字符串<br>子节点：无<br>父节点：Snapshot                                       | 否                      |
| NotifyTopic  | 用于通知用户高频截图操作结果的MNS的Topic。<br>类型：字符串<br>子节点：无<br>父节点：Snapshot                                                  | 否                      |
| Interval     | 高频截图的间隔长度，单位为s，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。                                                                 | 否                      |

|  |                                |  |
|--|--------------------------------|--|
|  | 类型：数字<br>子节点：无<br>父节点：Snapshot |  |
|--|--------------------------------|--|

## 细节分析

1. ChannelName必须符合ObjectName的命名规范，另外，ChannelName不能包含“ / ”。
2. 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。
3. 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。
4. 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。

## 响应元素

| 名称                      | 描述                                                                        |
|-------------------------|---------------------------------------------------------------------------|
| CreateLiveChannelResult | 保存CreateLiveChannel请求结果的容器。<br>类型：容器<br>子节点：PublishUrls,PlayUrls<br>父节点：无 |
| PublishUrls             | 保存推流地址的容器。<br>类型：容器<br>子节点：Url<br>父节点：CreateLiveChannelResult             |
| Url                     | 推流地址。<br>类型：字符串<br>子节点：无<br>父节点：PublishUrls                               |
| PlayUrls                | 保存推流地址的容器。<br>类型：容器<br>子节点：Url<br>父节点：CreateLiveChannelResult             |
| Url                     | 播放地址。<br>类型：字符串<br>子节点：无<br>父节点：PlayUrls                                  |

## 细节分析

1. 推流地址是未加签名的url，如果bucket acl非public-read-write，那么需要首先进行签名才能进行访问。

2. 播放地址是未加签名的url，如果bucket acl为private，那么需要首先进行签名才能进行访问。

## 实例

### 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+agA=
```

```
<?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>
```



# PostVodPlaylist

本接口用来为指定LiveChannel推流生成的，指定时间段内的ts文件生成一个点播用的播放列表（m3u8文件）。

## 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           | 描述                                | 是否必需 |
|--------------|-----------------------------------|------|
| ChannelName  | 指定LiveChannel名称，该LiveChannel必须存在。 | 是    |
| PlaylistName | 指定生成的点播播放列表的名称，必须以“.m3u8”结尾。      | 是    |
| StartTime    | 指定查询ts文件的起始时间，格式为Unix timestamp。  | 是    |
| EndTime      | 指定查询ts文件的终止时间，格式为Unix timestamp。  | 是    |

## 细节分析

1. EndTime必须大于StartTime，且时间跨度不能大于1天。
2. OSS会查询指定时间范围内的所有该LiveChannel推流生成的ts文件，并将其拼装为一个播放列表。

## 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 YjHKOKWDWINLKXv:ABiIgvnLtCHK+7fMHLeRIOUnzv0=

返回示例

HTTP/1.1 200  
content-length: 0  
server: AliyunOSS  
connection: close  
etag: "9C6104DD9CF1A0C4D0CFD21F43905D59"  
x-oss-request-id: 57BE9A96B92475920B002359  
date: Thu, 25 Aug 2016 07:13:26 GMT

GetLiveChannelStat

本接口用来获取指定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              | 描述                                                                                                |
|-----------------|---------------------------------------------------------------------------------------------------|
| LiveChannelStat | 保存GetLiveChannelStat返回结果的容器。<br>类型：容器<br>子节点<br>： Status , ConnectedTime , Video , Audio<br>父节点：无 |
| Status          | LiveChannel当前的推流状态描述。<br>类型：枚举字符串<br>子节点：无<br>父节点：LiveChannelStat<br>有效值： Disabled , Live , Idle  |
| ConnectedTime   | 当Status为Live时，表示当前客户端开始推流的时间，使用ISO8601格式表示。<br>类型：字符串<br>子节点：无                                    |

|            |                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------|
|            | 父节点：LiveChannelStat                                                                                         |
| RemoteAddr | 当Status为Live时，表示当前推流客户端的ip地址。<br>类型：字符串<br>子节点：无<br>父节点：LiveChannelStat                                     |
| Video      | 当Status为Live时，保存视频流信息的容器。<br>类型：容器<br>子节点：Width, Height, FrameRate, Bandwidth, Codec<br>父节点：LiveChannelStat |
| Width      | 表示当前视频流的画面宽度（单位：像素）。<br>类型：字符串<br>子节点：无<br>父节点：Video                                                        |
| Height     | 表示当前视频流的画面高度（单位：像素）。<br>类型：字符串<br>子节点：无<br>父节点：Video                                                        |
| FrameRate  | 表示当前视频流的帧率。<br>类型：字符串<br>子节点：无<br>父节点：Video                                                                 |
| Bandwidth  | 表示当前视频流的码率（单位：B/s）。<br>类型：字符串<br>子节点：无<br>父节点：Video                                                         |
| Codec      | 表示当前视频流的编码格式。<br>类型：枚举字符串<br>子节点：无<br>父节点：Video                                                             |
| Audio      | 当Status为Live时，保存音频流信息的容器。<br>类型：容器<br>子节点：SampleRate, Bandwidth, Codec<br>父节点：LiveChannelStat               |
| SampleRate | 表示当前音频流的采样率。<br>类型：字符串<br>子节点：无<br>父节点：Audio                                                                |
| Bandwidth  | 表示当前音频流的码率（单位：B/s）。<br>类型：字符串<br>子节点：无<br>父节点：Audio                                                         |
| Codec      | 表示当前音频流的编码格式。<br>类型：枚举字符串<br>子节点：无<br>父节点：Audio                                                             |

## 细节分析

1. Video , Audio容器只有在Status为Live时才会返回，但Status为Live时不一定会返回  
Video , Audio容器，例如，客户端已经连接到LiveChannel，但尚未发送音视频数据时不会返回。
2. 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。

## 示例

### 请求示例I

```
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:fOzwkAgVTVSO1VKLPiInQ0JYyOA=
```

### 返回示例I

```
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II

```
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:WeC5joEaRzfSSS8xK0tlo7WTK1I=
```

### 返回示例II

```
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>
```

# GetLiveChannelInfo

本接口用来获取指定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                       | 描述                                                                              |
|--------------------------|---------------------------------------------------------------------------------|
| LiveChannelConfiguration | 保存GetLiveChannelInfo返回结果的容器。<br>类型：容器<br>子节点：Description、Status、Target<br>父节点：无 |
| Description              | LiveChannel的描述信息。<br>类型：字符串                                                     |

|              |                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------|
|              | 子节点：无<br>父节点：LiveChannelConfiguration                                                                          |
| Status       | LiveChannel的状态信息。<br>类型：枚举字符串<br>子节点：无<br>父节点：LiveChannelConfiguration<br>有效值：enabled、disabled                 |
| Target       | 保存LiveChannel转储配置的容器。<br>类型：容器<br>子节点：Type、FragDuration、FragCount、PlaylistName<br>父节点：LiveChannelConfiguration |
| Type         | 当Type为HLS时，指定推流时转储文件类型。<br>类型：枚举字符串<br>子节点：无<br>父节点：Target<br>有效值：HLS                                          |
| FragDuration | 当Type为HLS时，指定每个ts文件的时长（单位：秒）。<br>类型：字符串<br>子节点：无<br>父节点: Target                                                |
| FragCount    | 当Type为HLS时，指定m3u8文件中包含ts文件的个数。<br>类型：字符串<br>子节点：无<br>父节点：Target                                                |
| PlaylistName | 当Type为HLS时，指定生成的m3u8文件的名称。<br>类型：字符串<br>子节点：无<br>父节点：Target                                                    |

## 细节分析

Target的子节点FragDuration，FragCount，PlaylistName只有当Type取值为HLS时才会返回。

## 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 YJhKOKWDWINLKXv:D6bDCRXKht58hin1BL83wxyGvl0=
```

## 返回示例

```
HTTP/1.1 200
content-length: 475
server: AliyunOSS
connection: close
x-oss-request-id: 57BE87A8B92475920B002098
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>
```

# GetLiveChannelHistory

本接口用来获取指定LiveChannel的推流记录。

## 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                 | 描述                                                                  |
|--------------------|---------------------------------------------------------------------|
| LiveChannelHistory | 保存GetLiveChannelHistory返回结果的容器。<br>类型：容器<br>子节点：LiveRecord<br>父节点：无 |
| LiveRecord         | 保存一次推流记录信息的容器。<br>类型：容器                                             |

|            |                                                            |
|------------|------------------------------------------------------------|
|            | 子节点：StartTime，EndTime，RemoteAddr<br>父节点：LiveChannelHistory |
| StartTime  | 推流开始时间，使用ISO8601格式表示。<br>类型：字符串<br>子节点：无<br>父节点：LiveRecord |
| EndTime    | 推流结束时间，使用ISO8601格式表示。<br>类型：字符串<br>子节点：无<br>父节点：LiveRecord |
| RemoteAddr | 推流客户端的ip地址。<br>类型：字符串<br>子节点：无<br>父节点：LiveRecord           |

## 细节分析

目前最多会返回指定LiveChannel最近的10次推流记录。

## 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:pqgDBP8JTXAytBoXpvNoZfo68k=
```

### 返回实例

```
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>

```

## ListLiveChannel

### 请求语法

```

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

```

### 请求参数

| 名称       | 描述                                                             | 是否必需         |
|----------|----------------------------------------------------------------|--------------|
| marker   | 设定结果从marker之后按字母排序的第一个开始返回。                                    | 否            |
| max-keys | 限定此次返回LiveChannel的最大数，如果不设定，默认为100，max-keys取值不能大于1000。         | 否<br>默认值：100 |
| prefix   | 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。 | 否            |

### 响应元素

| 名称 | 描述 |
|----|----|
|----|----|

|                       |                                                                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ListLiveChannelResult | 保存ListLiveChannel请求结果的容器。<br>类型：容器<br>字节点<br>：Prefix, Marker, MaxKeys, IsTruncated, NextMarker, LiveChannel<br>父节点：无                    |
| Prefix                | 本次查询结果的开始前缀。<br>类型：字符串<br>子节点：无<br>父节点：ListLiveChannelResult                                                                            |
| Marker                | 本次ListLiveChannel的起点。<br>类型：字符串<br>子节点：无<br>父节点：ListLiveChannelResult                                                                   |
| MaxKeys               | 响应请求内返回结果的最大数目。<br>类型：字符串<br>子节点：无<br>父节点：ListLiveChannelResult                                                                         |
| IsTruncated           | 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。<br>类型：字符串<br>子节点：无<br>父节点：ListLiveChannelResult                                 |
| NextMarker            | 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。<br>类型：字符串<br>子节点：无<br>父节点：ListLiveChannelResult                                    |
| LiveChannel           | 保存返回每个LiveChannel信息的容器。<br>类型：容器<br>子节点<br>：Name, Description, Status, LastModified, PublishUrls, PlayUrls<br>父节点：ListLiveChannelResult |
| Name                  | LiveChannel的名称。<br>类型：字符串<br>子节点：无<br>父节点：LiveChannel                                                                                   |
| Description           | LiveChannel的描述。<br>类型：字符串<br>子节点：无<br>父节点：LiveChannel                                                                                   |
| Status                | LiveChannel的状态。<br>类型：枚举字符串<br>子节点：无<br>父节点：LiveChannel<br>有效值：disabled, enabled                                                        |
| LastModified          | LiveChannel配置的最后修改时间，使用ISO8601格式表示。                                                                                                     |

|             |                                                                 |
|-------------|-----------------------------------------------------------------|
|             | 类型：字符串<br>子节点：无<br>父节点：LiveChannel                              |
| PublishUrls | 保存LiveChannel对应的推流地址的容器。<br>类型：容器<br>子节点：Url<br>父节点：LiveChannel |
| Url         | LiveChannel对应的推流地址。<br>类型：字符串<br>子节点：无<br>父节点：PublishUrls       |
| PlayUrls    | 保存LiveChannel对应的播放地址的容器。<br>类型：容器<br>子节点：Url<br>父节点：LiveChannel |
| Url         | LiveChannel对应的播放地址。<br>类型：字符串<br>子节点：无<br>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/Xsgpz6uRuqcbmUJsIHw=
```

### 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>
```

## DeleteLiveChannel

本接口用来删除指定的LiveChannel。

### 请求语法

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

### 细节分析

1. 当有客户端正在向LiveChannel推流时，删除请求会失败。
2. 注意，本接口只会删除LiveChannel本身，不会删除推流生成的文件。

### 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 YJhKOKWDWINLKXv:ZbfvQ3XwmYEE8O9CX8kwVQYNbzQ=
```

#### 返回示例

```
HTTP/1.1 204
content-length: 0
```

```
server: AliyunOSS  
connection: close  
x-oss-request-id: 57BE9F0AB92475920B0023E0  
date: Thu, 25 Aug 2016 07:32:26 GMT
```