阿里云
对象存储 OSS

API 参考

文档版本：20190111

法律声明

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

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

通用约定

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

目录

1 简介

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

使用限制

您使用的OSS资源和相关功能，都有一定的限制，具体请参见*OSS*使用限制。

使用说明

OSS API参考主要介绍接口的请求语法、相关参数含义以及请求和返回示例。如果要快速进行二次开发，建议您使用SDK开发包。关于SDK的安装和使用，请参见*OSS SDK*参考。

OSS定价

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

资源术语

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

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

2 API概览

OSS提供的API接口如下：

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

关于**Object**的操作

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

关于**Multipart Upload**的操作

API	描述
InitiateMultipartUpload	初始化MultipartUpload事件
UploadPart	分块上传文件
UploadPartCopy	分块复制上传文件
CompleteMultipartUpload	完成整个文件的MultipartUpload上传
AbortMultipartUpload	取消MultipartUpload事件
ListMultipartUploads	罗列出所有执行中的MultipartUpload事件

API	描述
ListParts	罗列出指定UploadID所属的所有已经上传成功Part

跨域资源共享(CORS)

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

关于Live Channel的操作

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

3 公共HTTP头定义

公共请求头（Common **Request** Headers）

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

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

公共响应头（Common **Response** Headers）

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

名称	类型	描述
Content-Length	字符串	*RFC2616*中定义的HTTP请求内容长度。 默认值：无 使用场景：需要向OSS提交数据的请求
Connection	枚举	标明客户端和OSS服务器之间的链接状态。 有效值：open、close 默认值：无
Date	字符串	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT 默认值：无
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。 默认值：无
Server	字符串	生成Response的服务器。 默认值：AliyunOSS
x-oss-request-id	字符串	x-oss-request-id是由Aliyun OSS创建，并唯一标识这个response的UUID。如果在使用OSS服务时遇到问题，可以凭借该字段联系OSS工作人员，快速定位问题。 默认值：无

4 访问控制

4.1 用户签名验证

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。

4.2 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。详情请参见*RFC2616 Content-MD5*。
- Content-Type 表示请求内容的类型，如”application/octet-stream”，也可以为空。
- Date 表示此次操作的时间，且必须为GMT格式，如”Sun, 22 Nov 2015 08:16:38 GMT”。
- CanonicalizedOSSHeaders 表示以 x-oss- 为前缀的HTTP Header的字典序排列。
- CanonicalizedResource 表示用户想要访问的OSS资源。

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

构建CanonicalizedOSSHeaders的方法

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

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

> 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`（如果没有ObjectName则CanonicalizedResource为”/BucketName/“，如果同时也没有BucketName则为“/”）
3. 如果请求的资源包括子资源(SubResource)，那么将所有的子资源按照字典序，从小到大排列并以 `&` 为分隔符生成子资源字符串。在CanonicalizedResource字符串尾添加 `?` 和子资源字符串。此时的CanonicalizedResource如：`/BucketName/ObjectName?acl&uploadId=UploadId`
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，如：`/BucketName/ObjectName?acl&response-content-type=ContentType&uploadId=UploadId`。

注意:

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

计算签名头规则

- 签名的字符串必须为 `UTF-8` 格式。含有中文字符的签名字符串必须先进行 `UTF-8` 编码，再与 `AccessKeySecret`计算最终签名。
- 签名的方法用*RFC 2104*中定义的HMAC-SHA1方法，其中Key为 AccessKeySecret` 。
- `Content-Type` 和 `Content-MD5` 在请求中不是必须的，但是如果请求需要签名验证，空值的话以换行符 `\n` 代替。

- 在所有非HTTP标准定义的header中，只有以 `x-oss-` 开头的header，需要加入签名字符串；其他非HTTP标准header将被OSS忽略（如上例中的x-oss-magic是需要加入签名字符串的）。
- 以 `x-oss-` 开头的header在签名验证前需要符合以下规范：
  - header的名字需要变成小写。
  - header按字典序自小到大排序。
  - 分割header name和value的冒号前后不能有空格。
  - 每个Header之后都有一个换行符“\n”，如果没有Header，CanonicalizedOSSHeaders就设置为空。

签名示例

假如AccessKeyId是”44CF9590006BF252F707”，AccessKeySecret是”OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV”

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

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("OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
             "PUT\nODBGOERFMDMzQTczRUY3NUE3NzA5QzdFNUYzMDQxNEM=\ntext
/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-magic:abracadabra\nx-oss-
meta-author:foo@bar.com\n/oss-example/nelson", sha)
Signature = base64.b64encode(h.digest())
print("Signature: %s" % Signature)
```

签名(Signature)计算结果应该为 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
```

细节分析如下：

- 如果传入的AccessKeyId不存在或inactive，返回403 Forbidden。错误码：InvalidAccessKeyId。
- 若用户请求头中Authorization值的格式不对，返回400 Bad Request。错误码：InvalidArgument。
- OSS所有的请求都必须使用HTTP 1.1协议规定的GMT时间格式。其中，日期的格式为：

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

  上述日期格式中，“天”所占位数都是“2 DIGIT”。因此，“Jun 2”、“2 Jun 1982”和“2-Jun-82”都是非法日期格式。
- 如果签名验证的时候，头中没有传入Date或者格式不正确，返回403 Forbidden错误。错误码：AccessDenied。
- 传入请求的时间必须在OSS服务器当前时间之后的15分钟以内，否则返回403 Forbidden。错误码：RequestTimeTooSkewed。
- 如果AccessKeyId是active的，但OSS判断用户的请求发生签名错误，则返回403 Forbidden，并在返回给用户的response中告诉用户正确的用于验证加密的签名字符串。用户可以根据OSS的response来检查自己的签名字符串是否正确。

  返回示例：

  ```
  <?xml version="1.0" ?>
  <Error>
   <Code>
       SignatureDoesNotMatch
   </Code>
   <Message>
       The request signature we calculated does not match the
  signature you provided. Check your key and signing method.
   </Message>
   <StringToSignBytes>
       47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31
  31 20 30 37 3a 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72 65 61 6c 74
   65 73 74 3f 61 63 6c
   </StringToSignBytes>
   <RequestId>
       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>
```

SDK 签名实现

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

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

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

常见问题

Content-MD5的计算方法

```
Content-MD5的计算
以消息内容为"123456789"来说，计算这个字符串的Content-MD5
正确的计算方式：
标准中定义的算法简单点说就是：
1. 先计算MD5加密的二进制数组（128位）。
2. 再对这个二进制进行base64编码（而不是对32位字符串编码）。
以Python为例子：
```

```
正确计算的代码为：
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
需要注意
正确的是：hash.digest()，计算出进制数组（128位）
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
常见错误是直接对计算出的32位字符串编码进行base64编码。
例如，错误的是：hash.hexdigest()，计算得到可见的32位字符串编码
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
错误的MD5值进行base64编码后的结果：
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

4.3 在URL中包含签名

除了使用Authorization 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?OSSAccessK
eyId=nz2pc56s936**9l&Expires=1141889120&Signature=vjbyPxybdZaNmGa%
2ByT272YEAiv4%3D
```

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

- Expires 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*Wikipedia*），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：

- 通过URL包含签名时，之前的Date参数换成Expires参数。
- 不支持同时在URL和Head中包含签名。
- 如果传入的Signature，Expires，OSSAccessKeyId出现不止一次，以第一次为准。
- 请求先验证请求时间是否晚于Expires时间，然后再验证签名。
- 将签名字符串放到url时，注意要对url进行urlencode

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

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

示例代码

URL中添加签名的python示例代码：

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

OSS SDK中提供了提供URL签名的方法，使用方法请参看SDK参考中的授权访问文档。

OSS SDK的URL签名实现，请参看下表：

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

细节分析

- 使用在URL中签名的方式，会将你授权的数据在过期时间以内曝露在互联网上，请预先评估使用风险。
- PUT和GET请求都支持在URL中签名。
- 在URL中添加签名时，Signature，Expires，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。
- 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。
- 如果Expires时间格式错误，返回403 Forbidden。错误码：AccessDenied。
- 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。
- 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）。

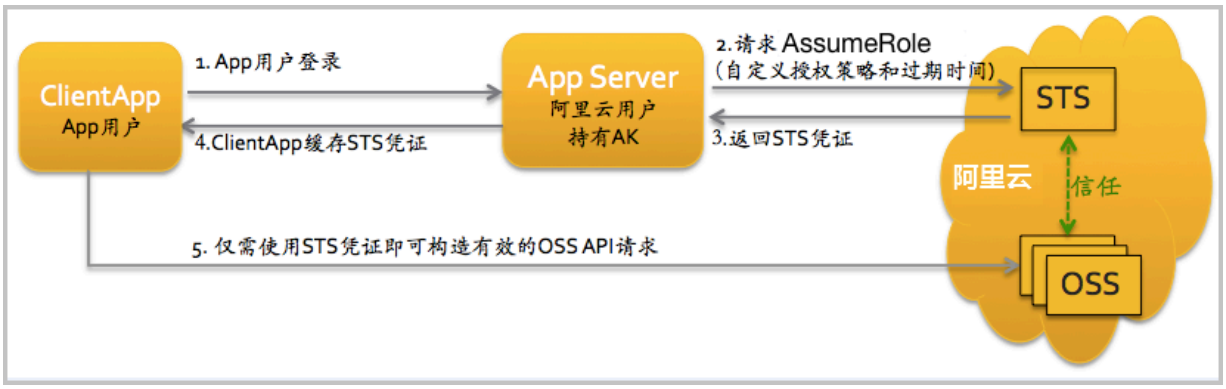
4.4 临时授权访问

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：

1. 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。

2. 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。

3. 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。

4. 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。

5. 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。

注意:

您可以调用STS服务接口*AssumeRole*来获取有效访问凭证，也可以直接使用*STS SDK*来调用该方法。

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。

4.5 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	• 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。 • 所有这些操作产生的费用由该Bucket的Owner承担。
public-read	公共读，私有写	• 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。 • 任何人（包括匿名访问）可以对Object进行读操作。

权限值	中文名称	权限对访问者的限制
private	私有读写	• 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。 • 其他人在未经授权的情况下无法访问该Bucket内的Object。

注意:

- 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。

5 关于Bucket的操作

5.1 PutBucket

PutBucket接口用于创建 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 Region 确定后，该 Bucket 中的所有 Object 将一直存放在对应 Region。详情请参见访问域名和数据中心。

请求语法

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

请求 Header

表 5-1: 请求 Header

	类型	描述
x-oss-acl	字符串	指定 Bucket 访问权限。 有效值：public-read-write、public-read、private

请求元素

表 5-2: 请求元素

	类型	描述
StorageClass	字符串	指定 Bucket 存储类型 有效值：Standard、IA、Archive

细节分析

- 如果创建的 Bucket 没有指定访问权限，则默认使用 private 权限。

- 如果创建的 Bucket 不符合命名规范，则返回 400 Bad Request 消息。错误码：InvalidBucketName。
- 如果用户发起 Put Bucket 请求时没有传入用户验证信息，则返回 403 Forbidden 消息。错误码：AccessDenied。
- 同一用户在同一 Region 内最多可创建 30 个Bucket。如果超过 30 个，则返回 400 Bad Request 消息。错误码：TooManyBuckets。
- 创建 Bucket 时，可以指定 Bucket 的数据容灾类型（DataRedundancyType），有效值为LRS（本地容灾类型，默认值）、ZRS（同城容灾类型）。

示例

请求示例：

```
PUT / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2017 03:15:40 GMT
x-oss-acl: private
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:77Dvh5wQgIjWjwO/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
```

5.2 PutBucketACL

PutBucketACL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

- 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。
- 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**。
- 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:KU5h8YMUC78M30dXqf3J
xrTZHiA=
```

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>5***9</RequestId>
  <HostId>***-test.example.com</HostId>
  <ArgumentName>x-oss-acl</ArgumentName>
  <ArgumentValue>error-acl</ArgumentValue>
```

```
</Error>
```

5.3 PutBucketLogging

PutBucketLogging接口用于为 Bucket 开启访问日志记录功能。

访问日志记录功能开启后，OSS 将自动记录访问这个 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。

注意:

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每一条访问信息。

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)

表 **5-3:** 请求元素

名称	类型	是否必需	描述
BucketLoggingStatus	容器	是	访问日志状态信息的容器。 子元素：LoggingEnabled 父元素：无
LoggingEnabled	容器	否	访问日志信息的容器。这个元素在开启时需要，关闭时不需要。 子元素：TargetBucket, TargetPrefix 父元素：BucketLoggingStatus
TargetBucket	字符	在开启访问日志的时候必需	指定存储访问日志的 Bucket。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

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

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。

一个实际用于存储 OSS 访问日志的 Object 名称示例 如下：

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

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

LOG文件格式

表 5-4: 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-URL	GET /aliyun-logo.png HTTP/1.1	用户请求的 URL （包括 query-string）
HTTP Status	200	OSS 返回的 HTTP 状态码
SentBytes	5576	用户从 OSS 下载的流量

名称	例子	含义
RequestTime (ms)	71	完成本次请求的时间（毫秒）
Referer	`http://www.aliyun.com/product/oss`	请求的 HTTP Referer
User-Agent	curl/7.15.5	HTTP 的 User-Agent header
HostName	oss-example.regionid.example.com	请求访问域名
Request ID	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 名称
Key	/aliyun-logo.png	用户请求的 Key
ObjectSize	5576	Object 大小
Server Cost Time (ms)	17	OSS 服务器处理本次请求所花的时间（毫秒）
Error Code	NoSuchBucket	OSS 返回的错误码
Request Length	302	用户请求的长度（Byte）
UserID	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 必须属于同一用户。如果源 Bucket 不存在，OSS返回错误码：NoSuchBucket。
- 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。

- 源 Bucket 和目标 Bucket 可以是同一个Bucket，也可以是不同的 Bucket；用户也可以将多个源 Bucket 的 LOG 都保存在同一个目标 Bucket 内（建议指定不同的 TargetPrefix）。
- 源 Bucket 被删除时，对应的 Logging 规则也将被删除。
- 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 名称前缀，可以为空。
- 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。
- 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。
- - PUT Bucket Logging
    - 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 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。
    - 如果 PUT Bucket Logging 请求中的XML不合法，返回错误码：MalformedXML。
  - OSS LOG
    - 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.regionid.example.com/aliyun-logo.png
      ```

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

OSS 处理上面两个请求，结果是一样的。但是在访问 LOG 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。

示例

开启 Bucket 访问日志的请求示例：

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

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 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:KU5h8YMUC78M30dXqf3J
xrTZHiA=
<?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
```

5.4 PutBucketWebsite

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

website

website接口主要有两个功能：

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

> **注意:**
> 镜像回源支持公共云、金融云。

website字段样例：

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

```
            <Set>
              <Key>myheader-key5</Key>
              <Value>myheader-value5</Value>
            </Set>
          </MirrorHeaders>
        </Redirect>
      </RoutingRule>
      <RoutingRule>
        <RuleNumber>2</RuleNumber>
        <Condition>
          <KeyPrefixEquals>abc/</KeyPrefixEquals>
          <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
          <IncludeHeader>
            <Key>host</Key>
            <Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
          </IncludeHeader>
        </Condition>
        <Redirect>
          <RedirectType>AliCDN</RedirectType>
          <Protocol>http</Protocol>
          <HostName>www.test.com</HostName>
          <PassQueryString>false</PassQueryString>
          <ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
          <HttpRedirectCode>301</HttpRedirectCode>
        </Redirect>
      </RoutingRule>
    </RoutingRules>
</WebsiteConfiguration>
```

请求语法

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

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

请求元素（Request Elements）

名称	类型	描述	是否必须
WebsiteConfiguration	容器	根节点 父节点：无	是

IndexDocument	容器	默认主页的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。
Suffix	字符串	默认主页 如果设置，则访问以正斜线（/）结尾的object时都会返回此默认主页。 父节点：IndexDocument	有条件，父节点IndexDocument指定时必须指定。
ErrorDocument	容器	404页面的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。
Key	容器	404页面 如果指定，则访问的object不存在时则返回此404页面。 父节点：ErrorDocument	有条件，父节点ErrorDocument指定时必须指定。
RoutingRules	容器	RoutingRule的容器 父节点：WebsiteConfiguration	有条件，IndexDocument、ErrorDocument、RoutingRules三个容器至少指定一个。
RoutingRule	容器	指定跳转规则或者镜像回源规则，最多指定5个RoutingRule。 父节点：RoutingRules	否
RuleNumber	正整数	匹配和执行RoutingRule的序号，匹配时会按照此序号按顺序匹配规则。如果匹配成功，则执行此规则，后续的规则不再执行。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
Condition	容器	匹配的条件 如果指定的项都满足，则执行此规则。此容器下的各个节点关系是与，即需要所有条件都满足才算匹配。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
KeyPrefixEquals	字符串	只有匹配此前缀的object才能匹配此规则。 父节点：Condition	否

HttpErrorCodeReturnedEquals	HTTP状态码	访问指定object时返回此status才能匹配此规则。当跳转规则是镜像回源类型时，此字段必须为404。 父节点：Condition	否
IncludeHeader	容器	只有请求中包含了指定header，且值为指定值时，才能匹配此规则。此容器可以最多重复5个。 父节点：Condition	否
Key	字符串	只有请求中包含了此header，且值为Equals指定值时，才能匹配此规则。 父节点：IncludeHeader	有条件，父节点IncludeHeader指定时必须指定。
Equals	字符串	只有请求中包含了Key指定的header，且值为指定的值时，才能匹配此规则 父节点：IncludeHeader	有条件，父节点IncludeHeader指定时必须指定。
Redirect	容器	指定匹配此规则后执行的动作。 父节点：RoutingRule	有条件，父节点RoutingRule指定时必须指定。
RedirectType	字符串	指定跳转的类型，取值必须为以下几项。 • Mirror（镜像回源） • External（外部跳转，即OSS会返回一个3xx请求，指定跳转到另外一个地址。） • Internal（内部跳转，即OSS会根据此规则将访问的object1转换成另外一个object2，相当于用户访问的是object2。） • AliCDN（阿里云CDN跳转，主要用于阿里云的CDN。与External不同的是，OSS会额外添加一个header。阿里云CDN识别到此header后会主动跳转到指定的地址，返回给用户获取到的数据，而不是将3xx跳转请求返回给客户。） 父节点：Redirect	有条件，父节点Redirect指定时必须指定

PassQueryString	bool型	执行跳转或者镜像回源时，是否要携带发起请求的请求参数，取值true或者false。 用户请求OSS时携带了请求参数”?a=b&c=d”，并且此项设置为true，那么如果规则是302跳转，则在跳转的Location头中会添加此请求参数。如”Location:www.test.com?a=b&c=d”，跳转类型是镜像回源，则在发起的回源请求中也会携带此请求参数。 默认值：false 父节点：Redirect	否
MirrorURL	字符串	只有在RedirectType为Mirror时有意义，表示镜像回源的源站地址。 如以http://或者https://开头，必须以正斜线（/）结尾，OSS会在此字符串的基础上加上object构成回源的url。比如指定为`http://www.test.com/`，访问的obejct是”myobject”，则回源的url为`http://www.test.com/myobject`，如果指定为`http://www.test.com/dir1/`，那么回源的url为`http://www.test.com/dir1/myobject`。 父节点：Redirect	有条件，如果RedirectType为Mirror则必须指定
MirrorPassQueryString	bool型	与PassQueryString作用相同，优先级比PassQueryString高，但只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorFollowRedirect	bool型	如果镜像回源获取的结果是3xx，是否要继续跳转到指定的Location获取数据。 比如发起镜像回源请求时，如果源站返回了302，并且指定了Location。如果此项为true，则oss会继续请求Location指定的地址（最多跳转10此次，如果超过10次，则返回镜像回源失败）。如果为false，那么OSS就会返回302，并将Location透传出去。只有在RedirectType为Mirror时生效。 默认值：true 父节点：Redirect	否

MirrorCheckMd5	bool型	是否要检查回源body的md5。 当此项为true且源站返回的response中含有Content-Md5头时，OSS检查拉取的数据md5是否与此header匹配，如果不匹配，则不保存在oss上。只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorHeaders	容器	用于指定镜像回源时携带的header。只有在RedirectType为Mirror时生效。 父节点：Redirect	否
PassAll	bool型	是否透传请求中所有的header（除了保留的几个header以及以oss-/x-oss-/x-drs-开头的header）到源站。只有在RedirectType为Mirror时生效。 默认值：false 父节点：MirrorHeaders	否
Pass	字符串	透传指定的header到源站，最多10个，每个header长度最多1024个字节，字符集为0-9、A-Z、a-z以及横杠。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Remove	字符串	禁止透传指定的header到源站，这个字段可以重复，最多10个，一般与PassAll一起使用。每个header长度最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Set	容器	设置一个header传到源站，不管请求中是否携带这些指定的header，回源时都会设置这些header。该容器可以重复，最多10组。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Key	字符串	设置header的key，最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定

Value	字符串	设置header的value，最多1024个字节，不能出现”\r\n”。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定。
Protocol	字符串	跳转时的协议，只能取值为http或者https。比如访问的文件为test，设定跳转到`www.test.com`，而且Protocol字段为https，那么Location头为`https://www.test.com/test`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType不为External或者AliCDN时需要指定。
HostName	字符串	跳转时的域名，需要符合域名规范。比如访问的object为test，Protocol为https，Hostname指定为`www.test.com`，那么Location头为`https://www.test.com/test`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType不为External或者AliCDN时需要指定
HttpRedirectCode	HTTP状态码	跳转时返回的状态码，取值为301、302或307。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType不为External或者AliCDN时需要指定。
ReplaceKeyPrefixWith	字符串	Redirect的时候object name的前缀将替换成这个值。如果前缀为空则将这个字符串插入在object namde的最前面。ReplaceKeyWith和ReplaceKeyPrefixWith这两个节点只能共存一个。 比如指定了KeyPrefixEquals为abc/，指定了ReplaceKeyPrefixWith为def/，那么如果访问的object为abc/test.txt那么Location头则为`http://www.test.com/def/test.txt`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType不为External或者AliCDN时需要指定。

ReplaceKeyWith	字符串	Redirect的时候object name将替换成这个值，可以支持变量（目前支持的变量是${key}，代表着该请求中的object name）。ReplaceKeyWith和ReplaceKeyPrefixWith这两个节点只能共存一个。 比如设置ReplaceKeyWith为prefix/${key}.suffix，访问的object为test，那么Location头则为`http://www.test.com/prefix/test.suffix`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当rectType不为External或者AliCDN时需要指定。

细节分析

- 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。
- 如果你想使用自己的域名来访问基于bucket的静态网站，可以通过域名CNAME来实现。具体配置方法请参见绑定自定义域名。
- 用户将一个bucket设置成静态网站托管模式时，必须指定索引页面，错误页面则是可选的。
- 用户将一个bucket设置成静态网站托管模式时，指定的索引页面和错误页面是该bucket内的一个object。
- 在将一个bucket设置成静态网站托管模式后，对静态网站根域名的匿名访问，OSS将返回索引页面；对静态网站根域名的签名访问，OSS将返回Get Bucket结果。
- 如果用户上传了Content-MD5请求头，OSS会计算body的Content-MD5并检查一致性，如果不一致，将返回InvalidDigest错误码。

示例

请求示例：

```
PUT /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 209
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZHiA=

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

```
<Key>error.html</Key>
</ErrorDocument>
</WebsiteConfiguration>
```

返回示例：

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

完整代码：

```
PUT /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 2064
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBNgkzzxcQMf8u:sNKIHT6ci/z231yIT5vYnetDLu4=
User-Agent: aliyun-sdk-python-test/0.4.0

<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
<RoutingRules>
<RoutingRule>
<RuleNumber>1</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
</Condition>
<Redirect>
<RedirectType>Mirror</RedirectType>
<PassQueryString>true</PassQueryString>
<MirrorURL>http://www.test.com/</MirrorURL>
<MirrorPassQueryString>true</MirrorPassQueryString>
<MirrorFollowRedirect>true</MirrorFollowRedirect>
<MirrorCheckMd5>false</MirrorCheckMd5>
<MirrorHeaders>
<PassAll>true</PassAll>
<Pass>myheader-key1</Pass>
<Pass>myheader-key2</Pass>
<Remove>myheader-key3</Remove>
<Remove>myheader-key4</Remove>
<Set>
<Key>myheader-key5</Key>
<Value>myheader-value5</Value>
</Set>
</MirrorHeaders>
</Redirect>
</RoutingRule>
<RoutingRule>
<RuleNumber>2</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
<IncludeHeader>
```

```
<Key>host</Key>
<Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
</IncludeHeader>
</Condition>
<Redirect>
<RedirectType>AliCDN</RedirectType>
<Protocol>http</Protocol>
<HostName>www.test.com</HostName>
<PassQueryString>false</PassQueryString>
<ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
<HttpRedirectCode>301</HttpRedirectCode>
</Redirect>
</RoutingRule>
</RoutingRules>
</WebsiteConfiguration>

HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5B5ADFD6ED3CC49176CBE29D
x-oss-server-time: 47
```

5.5 PutBucketReferer

PutBucketReferer接口用于设置一个Bucket的Referer访问白名单和是否允许Referer字段为空的请求访问。

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）

表 **5-5:** 请求元素

名称	类型	是否必需	描述
RefererConfiguration	容器	是	保存Referer配置内容的容器 子节点：AllowEmptyReferer节点、RefererList节点 父节点：无
AllowEmptyReferer	枚举字符串	是	指定是否允许referer字段为空的请求访问。取值： • `true` (默认值) • `false` 父节点：RefererConfiguration
RefererList	容器	是	保存referer访问白名单的容器。 父节点：RefererConfiguration 子节点：Referer
Referer	字符串	否	指定一条referer访问白名单。 父节点：RefererList

细节分析

- 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。
- 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。
- 此操作将用RefererList中的白名单列表覆盖之前配置的白名单列表，当用户上传的RefererList为空时（不包含Referer请求元素），此操作会覆盖已配置的白名单列表，即删除之前配置的RefererList。
- 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

不包含**Referer**的请求示例：

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

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZHiA=

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

包含**Referer**的请求示例：

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

<?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
```

5.6 PutBucketLifecycle

Bucket的拥有者可以通过**PutBucketLifecycle**接口来设置Bucket的Lifecycle。Lifecycle开启后，OSS将按照配置，定期自动删除与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)

表 5-6: 请求元素

名称	类型	是否必需	描述
CreatedBeforeDate	字符串	Days和CreatedBeforeDate二选一	指定哪天之前的数据被执行Lifecycle规则。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，表示将最后更新时间早于 2002-10-11T00:00:00.000Z 的Object删除或转换成其他存储类型，等于或晚于这个时间的Object不会被删除或转储。 父节点：Expiration或者AbortMultipartUpload
Days	正整数	Days和CreatedBeforeDate二选一	指定规则在对象最后更新时间过后多少天生效。 父节点：Expiration
Expiration	容器	否	指定Object规则的过期属性。 子节点：Days或CreatedBeforeDate 父节点：Rule
AbortMultipartUpload	容器	否	指定未完成的Part规则的过期属性。 子节点：Days或CreatedBeforeDate 父节点：Rule
ID	字符串	否	规则唯一的ID。最多由255字节组成。当用户没有指定，或者该值为空时，OSS会为用户生成一个唯一值。 子节点：无 父节点：Rule

名称	类型	是否必需	描述
LifecycleConfiguration	容器	是	Lifecycle配置的容器，最多可容纳1000条规则。 子节点：Rule 父节点：无
Prefix	字符串	是	指定规则所适用的前缀。只有匹配前缀的对象才可能被该规则所影响。不可重叠。 子节点：无 父节点：Rule
Rule	容器	是	表述一条规则 子节点：ID，Prefix，Status，Expiration 父节点：LifecycleConfiguration
Status	字符串	是	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。 父节点：Rule 有效值：Enabled，Disabled
StorageClass	字符串	如果父节点Transition已设置，则为必需	指定对象转储到OSS的目标存储类型。取值： • IA • Archive 父节点：Transition
Transition	容器	否	指定Object在有效生命周期中，OSS何时将对象转储到IA或者Archive存储类型 。

细节分析

- 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。
- 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。
- 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。

转储注意事项：

- 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。

- 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。
- 支持IA Bucket中Objects转储为Archive存储类型。
- 不支持Archvie Bucket创建转储规则。
- 不支持IA类型Object转储为Standard。
- 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 qn6qrrqxo2oawuk53otfjbyc: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
```

5.7 GetBucket (ListObject)

GetBucket接口可用来列出 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。

表 5-7: 请求参数

名称	类型	是否必需	描述
delimiter	字符串	否	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。 默认值：无
marker	字符串	否	设定结果从marker之后按字母排序的第一个开始返回。 默认值：无

名称	类型	是否必需	描述
max-keys	字符串	否	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。 默认值：100
prefix	字符串	否	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。 默认值：无
encoding-type	字符串	否	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。 默认值：无 可选值：url

响应元素(Response Elements)

表 5-8: 响应元素

名称	类型	描述
Contents	容器	保存每个返回Object meta的容器。 父节点：ListBucketResult
CommonPrefixes	字符串	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。 父节点：ListBucketResult
Delimiter	字符串	是一个用于对Object名字进行分组的字符。所有名字包含指定的前缀且第一次出现delimiter字符之间的object作为一组元素CommonPrefixes。 父节点：ListBucketResult
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，那会对返回结果中的Delimiter、Marker、Prefix、NextMarker和Key这些元素进行编码。 父节点：ListBucketResult

名称	类型	描述
DisplayName	字符串	Object 拥有者的名字。 父节点：ListBucketResult.Contents.Owner
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，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。 父节点：ListBucketResult.Contents
ID	字符串	Bucket拥有者的用户ID。 父节点：ListBucketResult.Contents.Owner
IsTruncated	枚举字符串	指明是否所有的结果都已经返回；“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。 有效值：`true`、`false` 父节点：ListBucketResult
Key	字符串	Object的Key. 父节点：ListBucketResult.Contents
LastModified	时间	Object最后被修改的时间。 父节点：ListBucketResult.Contents
ListBucket Result	容器	保存Get Bucket请求结果的容器。 子节点：Name, Prefix, Marker, MaxKeys, Delimiter, IsTruncated, Nextmarker, Contents 父节点：None
Marker	字符串	标明这次Get Bucket（List Object）的起点。 父节点：ListBucketResult
MaxKeys	字符串	响应请求内返回结果的最大数目。 父节点：ListBucketResult
Name	字符串	Bucket名字 父节点：ListBucketResult
Owner	容器	保存Bucket拥有者信息的容器。 子节点：DisplayName, ID 父节点：ListBucketResult
Prefix	字符串	本次查询结果的开始前缀。 父节点：ListBucketResult

名称	类型	描述
Size	字符串	Object的字节数。 父节点：ListBucketResult.Contents
StorageClass	字符串	Object的存储类型，目前只能是“Standard”类。 父节点：ListBucketResult.Contents

细节分析

- Object中用户自定义的meta，在GetBucket请求时不会返回。
- 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。
- 如果没有访问该Bucket的权限，返回403 Forbidden错误，错误码：AccessDenied。
- 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。
- 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。
- 若prefix，marker，delimiter参数不符合长度要求，返回400 Bad Request。错误码：InvalidArgument。
- prefix，marker用来实现分页显示效果，参数的长度必须小于1024字节。
- 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 qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0y
kboO4M=
```

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>&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>
    <Contents>
        <Key>oss.jpg</Key>
        <LastModified>2012-02-24T06:07:48.000Z</LastModified>
        <ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;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/ZghT7cGa0y
kboO4M=
```

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>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;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>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;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 qn6qrrqxo2oawuk53otfjbyc:DNrnx7xHk3sgysx7I8U9
I9IY1vY=
```

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>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</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>
```

5.8 GetBucketLocation

GetBucketLocation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)**

表 **5-9:** 响应元素

名称	类型	描述
LocationConstraint	字符串	Bucket所在的区域。 有效值： oss-cn-hangzhou、oss-cn-qingdao、oss-cn-beijing、oss-cn-hongkong、oss-cn-shenzhen、oss-cn-shanghai

细节分析

- 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。
- 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:ceOEyZavKY4QcjoUWYSp
YbJ3naA=
```

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 >
```

5.9 GetBucketInfo

`GetBucketInfo`接口用于查看Bucket的相关信息。

可查看内容包含如下：

- 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)

表 5-10: 响应元素

名称	类型	描述
`BucketInfo`	容器	保存 Bucket 信息内容的容器 子节点：Bucket节点 父节点：无
`Bucket`	容器	保存 Bucket 具体信息的容器 父节点：BucketInfo 节点

名称	类型	描述
CreationDate	时间	Bucket 创建时间。时间格式 2013-07-31T10:56:21.000Z 父节点：BucketInfo.Bucket
ExtranetEndpoint	字符串	Bucket 访问的外网域名 父节点：BucketInfo.Bucket
IntranetEndpoint	字符串	同区域 ECS 访问 Bucket 的内网域名 父节点：BucketInfo.Bucket
Location	字符串	Bucket 所在数据中心的区域 父节点：BucketInfo.Bucket
Name	字符串	Bucket 名称 父节点：BucketInfo.Bucket
Owner	容器	用于存放 Bucket 拥有者信息的容器。 父节点：BucketInfo.Bucket
ID	字符串	Bucket 拥有者的用户 ID。 父节点：BucketInfo.Bucket.Owner
DisplayName	字符串	Bucket拥有者的名称 (目前和ID一致)。 父节点：BucketInfo.Bucket.Owner
AccessControlList	容器	存储 ACL 信息的容器 父节点：BucketInfo.Bucket
Grant	枚举字符串	Bucket的ACL权限。 有效值：`private`、`public-read`、`public-read-write` 父节点：BucketInfo.Bucket.AccessControlList
DataRedundancyType	枚举字符串	数据容灾类型 有效值：LRS、ZRS 父节点：BucketInfo.Bucket
Comment	字符串	未知
StorageClass	字符串	Bucket 存储类型 有效值：`Standard`、`IA`、`Archive`

细节分析

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

- 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。
- 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+zNhH1AcF51
NNHD39zXw=
```

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>
```

5.10 GetBucketLogging

GetBucketLogging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)

表 **5-11:** 响应元素

名称	类型	描述
BucketLoggingStatus	容器	访问日志状态信息的容器。 子元素：LoggingEnabled 父元素：无
LoggingEnabled	容器	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。 子元素：TargetBucket, TargetPrefix 父元素：BucketLoggingStatus
TargetBucket	字符	指定存放访问日志的Bucket。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

名称	类型	描述
TargetPrefix	字符	指定最终被保存的访问日志文件前缀。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled

细节分析

- 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
- 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。
- 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:ceOEyZavKY4QcjoUWYSp
YbJ3naA=
```

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>
```

5.11 GetBucketWebsite

GetBucketWebsite接口用于查看bucket的静态网站托管状态以及跳转规则。

请求语法

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

响应元素(Response Elements)

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

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

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

MirrorURL	字符串	只有在RedirectType为Mirror时有意义，表示镜像回源的源站地址。 如以http://或者https://开头，必须以正斜线（/）结尾，OSS会在此字符串的基础上加上object构成回源的url。比如指定为`http://www.test.com/`，访问的obejct是”myobject”，则回源的url为`http://www.test.com/myobject`，如果指定为`http://www.test.com/dir1/`，那么回源的url为`http://www.test.com/dir1/myobject`。 父节点：Redirect	有条件，如果RedirectType为Mirror则必须指定
MirrorPassQueryString	bool型	与PassQueryString作用相同，优先级比PassQueryString高，但只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorFollowRedirect	bool型	如果镜像回源获取的结果是3xx，是否要继续跳转到指定的Location获取数据。 比如发起镜像回源请求时，如果源站返回了302，并且指定了Location。如果此项为true，则oss会继续请求Location指定的地址（最多跳转10此次，如果超过10次，则返回镜像回源失败）。如果为false，那么OSS就会返回302，并将Location透传出去。只有在RedirectType为Mirror时生效。 默认值：true 父节点：Redirect	否
MirrorCheckMd5	bool型	是否要检查回源body的md5。 当此项为true且源站返回的response中含有Content-Md5头时，OSS检查拉取的数据md5是否与此header匹配，如果不匹配，则不保存在oss上。只有在RedirectType为Mirror时生效。 默认值：false 父节点：Redirect	否
MirrorHeaders	容器	用于指定镜像回源时携带的header。只有在RedirectType为Mirror时生效。 父节点：Redirect	否

PassAll	bool型	是否透传请求中所有的header（除了保留的几个header以及以oss-/x-oss-/x-drs-开头的header）到源站。只有在RedirectType为Mirror时生效。 默认值：false 父节点：MirrorHeaders	否
Pass	字符串	透传指定的header到源站，最多10个，每个header长度最多1024个字节，字符集为0-9、A-Z、a-z以及横杠。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Remove	字符串	禁止透传指定的header到源站，这个字段可以重复，最多10个，一般与PassAll一起使用。每个header长度最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Set	容器	设置一个header传到源站，不管请求中是否携带这些指定的header，回源时都会设置这些header。该容器可以重复，最多10组。只有在RedirectType为Mirror时生效。 父节点：MirrorHeaders	否
Key	字符串	设置header的key，最多1024个字节，字符集与Pass相同。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定
Value	字符串	设置header的value，最多1024个字节，不能出现”\r\n”。只有在RedirectType为Mirror时生效。 父节点：Set	有条件，当父节点Set指定时必须指定。
Protocol	字符串	跳转时的协议，只能取值为http或者https。比如访问的文件为test，设定跳转到`www.test.com`，而且Protocol字段为https，那么Location头为`https://www.test.com/test`。只有在RedirectType为External或者AliCDN时生效。 父节点：Redirect	有条件，当RedirectType不为External或者AliCDN时需要指定。

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

细节分析

- 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
- 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。
- 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+zNhH1AcF51
NNHD39zXw=
```

已设置**LOG**规则的返回示例：

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

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

未设置**LOG**规则的返回示例：

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:56:46 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS

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

```
</Error>
```

完整代码：

```
GET /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:07:41 GMT
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBNgkzzxcQMf8u:0JzamofmyR5Wa0rsU9HUWomxsus=
User-Agent: aliyun-sdk-python-test/0.4.0

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

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
<RoutingRules>
<RoutingRule>
<RuleNumber>1</RuleNumber>
<Condition>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
</Condition>
<Redirect>
<RedirectType>Mirror</RedirectType>
<PassQueryString>true</PassQueryString>
<MirrorURL>http://www.test.com/</MirrorURL>
<MirrorPassQueryString>true</MirrorPassQueryString>
<MirrorFollowRedirect>true</MirrorFollowRedirect>
<MirrorCheckMd5>false</MirrorCheckMd5>
<MirrorHeaders>
<PassAll>true</PassAll>
<Pass>myheader-key1</Pass>
<Pass>myheader-key2</Pass>
<Remove>myheader-key3</Remove>
<Remove>myheader-key4</Remove>
<Set>
<Key>myheader-key5</Key>
<Value>myheader-value5</Value>
</Set>
</MirrorHeaders>
</Redirect>
</RoutingRule>
<RoutingRule>
<RuleNumber>2</RuleNumber>
<Condition>
<IncludeHeader>
<Key>host</Key>
<Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
</IncludeHeader>
<KeyPrefixEquals>abc/</KeyPrefixEquals>
<HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
```

```
</Condition>
<Redirect>
<RedirectType>AliCDN</RedirectType>
<Protocol>http</Protocol>
<HostName>www.test.com</HostName>
<PassQueryString>false</PassQueryString>
<ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
<HttpRedirectCode>301</HttpRedirectCode>
</Redirect>
</RoutingRule>
</RoutingRules>
</WebsiteConfiguration>
```

5.12 GetBucketReferer

GetBucketReferer操作用于查看bucket的Referer相关配置。

请求语法

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

响应元素(Response Elements)

表 5-12: 响应元素

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

细节分析

- 如果Bucket不存在，返回404错误。错误码：NoSuchBucket。
- 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。
- 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+zNhH1AcF51
NNHD39zXw=
```

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>
```

5.13 GetBucketLifecycle

GetBucketLifecycle用于查看Bucket的Lifecycle配置。

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

细节分析

- 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。
- 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:ceOEyZavKY4QcjoUWYSp
YbJ3naA=
```

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> BucketName.oss.example.com</HostId>
</Error>
```

5.14 DeleteBucket

DeleteBucket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

- 如果Bucket不存在，返回404 no content错误。错误码：NoSuchBucket。
- 为了防止误删除的发生，OSS不允许用户删除一个非空的Bucket。
- 如果试图删除一个不为空的Bucket，返回409 Conflict错误，错误码：BucketNotEmpty。
- 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:ceOEyZavKY4QcjoUWYSp
YbJ3naA=
```

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
```

5.15 DeleteBucketLogging

DeleteBucketLogging接口用于关闭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

- 如果Bucket不存在，返回404 no content错误，错误码：NoSuchBucket。
- 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。
- 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 qn6qrrqxo2oawuk53otfjbyc: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
```

5.16 DeleteBucketWebsite

DeleteBucketWebsite操作用于关闭bucket的静态网站托管模式以及跳转规则。

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

- 如果Bucket不存在，返回404 no content错误，错误码：NoSuchBucket。
- 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: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWi
cOMEkVg=
```

返回示例：

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

完整代码：

```
DELETE /?website HTTP/1.1
Date: Fri, 27 Jul 2018 09:10:52 GMT
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBNgkzzxcQMf8u:qPrKwuMaarA4Tfk1pqTCylFs1jY=
User-Agent: aliyun-sdk-python-test/0.4.0

HTTP/1.1 204 No Content
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:10:52 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5B5AE19C188DC1CE81DAD7C8
```

5.17 DeleteBucketLifecycle

通过**DeleteBucketLifecycle**来删除指定 Bucket 的生命周期规则 。

请求语法

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

```
Authorization: SignatureValue
```

细节分析

- 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。
- 如果 Bucket 不存在，返回 404 Not Found 错误，错误码：NoSuchBucket。
- 如果删除不存在的 Lifecycle，返回 204 状态码。
- 只有 Bucket 的拥有者才能删除 Bucket 的生命周期规则。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 /?lifecycle HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:35 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/
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
```

6 关于Object操作

6.1 PutObject

PutObject接口用于上传文件。

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

表 6-1: 请求Header

名称	类型	描述
Authorization	字符串	表示请求本身已被授权；更详细描述请参照*RFC2616*。 默认值：无
Cache-Control	字符串	指定该Object被下载时的网页的缓存行为；更详细描述请参照*RFC2616*。 默认值：无
Content-Disposition	字符串	指定该Object被下载时的名称；更详细描述请参照*RFC2616*。 默认值：无
Content-Encoding	字符串	指定该Object被下载时的内容编码格式；更详细描述请参照*RFC2616*。 默认值：无
Content-Md5	字符串	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编码作为一个消息的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。 默认值：无 限制：无

名称	类型	描述
Content-Length	字符串	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。
ETag	字符串	ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。不建议用户使用ETag来作为Object内容的MD5校验数据完整性。 默认值：无
Expires	字符串	过期时间；更详细描述请参照*RFC2616*。 默认值：无 **注意:** OSS不会对这个值进行限制和验证。
x-oss-server-side-encryption	字符串	指定OSS创建Object时的服务器端加密编码算法。 合法值：`AES256` 或 `KMS` **注意:** 您需要购买KMS套件，才可以使用KMS加密算法，否则会报KmsServiceNotEnabled错误码
x-oss-server-side-encryption-key-id	字符串	表示KMS托管的用户主密钥。 该参数在**x-oss-server-side-encryption**为KMS时有效。
x-oss-object-acl	字符串	指定OSS创建Object时的访问权限。 合法值：`public-read`，`private`，`public-read-write`

名称	类型	描述
x-oss-storage-class	字符串	指定Object的存储类型。 取值： • Standard • IA • Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、PutObjectSymlink、CopyObject。 **注意:** • 如果StorageClass的值不合法，返回400 错误。错误码：InvalidArgument。 • 对于任意存储类型Bucket，若上传Object时指定该值，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。

细节分析

- 对于试图添加的Object：
  - 所在的Bucket没有访问权限，返回403 Forbidden错误。错误码：AccessDenied。
  - 已经存在同名文件，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。
  - 所在的Bucket不存在，返回404 Not Found错误。错误码：NoSuchBucket。
- 如果传入的Object key长度大于1023字节，返回400 Bad Request。错误码：InvalidObjectName。
- Content length
  - 如果请求头中的**Content-Length**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。
  - 如果添加文件长度超过5G，返回400 Bad Request错误。错误码：InvalidArgument。
  - 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错误。错误码：RequestTimeout。
- HTTP Header
  - OSS支持 HTTP 协议规定的5个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

该Object被下载时，相应的Header值会被自动设置成上传时的值。Hearder 字段详情请参见 *RFC2616*。

- 如果Header不是*chunked encoding*编码方式，且没有加入Content length参数，将返回411 Length Required错误。错误码：MissingContentLength。
- PutObject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 Bad Request错误。错误码：InvalidEncryptionAlgorithmError。指定该Header后，在响应头中也会返回该Header，OSS会对上传的Object进行加密编码存储，当该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。
- PutObject时，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8KB。User meta支持短横线（-）、空格 、双引号、 数字、英文字母（a-z, A-Z），不支持下划线（_）在内的其他字符。

常见问题

Content-Md5计算方式错误

标准中定义的算法为： 先计算Md5加密的二进制数组（128位），再对这个二进制进行base64编码（而不是对32位字符串编码）。

以上传的内容`0123456789`为例，计算该字符串的Content-Md5。

用Python来实现，正确计算的代码为：

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```

注意:

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

常见错误是直接对计算出的32位字符串编码进行base64编码。 例如：`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编码后的结果为： `>>> base64.b64encode(hash.hexdigest()) 'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='`

示例

- 示例1

  简单上传的请求示例：

```
PUT /test.txt HTTP/1.1
Host: test.oss-cn-zhangjiakou.aliyuncs.com
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
Content-Type: text/plain
date: Tue, 04 Dec 2018 15:56:37 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+
dcGKw5x2PRrk=
Transfer-Encoding: chunked
```

  响应示例：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 04 Dec 2018 15:56:38 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5C06A3B67B8B5A3DA422299D
ETag: "D41D8CD98F00B204E9800998ECF8427E"
x-oss-hash-crc64ecma: 0
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
x-oss-server-time: 7
```

- 示例2

  带有归档存储类型的请求示例：

```
PUT /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com Cache-control: no-
cache
Expires: Fri, 28 Feb 2012 05:38:42 GMT
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Fri, 24 Feb 2012 06:03:28 GMT
Content-Type: image/jpg
Content-Length: 344606
x-oss-storage-class: Archive
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+
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-Type: image/jpg
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5650BD72207FB30443962F9A
```

```
x-oss-bucket-version: 1418321259

ETag: "A797938C31D59EDD08D86188F6D5B872"
```

- 注意:

  更多Put Object示例代码请参见*Python SDK*上传文件。

6.2 CopyObject

CopyObject操作用于在 Bucket 内或同地域的 Bucket 之间拷贝文件。

通过**CopyObject**接口可以发送一个 Put 请求给 OSS，并在 Put 请求 Header 中添加元素`x-oss-copy-source`来指定源 Object。OSS 会自动判断出这是一个拷贝操作，并直接在服务器端执行该操作。如果拷贝成功，则返回目标 Object 信息给用户。

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	类型	描述
x-oss-copy-source	字符串	指定拷贝的源地址。 默认值：无
x-oss-copy-source-if-match	字符串	如果源 Object 的 ETag 值和用户提供的ETag 相等，则执行拷贝操作，并返回 200 OK；否则返回 412 HTTP 错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-none-match	字符串	如果源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 OK；否则返回 304 HTTP 错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-unmodified-since	字符串	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常拷贝文件，并返回 200 OK；否则返回 412 precondition failed 错误。 默认值：无

名称	类型	描述
x-oss-copy-source-if-modified-since	字符串	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）。 默认值：无
x-oss-metadata-directive	字符串	指定如何设置目标 Object 的元信息。有效值为 COPY 和 REPLACE 。 • COPY （默认值）：复制源 Object 的元信息到目标 Object。注意，源 Object 的**x-oss-server-side-encryption**不会被复制，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Header来决定。 • REPLACE：忽略源 Object 的元信息，直接采用用户请求中指定的元信息。 **注意:** 如果拷贝操作的源 Object 地址和目标Object 地址相同，则无论**x-oss-metadata-directive**为何值，都会直接替换源 Object 的元信息。

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

名称	类型	描述
x-oss-storage-class	字符串	指定Object的存储类型。 取值： • Standard • IA • Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。 注意: • 如果 StorageClass 的值不合法，返回 400 错误。错误码：InvalidArgument。 • PutObjectSymlink 的存储类型建议不要指定为 IA 或 Archive 类型（因为 IA 与 Archive 类型的单个文件如不足 64KB，会按 64KB计量计费）。 • 对于任意存储类型 Bucket，若上传 Object 时指定该值，则此次上传的 Object 将存储为指定的类型。例如，在 IA 类型的 Bucket 中上传 Object 时，若指定 x-oss-storage-class 为 Standard，则该 Object 直接存储为Standard。 • 更改 Object 存储类型涉及到数据覆盖，如果 IA 或 Archive 类型 Object 分别在创建后 30 和 60 天内被覆盖，则它们会产生提前删除费用。比如，IA 类型Object 创建 10 天后，被覆盖成 Archive 或 Standard，则会产生 20 天的提前删除费用。

响应元素(Response Elements)

表 **6-2:** 响应元素

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

细节分析

- 限制条件
  - 仅支持小于 1GB 的文件。如果要拷贝大于 1GB 的文件，必须使用 MultipartUpload 操作，详情请参见*UploadPartCopy*。
  - 请求者必须对源 Object 有读权限。
  - 源 Object 和目标 Object 必须属于同一地域（数据中心）。
  - 不能拷贝通过追加上传方式产生的 Object。
  - 如果源 Object 为软链接，只拷贝软链接，不拷贝软链接指向的文件内容。
- 计量计费
  - 对源 Object 所在的 Bucket 增加一次 Get 请求次数。
  - 对目标 Object 所在的 Bucket 增加一次 Put 请求次数。
  - 对目标 Object 所在的 Bucket 增加相应的存储量。
- 预判断请求Header
  - 四个预判断请求H eader（**x-oss-copy-source-if-match**、**x-oss-copy-source-if-none-match**、**x-oss-copy-source-if-unmodified-since**、**x-oss-copy-source-if-modified-since**）中，任意个数可同时出现。相应逻辑请参见 GetObject 操作中的细节分析。
  - 拷贝操作涉及到的请求 Header 都是以**x-oss-**开头，所以要加到签名字符串中。

示例

- 示例 1

  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-storage-class: Archive
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>
```

- 示例 2

  请求示例：

```
PUT /test%2FAK.txt HTTP/1.1
Host: tesx.oss-cn-zhangjiakou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
x-oss-copy-source: /test/AK.txt
date: Fri, 28 Dec 2018 09:41:55 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:gmnwPKuu20LQEjd+iPkL259A
+n0=
Content-Length: 0
```

  返回示例：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 28 Dec 2018 09:41:56 GMT
Content-Type: application/xml
Content-Length: 184
Connection: keep-alive
x-oss-request-id: 5C25EFE4462CE00EC6D87156
ETag: "F2064A169EE92E9775EE5324D0B1682E"
x-oss-hash-crc64ecma: 12753002859196105360
x-oss-server-time: 150

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
  <ETag>"F2064A169EE92E9775EE5324D0B1682E"</ETag>
  <LastModified>2018-12-28T09:41:56.000Z</LastModified>
</CopyObjectResult>
```

  注意:

  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。进行 COPY 操作时，生成的 Object 不保证具有 crc64 的值。

6.3 GetObject

GetObject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

OSS支持用户在发送GET请求时，可以自定义OSS返回请求中的一些Header。

注意:

用户发送的GET请求必须携带签名。

自定义返回请求的Header包括 ：

名称	类型	描述
response-content-type	字符串	设置OSS返回请求的content-type头 默认值：无
response-content-language	字符串	设置OSS返回请求的content-language头 默认值：无
response-expires	字符串	设置OSS返回请求的expires头 默认值：无
response-cache-control	字符串	设置OSS返回请求的cache-control头 默认值：无
response-content-disposition	字符串	设置OSS返回请求的content-disposition头 默认值：无
response-content-encoding	字符串	设置OSS返回请求的content-encoding头 默认值：无

注意:

在自定义OSS返回请求中的一些Header，只有请求处理成功（即返回码为 200 OK ），OSS才会将请求的Header设置成用户GET请求参数中指定的值。

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。

请求Header

名称	类型	描述
Range	字符串	指定文件传输的范围。 默认值：无 在请求Header中使用Range参数 • 符合范围规范，返回消息中则会包含整个Object的长度和此次返回的范围。例如：Content-Range: bytes 0-9/44，表示整个Object长度为44，此次返回的范围为0-9。 • 不符合范围规范，则传送整个Object，并且不在结果中提及Content-Range。
If-Modified-Since	字符串	如果指定的时间早于实际修改时间或指定的时间不符合规范，直接返回Object，并返回200 OK；否则返回304 not modified。 默认值：无 时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT
If-Unmodified-Since	字符串	如果传入参数中的时间等于或者晚于Object实际修改时间，则正常传输Object，并返回200 OK；否则返回412 precondition failed错误 。 默认值：无 时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT If-Modified-Since和If-Unmodified-Since可以同时使用。
If-Match	字符串	如果传入期望的ETag和Object的ETag匹配，则正常传输Object，并返回200 OK；否则返回412 precondition failed错误。 默认值：无
If-None-Match	字符串	如果传入的ETag值和Object的ETag不匹配，则正常传输Object，并返回200 OK；否则返回304 Not Modified。 默认值：无 If-Match和If-None-Match可以同时使用。

常见错误码：

错误码	HTTP状态码	2说明
NoSuchKey	404	Object不存在。
SymlinkTargetNotExist	404	Object类型为符号链接，并且目标Object不存在。

错误码	HTTP状态码	2说明
InvalidTargetType	400	Object类型为符号链接，并且目标Object类型为符号链接。
InvalidObjectState	403	对于归档类型（Archive）的Object，没有提交Restore请求或者上一次提交Restore已经超时。
InvalidObjectState	403	对于归档类型的Object，已经提交Restore请求，但数据的Restore操作还没有完成。

细节分析

- 若Object为服务器端熵编码加密存储，则在GET请求时会自动解密返回给用户，并且在响应Header中返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
- 如果返回内容进行GZIP压缩传输，需要在请求的Header中以显示方式加入Accept-Encoding: gzip，OSS会根据Object的Content-Type和Object大小（不小于1KB），判断是否返回经过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。
- 如果Object类型为符号链接，返回目标Object的内容。响应头中`Content-Length`、`ETag`、`Content-Md5` 均为目标Object的元信息；`Last-Modified`是目标Object和符号链接的最大值；其他均为符号链接的元信息。
- 如果Object类型为归档类型，需要完成Restore请求且该请求不能超时。

示例

简单的**GET**请求示例：

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

带有**Range**参数的请求示例：

```
GET //oss.jpg HTTP/1.1
Host:oss-example. oss-cn-hangzhou.aliyuncs.com
Date: Fri, 28 Feb 2012 05:38:42 GMT
Range: bytes=100-900
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtF
cCVknM=
```

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]
```

带自定义返回消息头的请求示例：

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

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 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]
```

Object类型为符号链接的请求示例：

```
GET /link-to-oss.jpg HTTP/1.1
```

```
Accept-Encoding: identity
Date: Tue, 08 Nov 2016 03:17:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Authorization:OSS 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==
```

Restore操作已经完成的请求示例：

```
GET /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 09:38:30 GMT
Authorization: OSS 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/jpg
Content-Length: 344606
Server: AliyunOSS
[354606 bytes of object data]
```

6.4 AppendObject

AppendObject接口用于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	类型	描述
Cache-Control	字符串	指定该Object被下载时的网页的缓存行为；更详细描述请参照*RFC2616*。 默认值：无
Content-Disposition	字符串	指定该Object被下载时的名称；更详细描述请参照*RFC2616*。 默认值：无
Content-Encoding	字符串	指定该Object被下载时的内容编码格式；更详细描述请参照*RFC2616*。 默认值：无
Content-MD5	字符串	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。 默认值：无 限制：无
Expires	整数	过期时间；更详细描述请参照*RFC2616*。 默认值：无
x-oss-server-side-encryption	字符串	指定OSS创建Object时的服务器端加密编码算法。 合法值：AES256 或 KMS **注意:** 您需要购买KMS套件，才可以使用KMS加密算法，否则会报KmsServiceNotEnabled错误码。
x-oss-object-acl	字符串	指定OSS创建Object时的访问权限。 合法值：public-read、private、public-read-write

名称	类型	描述
x-oss-storage-class	字符串	指定Object的存储类型。 取值： • Standard • IA • Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、PutObjectSymlink、CopyObject。 **注意:** • 如果StorageClass的值不合法，则返回400 错误。错误码：InvalidArgument。 • 对于任意存储类型Bucket，若上传Object时指定该值，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。 • 首次执行AppendObject操作时，指定该值有效，后续追加时指定该值无效。

响应Header

名称	类型	描述
x-oss-next-append-position	64位整型	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。
x-oss-hash-crc64ecma	64位整型	表明Object的64位CRC值。该64位CRC根据*ECMA-182*标准计算得出。

和其他操作的关系

- 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。
- 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。
- 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。

- Get Bucket（List Objects）请求的响应XML中，会把Appendable Object的Type设为Appendable。
- 不能使用Copy Object来拷贝一个Appendable Object，也不能改变它的服务器端加密的属性。可以使用Copy Object来改变用户自定义元信息。

细节分析

- URL参数append和position均为CanonicalizedResource，需要包含在签名中。
  - URL的参数必须包含append，用来指定这是一个Append Object操作。
  - 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。
- 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。
  - 如果没有同名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。
- 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。
- 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。
- Append Object产生的Object长度限制和Put Object一样。每次Append Object都会更新该Object的最后修改时间。
- 在position值正确的情况下，对已存在的Appendable Object追加一个长度为0的内容，该操作不会改变Object的状态。
- 处于WORM保护期的Object不支持Append Object操作。

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, 0xffffffff
ffffffffULL, 0xffffffffffffffffULL, true, true> boost_ecma;

uint64_t do_boost_crc(const char* buffer, int length)
{
    boost_ecma crc;
    crc.process_bytes(buffer, length);
    return crc.checksum();
}
```

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
x-oss-storage-class: Archive
Content-Disposition: attachment;filename=oss_download.jpg
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1717
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PR
rk=
[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
```

6.5 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

- DeleteObject要求对该Object要有写权限。
- 如果要删除的Object不存在，返回204 No Content。
- 如果要删除的Bucket不存在，返回404 Not Found。
- 如果Object类型为符号链接，只删除符号链接。

示例

请求示例：

```
DELETE /AK.txt HTTP/1.1
Host: test.oss-cn-zhangjiakou.aliyuncs.com
Accept-Encoding: identity
User-Agent: aliyun-sdk-python/2.6.0(Windows/7/AMD64;3.7.0)
Accept: */*
Connection: keep-alive
date: Wed, 02 Jan 2019 13:28:38 GMT
authorization: OSS qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+
NIUs=zV0vhg=
Content-Length: 0
```

返回示例：

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

6.6 DeleteMultipleObjects

DeleteMultipleObjects支持用户通过一个HTTP请求删除同一个Bucket中的多个Object。

DeleteMultipleObjects操作支持单次请求最多可删除1000个Object，并提供两种消息返回模式。

- 详细模式(verbose)：OSS返回的消息体中会包含每一个删除Object的结果。默认采用详细模式。
- 简单模式(quiet)：OSS返回的消息体中只包含删除过程中出错的Object结果。如果所有删除都成功的话，则没有消息体。

注意:

- 消息体最大允许2MB的内容，超过2MB返回MalformedXML错误码。
- DeleteMultipleObjects单次请求超过1000个Object返回MalformedXML错误码。

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)

DeleteMultipleObjects操作时，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。 数据类型：字符串 默认值：无 可选值：`url`

请求元素(Request Elements)

名称	类型	描述
Delete	容器	保存DeleteMultipleObjects请求的容器。 子节点：一个或多个Object元素，即可选的Quiet元素。 父节点： None
Key	字符串	被删除Object的名字。 父节点：Object
Object	容器	保存一个Object信息的容器。 子节点：Key 父节点：Delete
Quiet	枚举字符串	打开简单响应模式的开关。 有效值：`true`、`false` 默认值：`false` 父节点：Delete

响应元素(Response Elements)

名称	类型	描述
Deleted	容器	保存被成功删除的Object的容器。 子节点：Key 父节点：DeleteResult
DeleteResult	容器	保存Delete Multiple Object请求结果的容器。 子节点：Deleted 父节点：None
Key	字符串	OSS执行删除操作的Object名字。 父节点：Deleted
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，则返回的结果会对Key进行编码。 父节点：容器

细节分析

- DeleteMultipleObjects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，消息体正确才会执行删除操作。

- 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，如果不一致返回InvalidDigest错误码。
- 生成Content-MD5字段内容方法：

  1. 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。
  2. 再将该字节数组用base64算法编码，编码后得到的字符串即Content-MD5字段内容。

示例

关闭简单响应模式的请求示例：

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:+z3gBfnFAxBcBDgx27Y/
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>
```

打开简单响应模式的请求示例：

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:WuV0Jks8RyGSNQrBca64
kEExJDs=
<?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
```

6.7 HeadObject

HeadObject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	类型	描述
If-Modified-Since	字符串	如果传入参数中的时间早于实际修改时间，则返回200 OK和Object Meta；否则返回304 not modified 默认值：无

名称	类型	描述
If-Unmodified-Since	字符串	如果传入参数中的时间等于或者晚于文件实际修改时间，则返回200 OK和Object Meta；否则返回412 precondition failed错误 默认值：无
If-Match	字符串	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 默认值：无
If-None-Match	字符串	如果传入期望的ETag值和Object的ETag不匹配，则返回200 OK和Object Meta；否则返回304 Not Modified 默认值：无

响应Header

名称	类型	描述
x-oss-meta-*	字符串	以x-oss-meta-为前缀的参数作为用户自定义meta header。当用户在PutObject时设置了以x-oss-meta-为前缀的自定义meta，则响应中会包含这些自定义meta。
非x-oss-meta-开头的自定义header	字符串	当用户在PutObject时，自定义一些非x-oss-meta为前缀的Header，如x-oss-persistent-headers:`key1:base64_encode(value1)`，响应中会增加相应的自定义Header。详情请参见*OSS如何添加非x-oss-meta-开头的自定义*

名称	类型	描述
x-oss-server-side-encryption	字符串	指定OSS创建Object时的服务器端加密编码算法。 合法值：AES256 或 KMS **注意:** 您需要购买KMS套件才可以使用KMS加密算法，否则会报KmsServiceNotEnabled错误码。
x-oss-server-side-encryption-key-id	字符串	如果用户在创建Object时使用了服务端加密，且加密方法为KMS，则响应中会包含此Header，表示加密所使用的用户KMS key ID。
x-oss-storage-class	字符串	表示Object的存储类型，分别为：标准存储类型（Standard）、低频访问存储类型（Infrequent Access）、归档存储类型（Archive）。 • 标准存储类型提供高可靠、高可用、高性能的对象存储服务，能够支持频繁的数据访问。 • 低频访问存储类型适合需要长期存储但不经常被访问的数据（平均每月访问频率1到2次）。 • 归档存储类型适合需要长期存储（建议半年以上）的归档数据，在存储周期内极少被访问，数据进入到可读取状态需要1分钟的解冻时间。

名称	类型	描述
x-oss-object-type	字符串	表示Object的类型。 • 通过PutObject上传的Object类型为Normal。 • 通过AppendObject上传的Object类型为Appendable • 通过MultipartUpload上传的Object类型为Multipart。
x-oss-next-append-position	字符串	对于Appendable类型的Object会返回此Header，指明下一次请求应当提供的position。
x-oss-hash-crc64ecma	字符串	表示该Object的64位CRC值。该64位CRC根据ECMA-182标准计算得出。请注意，有些较老的Object可能没有这个Header。
x-oss-expiration	字符串	如果用户为该Object设置了生命周期规则（Lifecycle），响应中将包含x-oss-expiration header。其中expiry-date的值表示该Object的过期日期，rule-id的值表示相匹配的规则ID。

名称	类型	描述
x-oss-restore	字符串	如果Bucket类型为Archive，且用户已经提交Restore请求，则响应头中会以x-oss-restore返回该Object的Restore状态，分如下几种情况： • 如果没有提交Restore或者Restore已经超时，则不返回该字段。 • 如果已经提交Restore，且Restore没有完成，则返回的x-oss-restore值为: ongoing-request=”true”。 • 如果已经提交Restore，且Restore已经完成，则返回的x-oss-restore值为: ongoing-request=”false”, expiry-date=”Sun, 16 Apr 2017 08:12:33 GMT”，其中expiry-date是Restore完成后Object进入可读状态的过期时间。
x-oss-process-status	字符串	当用户通过MNS消息服务创建OSS事件通知后，在进行请求OSS相关操作时如果有匹配的事件通知规则，则响应中会携带这个Header，值为经过Base64编码json格式的事件通知结果。
x-oss-request-charged	字符串	当Object所属的Bucket被设置为请求者付费模式，且请求者不是Bucket的拥有者时，响应中将携带此Header，值为requester。

名称	类型	描述
Content-Md5	字符串	对于Normal类型的Object，根据RFC 1864标准对消息内容（不包括Header）计算Md5值获得128比特位数字，对该数字进行base64编码作为一个消息的Content-Md5值。Multipart和Appendable类型的文件不会返回这个Header。
Last-Modified	字符串	Object最后一次修改的日期，格式为HTTP 1.1协议中规定的GMT时间。
Access-Control-Allow-Origin	字符串	当Object所在的Bucket配置了CORS规则，如果请求的Origin满足指定的CORS规则时会在响应中包含这个Origin。
Access-Control-Allow-Methods	字符串	当Object所在的Bucket配置了CORS规则，如果请求的Access-Control-Request-Method满足指定的CORS规则时会在响应中包含允许的Methods。
Access-Control-Max-Age	字符串	当Object所在的Bucket配置了CORS规则，如果请求满足Bucket配置的CORS规则时会在响应中包含MaxAgeSeconds。
Access-Control-Allow-Headers	字符串	当Object所在的Bucket配置了CORS规则，如果请求满足指定的CORS规则时会在响应中包含这些Headers。
Access-Control-Expose-Headers	字符串	表示允许访问客户端JavaScript程序的headers列表。当Object所在的Bucket配置了CORS规则，如果请求满足指定的CORS规则时会在响应中包含ExposeHeader。

细节分析

- 不论正常返回200 OK还是非正常返回，Head Object都不返回消息体。
- 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。
- 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。
- 如果文件不存在返回404 Not Found错误。
- 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。
- 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。
- 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。
- 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。
- x-oss-storage-class展示Object的存储类型：Standard，IA，Archive。
- 如果Bucket类型为Archive，且用户已经提交Restore请求，则响应头中会以x-oss-restore返回该Object的Restore状态，分如下几种情况：
  - 如果没有提交Restore或者Restore已经超时，则不返回该字段。
  - 如果已经提交Restore，且Restore没有时完成，则返回的x-oss-restore值为: ongoing-request=”true”。
  - 如果已经提交Restore，且Restore已经完成，则返回的x-oss-restore值为: ongoing-request=”false”, expiry-date=”Sun, 16 Apr 2017 08:12:33 GMT”，其中expiry-date是Restore完成后文件进入可读状态的过期时间。

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:JbzF2LxZUtanlJ5dLA092wpDC/
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/jpg
Connection: keep-alive
Server: AliyunOSS
```

提交Restore请求但restore没有完成时的请求示例

```
HEAD /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 07:32:52 GMT
Authorization: OSS e1Unnbm1rgdnpI:KKxkdNrUBu2t1kqlDh0MLbDb99I=
```

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

提交Restore请求且restore已经完成时的请求示例

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

6.8 GetObjectMeta

GetObjectMeta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

- 无论正常返回还是非正常返回，Get Object Meta均不返回消息体。
- Get Object Meta需包含objectMeta请求参数，否则表示Get Object请求。
- 如果文件不存在返回404 Not Found错误。
- 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。
- 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+WwIfNm0Fmg
brQ0=
```

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
```

6.9 PutObjectACL

PutObjectACL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，在请求头中带上x-oss-object-acl来设置Object ACL，效果与PutObjectA ACL等同。例如PutObject时在请求头中带上x-oss-object-acl可以在上传一个Object的同时设置某个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才有权限调用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:KU5h8YMUC78M30dXqf3J
xrTZHiA=
```

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
```

6.10 GetObjectACL

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

细节分析

- 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。
- 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+WwIfNm0Fmg
brQ0=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 253
Content-Tupe: application/xml
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" ?>
<AccessControlPolicy>
    <Owner>
        <ID>00220120222</ID>
        <DisplayName>00220120222</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>public-read </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

6.11 PostObject

PostObject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	类型	描述	必须
OSSAccessKeyId	字符串	Bucket 拥有者的Access Key Id。 默认值：无 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	字符串	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。 默认值：无 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

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

名称	类型	描述	必须
success_action_status	字符串	未指定success_action_redirect表单域时，该表单域指定了上传成功后返回给客户端的状态码。 接受值为200, 201, 204（默认）。 如果该域的值为200或者204，OSS返回一个空文档和相应的状态码。 如果该域的值设置为201，OSS返回一个XML文件和201状态码。 如果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。 默认值：无	
x-oss-meta-*	字符串	用户指定的user meta值。 OSS不会检查或者使用该值。 默认值：无	可选
x-oss-server-side-encryption	字符串	指定OSS创建object时的服务器端加密编码算法。 合法值：AES256	可选
x-oss-server-side-encryption-key-id	字符串	表示KMS托管的用户主密钥。 该参数在**x-oss-server-side-encryption**为KMS时有效。	可选
x-oss-object-acl	字符串	指定oss创建object时的访问权限。 合法值：public-read，private，public-read-write	可选
x-oss-security-token	字符串	若本次访问是使用STS临时授权方式，则需要指定该项为SecurityToken的值，同时OSSAccessKeyId需要使用与之配对的临时AccessKeyId，计算签名时，与使用普通AccessKeyId签名方式一致。 默认值：无	可选

响应Header

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

响应元素(Response Elements)

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

细节分析

- 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。
- 无论bucket是否为public-read-write，一旦上传OSSAccessKeyId, policy, Signature表单域中的任意一个，则另两个表单域为必选项，缺失时OSS会返回错误码：InvalidArgument。
- post操作提交表单编码必须为“multipart/form-data”，即header中Content-Type为`multipart/form-data;boundary=xxxxxx` 这样的形式，boundary为边界字符串。
- 提交表单的URL为bucket域名即可，不需要在URL中指定object。即请求行是`POST / HTTP/1.1`，不能写成`POST /ObjectName HTTP/1.1`。
- 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。

- 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。
- 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。
- 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。
- 如果POST请求中包含Header签名信息或URL签名信息，OSS不会对它们做检查。
- 如果请求中携带以x-oss-meta-为前缀的表单域，则视为user meta，比如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的user meta总大小不能超过8k。
- 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。
- 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。
- 表单域为大小写不敏感的，但是表单域的值为大小写敏感的。

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/objectName.txt
--9431149156168
Content-Disposition: form-data; name="success_action_status"
200
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
content_disposition
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-uuid"
uuid
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-tag"
metadata
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
44CF9590006BF252F707
--9431149156168
Content-Disposition: form-data; name="policy"
eyJleHBpcmF0aW9uIjoiMjAxMy0xMi0wMVQxMjowMDowMFoiLCJjb25kaXRp
b25zIjpbWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDAsIDEwNDg1NzYwXSx7
ImJ1Y2tldCI6ImFoYWhhIn0sIHsiQSI6ICJhIn0seyJrZXkiOiAiQUJDIn1dfQ==
--9431149156168
Content-Disposition: form-data; name="Signature"
kZoYNv66bsmc10+dcGKw5x2PRrk=
--9431149156168
```

```
Content-Disposition: form-data; name="file"; filename="MyFilename.
txt"
Content-Type: text/plain
abcdefg
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OSS
--9431149156168--
```

- 返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 61d2042d-1b68-6708-5906-33d81921362e
Date: Fri, 24 Feb 2014 06:03:28 GMT
ETag: 5B3C1A2E053D763E1B002CC607C5A0FE
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

Post Policy

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

policy示例

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

Post policy中必须包含expiration和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。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contion-length-range匹配方式。

名称	描述
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

转义字符	描述
\	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，请参见*JavaScript客户端签名直传*。

6.12 Callback

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

背景信息

- 目前支持 Callback 的 API 接口有：*PutObject*、*PostObject*、*CompleteMultipartUpload*。
- 目前支持Callback的地域有：华北 2（北京）、华东 1（杭州）、华北 1（青岛）、华东 2（上海）、上海金融云、华南 1 金融云 、华南 1（深圳）、香港、华北 5（呼和浩特）、华北 3（张家口）、中东东部 1（迪拜）、亚太东北 1（日本）、欧洲中部 1（法兰克福）、亚太东南 1

（新加坡）、美国东部 1（弗吉尼亚）、美国西部 1（硅谷）、亚太东南 2（悉尼）以及亚太东南 3（吉隆坡）。

- 更多 Callback 详情请参见原理介绍。

步骤1：构造参数

- Callback 参数

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

  Json 字段如下：

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

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

Json 字段示例如下：

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

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

callbackBody 中可以设置的系统参数如下表所示：

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

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

 注意:

imageInfo 针对图片格式，如果为非图片格式

，imageInfo.height、imageInfo.width、imageInfo.format 都为空。

- callback-var 自定义参数

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

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

   注意:

  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"
}
```

 注意:

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

- PutObject() 和 CompleteMultipartUpload() 接口中 url 和 header 同时传入 callback(x-oss-callback) 或者 callback-var(x-oss-callback-var) 参数。
- callback 或者 callback-var (PostObject() 由于没有 callback-var 参数，因此没有此限制，下同)参数过长（超过 5 KB）。

- callback 或者 callback-var 参数没有经过 base64 编码或经过 base64 解码后不是合法的 Json 格式。
- callback 参数解析后 callbackUrl 字段包含的 url 超过限制（5 个），或者 url 中传入的 port 不合法，比如

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

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

步骤2：构造回调请求

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

附加方式共有三种，方式如下：

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

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

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

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

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

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

```
PUT /test.txt HTTP/1.1
Host: callback-test.oss-test.aliyun-inc.com
Accept-ncoding: identity
Content-Length: 5
x-oss-callback-var: eyJ4Om15X3ZhciI6ImZvci1jYWxsYmFjay10ZXN0In0=
User-Agent: aliyun-sdk-python/0.4.0 (Linux/2.6.32-220.23.2.ali1089.
el5.x86_64/x86_64;2.5.4)
x-oss-callback: eyJjYWxsYmFja1VybCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbm
RleC5odG1sIiwgICJjYWxsYmFja0JvZHkiOiJidWNrZXQ9JHtidWNrZXR9Jm
9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW
1lVHlwZT0ke21pbWVUeXBlfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbm
ZvLmhlaWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLndpZHRofS
ZpbWFnZUluZm8uZm9ybWF0PSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPS
R7eDpteV92YXJ9In0=
Host: callback-test.oss-test.aliyun-inc.com
Expect: 100-Continue
Date: Mon, 14 Sep 2015 12:37:27 GMT
Content-Type: text/plain
Authorization: OSS mlepou3zr4u7b14:5a74vhd4UXpmyuudV14Kaen5cY4=
Test
```

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

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

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

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

那么 POST 请求的表单域为：

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

```
Content-Disposition: form-data; name="x:var1"
value1
--9431149156168
Content-Disposition: form-data; name="x:var2"
value2
```

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

```
{ "expiration": "2014-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith" },
    {"callback": "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6
ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2
LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJj
YWxsYmFja0JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxl
bmNvZGVkIn0="},
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

步骤3：发起回调请求

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

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

（可选）步骤4：回调签名

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

- 生成签名

  签名发生在 OSS 端，采用 RSA 非对称方式。

  私钥加密生成签名的过程为：

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

   注意:

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

签名过程分为以下三步：

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

示例如下：

```
POST /index.php?id=1&index=2 HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 18
authorization: kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgb
WEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9j
YWxsYmFja19wdWJfa2V5X3YxLnBlbQ==
bucket=yonghu-test
```

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

- 验证签名

  验证签名的过程即为签名的逆过程，由应用服务器验证，过程如下：

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

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

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

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

      **注意:**

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

2. 获取 base64 解码后的签名

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

3. 获取待签名字符串，方法与签名一致

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

4. 验证签名

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

以上例为例：

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

- 应用服务器示例

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

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

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

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

其它语言的服务端代码如下：

Java 版本：

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

PHP 版本：

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

Python 版本：

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

C# 版本：

- 下载地址：点击这里
- 运行方法：解压后参看 `README.md`。

.NET 版本：

- 下载地址：点击这里
- 运行方法：解压后参看 `README.md`。

Go 版本：

- 下载地址：点击这里
- 运行方法：解压后参看 `README.md`。

Ruby 版本：

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

步骤5：返回回调结果

应用服务器返回响应给OSS。

返回的回调请求为：

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

注意:

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

步骤6：返回上传结果

OSS将应用服务器返回的内容返回给用户。

返回的内容响应为：

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

注意:

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

6.13 PutSymlink

PutSymlink可用于针对OSS上的TargetObject创建软链接，用户可以通过该软链接访问TargetObject。

请求语法

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

请求Header

名称	类型	描述
x-oss-symlink-target	字符串	软链接指向的目标文件。 合法值：命名规范同Object。
x-oss-storage-class	字符串	指定Object的存储类型。 取值： • Standard • IA • Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。 **注意:** • 如果StorageClass的值不合法，返回400 错误。错误码：InvalidArgument。 • PutObjectSymlink的存储类型建议不要指定为IA或Archive类型（因为IA与Archive类型的单个文件如不足64KB，会按64KB计量计费）。 • 对于任意存储类型Bucket，若上传Object时指定该值，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。

细节分析

- TargetObjectName同ObjectName一样，需要URL encode。

- 软链接的目标文件类型不能为软链接。
- 创建软链接：
  - 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。
- 试图添加的文件已经存在，并且有访问权限。新添加的文件将覆盖原来的文件，成功将返回200 OK。
- PutSymlink时，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8KB。

示例

请求示例：

```
PUT /link-to-oss.jpg?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
Content-Disposition: attachment;filename=oss_download.jpg
Date: Tue, 08 Nov 2016 02:00:25 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PR
rk= x-oss-symlink-target: oss.jpg
x-oss-storage-class: Standard
```

返回示例：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 08 Nov 2016 02:00:25 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 582131B9109F4EE66CDE56A5
ETag: "0A477B89B4602AA8DECB8E19BFD447B6"
```

6.14 GetSymlink

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

细节分析

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:UNQDb7GapEgJCZkcde6O
hZ9Jfe8=
```

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"
```

6.15 RestoreObject

RestoreObject接口用于服务端执行解冻任务。

RestoreObject操作只针对归档类型（Archive）的Object，不适用于标准类型（Standard）和低频访问类型（IA）的Object。

归档类型的Object在执行Restore前后的状态变换过程如下：

1. 一个归档类型的Object初始时处于冷冻状态。
2. 提交一次Restore请求后，Object将处于解冻中的状态，服务端执行解冻。
3. 服务端完成解冻任务后，Object进入解冻状态，此时用户可以读取Object。
4. 解冻状态默认持续24小时，24小时内再次调用RestoreObject接口则解冻状态会自动延长24小时，最多可延长7天。解冻状态结束后，Object回到初始时的冷冻状态。

状态变换过程中产生的相关费用如下：

- 对一个处于冷冻状态的Object执行Restore操作，会产生数据取回费用。
- 解冻状态最多延长7天。在此期间不再重复收取数据取回费用。
- 解冻状态结束后，Object又回到冷冻状态，再次执行Restore操作会收取数据取回费用。

请求语法

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

细节分析

- 如果Object不存在，则返回404。
- 如果针对非归档类型的Object发送Restore请求，则返回400错误。错误码：OperationNotSupported。
- 如果针对该Object第一次调用**RestoreObject**接口，则返回202。
- 如果已经成功调用过**RestoreObject**接口，但Object未完成解冻，再次调用时返回409错误，错误码为：RestoreAlreadyInProgress，代表服务端正在执行Restore操作，用户只需要等待作业完成，最长等待时间为4小时。
- 如果已经成功调用过**RestoreObject**接口，且Object已完成解冻，再次调用时返回200 OK，且会将Object的可下载时间延长1天，最多延长7天。

示例

- 首次提交Restore请求，Object处于冷冻状态。

  请求示例：

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

- 提交Restore请求，Object处于解冻中的状态。

请求示例：

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

- 提交Restore请求，Object处于解冻状态。

  请求示例：

  ```
  POST /oss.jpg?restore HTTP/1.1
  Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
  Date: Sat, 15 Apr 2017 07:45:29 GMT
  Authorization: OSS e1Unnbm1rgdnpI:u6O6FMJnn+WuBwbByZxm1+y4eyu+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
  ```

6.16 SelectObject

功能介绍

对象存储（Object Storage Service，简称OSS） 是基于阿里云飞天分布式系统的海量、安全和高可靠的云存储服务，是一种面向互联网的大规模、低成本、通用存储，提供RESTful API，具备容量和处理的弹性扩展能力。OSS不仅非常适合存储海量的媒体文件，也适合作为数据仓库存储海量的数据文件。目前Hadoop 3.0已经支持OSS，在EMR上运行Spark/Hive/Presto等服务以及阿里自研的MaxCompute、HybridDB以及新上线的Data Lake Analytics都支持从OSS直接处理数据。

然而，目前OSS提供的GetObject接口决定了大数据平台只能把OSS数据全部下载到本地然后进行分析过滤，在很多查询场景下浪费了大量带宽和客户端资源。

SelectObject接口是对上述问题的解决方案。其核心思想是大数据平台将条件、Projection下推到OSS层，让OSS做基本的过滤，从而只返回有用的数据。客户端一方面可以减少网络带宽，另一方面也减少了数据的处理量，从而节省了CPU和内存用来做其他更多的事情。这使得基于OSS的数据仓库、数据分析成为一种更有吸引力的选择。

SelectObject提供了Java和Python 的SDK。目前支持RFC 4180标准的CSV（包括TSV等类CSV文件，文件的行列分隔符以及Quote字符都可自定义），且文件编码为UTF-8。支持加密文件（OSS完全托管、KMS托管主密钥）。

支持的SQL语法如下：

- SQL 语句： Select From Where
- 数据类型：String, int(64bit), float(64bit), decimal(128) , timestamp, Boolean
- 操作： 逻辑条件（AND,OR,NOT)， 算术表达式（+-*/%)， 比较操作(>,=, <, >=, <=, !=)，String 操作 (LIKE, ||)

和GetObject提供了基于Byte的分片下载类似，SelectObject也提供了分片查询的机制，包括两种分片方式：按行分片和按Split分片。按行分片是常用的分片方式，然而对于稀疏数据来说，按行分片可能会导致分片时负载不均衡。Split是OSS用于分片的一个概念，一个Split包含多行数据，每个Split的数据大小大致相等，相对按行来，按Spit是更加高效的分片方式。尤其是对于CSV数据来说，基于Byte的分片可能会将数据破坏，因此按Spit分片更加合适。

关于数据类型，OSS中的CSV数据默认都是String类型，用户可以使用CAST函数实现数据转换，比如下面的SQL查询将_1和_2转换为int后进行比较。

```
Select * from OSSOBject where cast (_1 as int) > cast(_2 as int)
```

同时，对于SelectObject支持在Where条件中进行隐式的转换，比如下面的语句中第一列和第二列将被转换成int：

```
Select _1 from ossobject where _1 + _2 > 100
```

SelectObject使用说明

对目标CSV文件执行SQL语句，返回执行结果。同时该命令会自动保存CSV文件的metadata信息，比如总的行数和列数等。

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

- 请求语法

```
POST /object?x-oss-process=csv/select HTTP/1.1
HOST: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: time GMT
Content-Length: ContentLength
Content-MD5: MD5Value
Authorization: Signature

<?xml  version="1.0"  encoding="UTF-8"?>
<SelectRequest>
 <Expression>base64 encode(Select * from OSSObject where ...)</
Expression>
 <InputSerialization>
  <CompressionType>None|GZIP</CompressionType>
  <CSV>
   <FileHeaderInfo>NONE|IGNORE|USE</FileHeaderInfo>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>
   <QuoteCharacter>base64 encode</QuoteCharacter>
   <CommentCharacter>base64 encode</CommentCharacter>
   <Range>line-range=start-end|split-range=start-end</Range>
  </CSV>
  </InputSerialization>
  <OutputSerialization>
                <CSV>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>
  </CSV>
                <KeepAllColumns>false|true</KeepAllColumns>
 <OutputRawData>false|true</OutputRawData>
     <EnablePayloadCrc>true</EnablePayloadCrc>
  <OutputHeader>false</OutputHeader>
  </OutputSerialization>
  <Options>
 <SkipPartialDataRecord>false</SkipPartialDataRecord>
  </Options>
</SelectRequest>
```

- 请求元素

名称	类型	描述
SelectRequest	容器	保存Select请求的容器 子节点：Expression , InputSerialization， OutputSerialization 父节点：None
Expression	字符串	以Base64 编码的SQL语句 子节点：None 父节点：SelectRequest

名称	类型	描述
InputSerialization	容器	输入序列化参数（可选） 子节点:CompressionType, CSV 父节点：SelectRequest
OutputSerialization	容器	输出序列化参数（可选） 子节点：CSV，OutputRawData 父节点：SelectRequest
CSV（InputSerialization）	容器	输入CSV的格式参数（可选） 子节点：FileHeaderInfo、RecordDelimiter、FieldDelimiter、QuoteCharacter、CommentCharacter、 Range 父节点：InputSerialization
CSV(OutputSerialization)	容器	输出CSV的格式参数（可选） 子节点： RecordDelimiter、FieldDelimiter 父节点：OutputSerialization
OutputRawData	bool，默认false	指定输出数据为纯数据（不是下面提到的基于Frame格式）（可选） 子节点：None 父节点：OutputSerialization
CompressionType	枚举	指定文件压缩类型：None|GZIP 子节点：None 父节点：InputSerialization
FileHeaderInfo	枚举	指定CSV文件头信息（可选） 取值： 子节点：None 父节点：CSV（输入）

名称	类型	描述
RecordDelimiter	字符串	指定CSV换行符，以Base64编码。默认值为\n（可选）。未编码前的值最多为两个字符，以字符的ANSI值表示，比如在Java里用\n表示换行。 子节点：None 父节点：CSV（输入、输出）
FieldDelimiter	字符串	指定CSV列分隔符，以Base64编码。默认值为，（可选） 未编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用，表示逗号。 子节点：None 父节点：CSV（输入，输出）
QuoteCharacter	字符串	指定CSV的引号字符，以Base64编码。默认值为\”（可选）。在CSV中引号内的换行符，列分隔符将被视作普通字符。为编码前的值必须为一个字符，以字符的ANSI值表示，比如Java里用\”表示引号。 子节点：None 父节点：CSV（输入）
CommentCharacter	字符串	指定CSV的注释符，以Base464编码。默认值为#（可选）
Range	字符串	指定查询文件的范围（可选）。支持两种格式： 其中start和end均为inclusive。其格式和range get中的range参数一致。 子节点：None 父节点：CSV（输入）

名称	类型	描述
KeepAllColumns	bool	指定返回结果中包含CSV所有列的位置（可选，默认值为false）。但仅仅在select 语句里出现的列会有值，不出现的列则为空，返回结果中每一行的数据按照CSV列的顺序从低到高排列。比如下面语句：`select _5, _1 from ossobject.` 如果KeepAllColumn = true，假设一共有6列数据，则返回的数据如下： Value of 1st column,,,,Value of 5th column,\n 子节点：None 父节点：OutputSerialization
EnablePayloadCrc	bool	在每个Frame中会有一个32位的crc32校验值。客户端可以计算相应payload的Crc32值进行数据完整性校验。 子节点：None 父节点：OutputSerialization
Options	容器	额外的可选参数 类型：容器 子节点：SkipPartialDataRecord 父节点：SelectRequest
OutputHeader	bool	在返回结果开头输出CSV头信息。 类型：bool，默认 false。 子节点：None 父节点：OutputSerialization

名称	类型	描述
SkipPartialDataRecord	bool	忽略缺失数据的行。当该参数为true时 ，OSS会忽略缺失某些列的行而不报错。当该参数为false时，CSV文件中有任何一行数据不完整时，OSS会报错并停止处理。 类型：bool，默认 false。 子节点:None 父节点：Options

- 返回结果

  请求结果以一个个Frame形式返回。每个Frame的格式如下,其中checksum均为CRC32：

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

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

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

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

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

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

- 样例请求

```
POST /oss-select/bigcsv_normal.csv?x-oss-process=csv%2Fselect HTTP/1
.1
Date: Fri, 25 May 2018 22:11:39 GMT
Content-Type:
Authorization: OSS LTAIJPXxMLocA0fD:FC/9JRbBGRw4o2QqdaL246Pxuvk=
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.0.
30319.42000)
Content-Length: 748
Expect: 100-continue
Connection: keep-alive
Host: host name

<?xml version="1.0"?>
<SelectRequest>
 <Expression>c2VsZWN0IGNvdW50KCopIGZyb20gb3Nzb2JqZWN0IHdoZXJlIF
80ID4gNDU=
 </Expression>
 <InputSerialization>
  <Compression>None</Compression>
```

```
  <CSV>
   <FileHeaderInfo>Ignore</FileHeaderInfo>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
   <Comments>Iw==</Comments>
  </CSV>
 </InputSerialization>
 <OutputSerialization>
  <CSV>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
         </CSV>
                <KeepAllColumns>false</KeepAllColumns>
         <OutputRawData>false</OutputRawData>
        </OutputSerialization>
</SelectRequest>
```

- SQL语句正则表达式

  ```
  SELECT select-list from OSSObject where_opt limit_opt
  ```

  其中SELECT, OSSOBJECT以及 WHERE为关键字不得更改。

  ```
  select_list: column name

  | column index (比如_1, _2)

  | function(column index | column name)

  | select_list AS alias
  ```

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

  ```
  Where_opt:
  | WHERE expr
  expr:
  | literal value
  | column name
  | column index
  | expr op expr
  | expr OR expr
  | expr AND expr
  | expr IS NULL
  | expr IS NOT NULL
  | expr IN (value1, value2,….)
  | expr NOT in (value1, value2,…)
  | expr between value1 and value2
  | NOT (expr)
  | expr op expr
  | (expr)
  | cast (column index or column name or literal as INT|DOUBLE|
  DATETIME)
  ```

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

cast: 对于同一个column，只能cast成一种类型。

`limit_opt:`

| limit 整数

聚合和Limit的混用

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

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

- SQL语句限制
  - 目前仅仅支持UTF-8编码的文本文件以及GZIP压缩过的UTF-8文本。GZIP文件不支持deflate格式。
  - 仅支持单文件查询，不支持join, order by, group by, having
  - Where语句里不能包含聚合条件(e.g. where max(cast(age as int)) > 100这个是不允许的)。
  - 支持的最大的列数是1000，SQL中最大的列名称为1024。
  - 在LIKE语句中，支持最多5个%通配符。*和%是等价的，表示0或多个任意字符。
  - 在IN语句中，最多支持1024个常量项。
  - Select后的Projection可以是列名，列索引(_1, _2等)，或者是聚合函数，或者是CAST函数；不支持其他表达式。 比如select _1 + _2 from ossobject是不允许的。
  - 支持的最大行及最大列长度是都是256K。
  - SQL最大长度16K，where后面表达式个数最多20个，表达式深度最多10层，聚合操作最多100个。

CreateSelectObjectMeta

CreateSelectObjectMeta用于获取目标CSV文件的总的行数，总的列个数，以及Splits个数。如果该信息不存在，则会扫描整个文件，分析并记录下CSV文件的上述信息。如果该API执行正确，返回200。否则如果目标CSV文件为非法、或者指定的分隔符和目标CSV不匹配，则返回400。

- 请求语法

```
POST  /samplecsv?x-oss-process=csv/meta


<CsvMetaRequest>
 <InputSerialization>
  <CompressionType>None</CompressionType>
```

```
  <CSV>
   <RecordDelimiter>base64 encode</RecordDelimiter>
   <FieldDelimiter>base64 encode</FieldDelimiter>
   <QuoteCharacter>base64 encode</QuoteCharacter>
  </CSV>
 </InputSerialization>
 <OverwriteIfExists>false|true</OverwriteIfExists>
</CsvMetaRequest>
```

- 请求元素

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

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

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

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

Response Header：无专门header。

- 样例请求

```
POST /oss-select/bigcsv_normal.csv?x-oss-process=csv%2Fmeta HTTP/1.1
Date: Fri, 25 May 2018 23:06:41 GMT
Content-Type:
Authorization: OSS LTAIJPXxMLocA0fD:2WF2l6zozf+hzTj9OSXPDklQCvE=
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.0.
30319.42000)
Content-Length: 309
Expect: 100-continue
Connection: keep-alive
Host: Host

<?xml version="1.0"?>
<CsvMetaRequest>
 <InputSerialization>
  <CSV>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
  </CSV>
 </InputSerialization>
 <OverwriteIfExisting>false</OverwriteIfExisting>
</CsvMetaRequest>
```

- 返回响应

```
HTTP/1.1 200 OK
```

```
Server: AliyunOSS
Date: Fri, 25 May 2018 23:06:42 GMT
Content-Type: application/vnd.ms-excel
Content-Length: 0
Connection: close
x-oss-request-id: 5B089702461FB4C07B000C75
x-oss-location: oss-cn-hangzhou-a
x-oss-access-id: LTAIJPXxMLocA0fD
x-oss-sign-type: NormalSign
x-oss-object-name: bigcsv_normal.csv
Accept-Ranges: bytes
ETag: "3E1372A912B4BC86E8A51234AEC0CA0C-400"
Last-Modified: Wed, 09 May 2018 00:22:32 GMT
x-oss-object-type: Multipart
x-oss-bucket-storage-type: standard
x-oss-hash-crc64ecma: 741622077104416154
x-oss-storage-class: Standard
**x-oss-select-csv-rows: 54000049**
**x-oss-select-csv-columns: 4**
**x-oss-select-csv-splits: 960**
```

Python SDK 样例

```
import os
import oss2

def  select_call_back(consumed_bytes, total_bytes =  None):
 print('Consumed Bytes:'  +  str(consumed_bytes) +  '\n')

# 首先初始化AccessKeyId、AccessKeySecret、Endpoint等信息。
# 通过环境变量获取，或者把诸如“<yourAccessKeyId>”替换成真实的AccessKeyId等。
#
# 以杭州区域为例，Endpoint可以是：
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com

access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<yourAccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<
yourAccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<yourBucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<yourEndpoint>')

# 创建存储空间实例，所有文件相关的方法都需要通过存储空间实例来调用。
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
key =  'python_select.csv'
content =  'Tom Hanks,USA,45\r\n'*1024
filename =  'python_select.csv'
# 上传文件
bucket.put_object(key, content)
csv_meta_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n'}
select_csv_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n',
'LineRange': (500, 1000)}

csv_header = bucket.create_select_object_meta(key, csv_meta_params)
print(csv_header.csv_rows)
print(csv_header.csv_splits)
result = bucket.select_object(key, "select * from ossobject where _3
 > 44 limit 100000", select_call_back, select_csv_params)
```

```
content_got =  b''
for chunk in result:
 content_got += chunk
print(content_got)

result = bucket.select_object_to_file(key, filename,
"select * from ossobject where _3 > 44 limit 100000", select_call_back
, select_csv_params)

bucket.delete_object(key)
```

Java SDK 样例

```
package samples;

import com.aliyun.oss.event.ProgressEvent;
import com.aliyun.oss.event.ProgressListener;
import com.aliyun.oss.model.*;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;

import java.io.BufferedOutputStream;
import java.io.ByteArrayInputStream;
import java.io.FileOutputStream;

/**
 * Examples of create select object metadata and select object.
 *
 */
public class SelectObjectSample {
    private static String endpoint = "<endpoint, http://oss-cn-
hangzhou.aliyuncs.com>";
    private static String accessKeyId = "<accessKeyId>";
    private static String accessKeySecret = "<accessKeySecret>";
    private static String bucketName = "<bucketName>";
    private static String key = "<objectKey>";

    public static void main(String[] args) throws Exception {
        // OSSClientBuilder 只针对Java SDK 3.0及以上版本。
        OSS client = new OSSClientBuilder().build(endpoint, accessKeyI
d, accessKeySecret);
        String content = "name,school,company,age\r\n" +
                "Lora Francis,School A,Staples Inc,27\r\n" +
                "Eleanor Little,School B,\"Conectiv, Inc\",43\r\n" +
                "Rosie Hughes,School C,Western Gas Resources Inc,44\r\
n" +
                "Lawrence Ross,School D,MetLife Inc.,24";

        client.putObject(bucketName, key, new ByteArrayInputStream(
content.getBytes()));

        SelectObjectMetadata selectObjectMetadata = client.createSele
ctObjectMetadata(
                new CreateSelectObjectMetadataRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInp
utFormat(
                                        new CSVFormat().withHeaderInfo
(CSVFormat.Header.Use).withRecordDelimiter("\r\n"))));
        System.out.println(selectObjectMetadata.getCsvObjectMetadata
().getTotalLines());
```

```
        System.out.println(selectObjectMetadata.getCsvObjectMetadata
().getSplits());

        SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInp
utFormat(
                                        new CSVFormat().withHeaderInfo
(CSVFormat.Header.Use).withRecordDelimiter("\r\n")))
                        .withOutputSerialization(new OutputSeri
alization().withCsvOutputFormat(new CSVFormat()));
        selectObjectRequest.setExpression("select * from ossobject
where _4 > 40");
        OSSObject ossObject = client.selectObject(selectObjectRequest
);
        // read object content from ossObject
        BufferedOutputStream outputStream = new BufferedOutputStream(
new FileOutputStream("result.data"));
        byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = ossObject.getObjectContent().read(buffer
)) != -1) {
            outputStream.write(buffer, 0, bytesRead);
        }
        outputStream.close();
    }
}
```

常见的SQL用例

应用场景	SQL语句
返回前10行数据	select * from ossobject limit 10
返回第1列和第3列的整数，并且第1列大于第3列	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)
返回第1列以'陈'开头的记录的个数(注：此处like后的中文需要用UTF-8编码)	select count(*) from ossobject where _1 like '陈%'
返回所有第2列时间大于2018-08-09 11:30:25且第3列大于200的记录	select * from ossobject where _2 > cast('2018-08-09 11:30:25' as timestamp) and _3 > 200
返回第2列浮点数的平均值，总和，最大值，最小值	select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))
返回第1列和第3列连接的字符串中以'Tom'为开头以’Anderson‘结尾的所有记录	select * from ossobject where (_1 || _3) like 'Tom%Anderson'
返回第1列能被3整除的所有记录	select * from ossobject where (_1 % 3) == 0
返回第1列大小在1995到2012之间的所有记录	select * from ossobject where _1 between 1995 and 2012

应用场景	SQL语句
返回第5列值为N,M,G,L的所有记录	select * from ossobject where _5 in ('N', 'M', 'G', 'L')
返回第2列乘以第3列比第5列大100以上的所有记录	select * from ossobject where _2 * _3 > _5 + 100

支持的时间格式

您无需指定日期格式，SelectObject可以将以下格式的字符串自动转换成timestamp类型。比如cast-('20121201' as timestamp)会被解析成2012年12月1日。

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

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

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

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

格式	说明
MM-DD-YYYY	月-日-年
MM.DD.YYYY	月.日.年

最佳实践

当一个文件很大时，要有效实现分片查询，推荐的流程如下：

1. 调用Create Select Object Meta API获得该文件的总的Split数。理想情况下如果该文件需要用SelectObject，则该API最好在查询前进行异步调用，这样可以节省扫描时间。
2. 根据客户端资源情况选择合适的并发度n，用总的Split数除以并发度n得到每个分片查询应该包含的Split个数。
3. 在请求Body中用诸如split-range=1-20的形式进行分片查询。
4. 如果需要最后可以合并结果。

SelectObject和Normal类型文件配合性能更佳。Multipart 以及Appendable类型的文件由于其内部结构差异导致性能较差。

7 关于MultipartUpload的操作

7.1 简介

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

7.2 InitiateMultipartUpload

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

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

请求Header

名称	类型	描述
Cache-Control	字符串	指定该Object被下载时的网页的缓存行为；更详细描述请参照*RFC2616*。 默认值：无
Content-Disposition	字符串	指定该Object被下载时的名称；更详细描述请参照*RFC2616*。 默认值：无
Content-Encoding	字符串	指定该Object被下载时的内容编码格式；更详细描述请参照*RFC2616*。 默认值：无
Expires	整数	过期时间（milliseconds）；更详细描述请参照*RFC2616*。 默认值：无
x-oss-server-side-encryption	字符串	指定上传该Object每个part时使用的服务器端加密编码算法，OSS会对上传的每个part采用服务器端加密编码进行存储。 合法值：AES256 或 KMS 注意：用户需要在控制台开通KMS（密钥管理服务），才可使用KMS加密算法，否则会报KmsServiceNotEnabled错误码。
x-oss-server-side-encryption-key-id	字符串	表示KMS托管的用户主密钥。 该参数在**x-oss-server-side-encryption**为KMS时有效。

名称	类型	描述
x-oss-storage-class	字符串	指定Object的存储类型。 取值： • Standard • IA • Archive 支持的接口：PutObject、InitMultipartUpload、AppendObject、 PutObjectSymlink、CopyObject。 **注意:** • 如果StorageClass的值不合法，返回400 错误。错误码：InvalidArgument。 • 对于任意存储类型Bucket，若上传Object时指定该值，则此次上传的Object将存储为指定的类型。例如，在IA类型的Bucket中上传Object时，若指定x-oss-storage-class为Standard，则该Object直接存储为Standard。

响应元素(Response Elements)

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

细节分析

- 该操作计算认证签名的时候，需要加“?uploads”到CanonicalizedResource中。
- 初始化Multipart Upload请求，支持如下标准的HTTP请求头：Cache-Control、Content-Disposition、Content-Encoding、Content-Type、Expires，以及以`x-oss-meta-`开头的用户自定义Headers。具体含义请参见*PutObject*。
- 初始化Multipart Upload请求，并不会影响已经存在的同名Object。
- 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。
- 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AES256和KMS加密，指定其他值会返回400 错误和错误码：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 错误和错误码：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
x-oss-storage-class: Archive
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:/cluRFtRwMTZpC2hTj4F6
7AGdM4=
```

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>
```

7.3 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。分片数量范围1-10000，单个分片大小范围100KB-5GB，最后一片可以小于100KB。

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

- 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。
- 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。
- 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。
- 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。
- 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/lICfXEvPmmSW86bBAfMm
UmWjI=
[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
```

7.4 UploadPartCopy

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

通过在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	类型	描述
x-oss-copy-source	字符串	复制源地址（必须有可读权限） 默认值：无

名称	类型	描述
x-oss-copy-source-range	整型	源Object的拷贝范围。如，设定 bytes=0-9，表示传送第0到第9这10个字符。 当拷贝整个源Object时不需要该请求Header。 默认值：无

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

名称	类型	描述
x-oss-copy-source-if-match	字符串	如果源Object的ETAG值和用户提供的ETAG相等，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。 默认值：无
x-oss-copy-source-if-none-match	字符串	如果传入的ETag值和Object的ETag不匹配，则正常传输文件，并返回200 OK；否则返回304 Not Modified 默认值：无
x-oss-copy-source-if-unmodified-since	字符串	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误。 默认值：无
x-oss-copy-source-if-modified-since	字符串	如果指定的时间早于实际修改时间，则正常传送文件，并返回200 OK；否则返回304 not modified 默认值：无 时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT

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/lICfXEvPmmSW86bBAfMmUmWjI=
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>
```

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

请求元素(Request Elements)

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

响应元素(Response Elements)

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

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

细节分析

- 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。
- 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。
- 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。
- OSS处理Complete Multipart Upload请求成功后，该Upload ID就会变成无效。
- 同一个Object可以同时拥有不同的Upload Id，当Complete一个Upload ID后，该Object的其他Upload ID不受影响。
- 若调用Initiate Multipart Upload接口时，指定了x-oss-server-side-encryption请求头，则在Complete Multipart Upload的响应头中返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
- 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 qn6qrrqxo2oawuk53otfjbyc:8VwFhFUWmVecK6jQlHlXMK/
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>
```

7.6 AbortMultipartUpload

AbortMultipartUpload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

- 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。
- 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/lICfXEvPmmSW86bBAfMm
UmWjI=
```

返回示例:

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

7.7 ListMultipartUploads

ListMultipartUploads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	类型	描述
delimiter	字符串	是一个用于对Object名字进行分组的字符。所有名字包含指定的前缀且第一次出现delimiter字符之间的object作为一组元素——CommonPrefixes。
max-uploads	字符串	限定此次返回Multipart Uploads事件的最大数目，如果不设定，默认为1000，max-uploads取值不能大于1000。
key-marker	字符串	与upload-id-marker参数一同使用来指定返回结果的起始位置。 • 如果upload-id-marker参数未设置，查询结果中包含：所有Object名字的字典序大于key-marker参数值的Multipart事件。 • 如果upload-id-marker参数被设置，查询结果中包含：所有Object名字的字典序大于key-marker参数值的Multipart事件和Object名字等于key-marker参数值，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。
prefix	字符串	限定返回的object key必须以prefix作为前缀。注意使用prefix查询时，返回的key中仍会包含prefix。

名称	类型	描述
upload-id-marker	字符串	与key-marker参数一同使用来指定返回结果的起始位置。 • 如果key-marker参数未设置，则OSS忽略upload-id-marker参数。 • 如果key-marker参数被设置，查询结果中包含：所有Object名字的字典序大于key-marker参数值的Multipart事件和Object名字等于key-marker参数值，但是Upload ID比upload-id-marker参数值大的Multipart Uploads事件。
encoding-type	字符串	指定对返回的内容进行编码，指定编码的类型。Delimiter、KeyMarker、Prefix、NextKe 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、NextKeyM 行编码。 默认值：无

响应元素(Response Elements)

名称	类型	描述
ListMultipartUploadsResult	容器	保存List Multipart Upload请求结果的容器。 子节点：Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MasUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload 父节点：None
Bucket	字符串	Bucket名称。 父节点：ListMultipartUploadsResult
EncodingType	字符串	指明返回结果中编码使用的类型。如果请求的参数中指定了encoding-type，那返回的结果会对Delimiter、KeyMarker、Prefix、NextKeyM些元素进行编码。 父节点：ListMultipartUploadsResult
KeyMarker	字符串	列表的起始Object位置。 父节点：ListMultipartUploadsResult
UploadIdMarker	字符串	列表的起始UploadID位置。 父节点：ListMultipartUploadsResult
NextKeyMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextKeyMarker元素，用于标明接下来请求的KeyMarker值。 父节点：ListMultipartUploadsResult

名称	类型	描述
NextUploadMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextUploadMarker元素，用于标明接下来请求的UploadMarker值。 父节点：ListMultipartUploadsResult
MaxUploads	整数	返回的最大Upload数目。 父节点：ListMultipartUploadsResult
IsTruncated	枚举字符串	标明是否本次返回的MultipartUpload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。 有效值`false`、`true` 默认值：`false` 父节点：ListMultipartUploadsResult
Upload	容器	保存Multipart Upload事件信息的容器。 子节点：Key, UploadId, Initiated 父节点：ListMultipartUploadsResult
Key	字符串	初始化Multipart Upload事件的Object名字。 父节点：Upload
UploadId	字符串	Multipart Upload事件的ID。 父节点：Upload
Initiated	日期	Multipart Upload事件初始化的时间。 父节点：Upload

细节分析

- **max-uploads**参数最大值为1000。

- 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。
- 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）。
- 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+dcivn7kwBM
vOY=
```

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</NextUpload
IdMarker>
    <Delimiter></Delimiter>
    <Prefix></Prefix>
    <MaxUploads>1000</MaxUploads>
    <IsTruncated>false</IsTruncated>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0004B999EF518A1FE585B0C9360DC4C8</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>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>
```

7.8 ListParts

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

请求语法

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

请求参数**(Request Parameters)**

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

响应元素(Response Elements)

名称	类型	描述
ListPartsResult	容器	保存List Part请求结果的容器。 子节点：Bucket, Key, UploadId, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part 父节点：无
Bucket	字符串	Bucket名称。 父节点：ListPartsResult
EncodingType	字符串	指明对返回结果进行编码使用的类型。如果请求的参数中指定了encoding-type，那会对返回结果中的Key进行编码。 父节点：ListPartsResult
Key	字符串	Object名称。 父节点：ListPartsResult
UploadId	字符串	Upload事件ID。 父节点：ListPartsResult
PartNumberMarker	整数	本次List结果的Part Number起始位置。 父节点：ListPartsResult
NextPartNumberMarker	整数	如果本次没有返回全部结果，响应请求中将包含NextPartNumberMarker元素，用于标明接下来请求的PartNumberMarker值。 父节点：ListPartsResult
MaxParts	整数	返回请求中最大的Part数目。 父节点：ListPartsResult
IsTruncated	枚举字符串	标明是否本次返回的List Part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。 有效值：`true`、`false` 父节点：ListPartsResult

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

细节分析

- List Parts支持max-parts和part-number-marker两种请求参数。
- max-parts参数最大值为1000；默认值也为1000。
- 在OSS的返回结果按照Part号码升序排列。
- 由于网络传输可能出错，所以不推荐用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:4qOnUMc9UQWqkz8wDqD3
lIsa9P8=
```

返回示例：

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

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

```
    <Bucket>multipart_upload</Bucket>
    <Key>multipart.data</Key>
    <UploadId>0004B999EF5A239BB9138C6227D69F95</UploadId>
    <NextPartNumberMarker>5</NextPartNumberMarker>
    <MaxParts>1000</MaxParts>
    <IsTruncated>false</IsTruncated>
    <Part>
        <PartNumber>1</PartNumber>
        <LastModified>2012-02-23T07:01:34.000Z</LastModified>
        <ETag>&quot;3349DC700140D7F86A078484278075A9&quot;</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>2</PartNumber>
        <LastModified>2012-02-23T07:01:12.000Z</LastModified>
        <ETag>&quot;3349DC700140D7F86A078484278075A9&quot;</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <LastModified>2012-02-23T07:02:03.000Z</LastModified>
        <ETag>&quot;7265F4D211B56873A381D321F586E4A9&quot;</ETag>
        <Size>1024</Size>
    </Part>
</ListPartsResult>
```

8 关于LiveChannel的操作

8.1 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。

8.2 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。注意：生成的m3u8名称仍然会被添加”${channel_name}/“前缀。

推流地址的签名规则

一个带签名的推流地址形如：`rtmp://${bucket}.${host}/live/${channel}?OSSAccessKeyId=xxx&Expires=yyy&Signature=zzz&${params}`

参数名称	描述
OSSAccessKeyId	意义同OSS的HTTP签名的AccessKeyId
Expires	过期时间戳，格式采用Unit timestamp
Signature	签名字符串，后文会描述其计算方法
params	其他参数，所有的参数都需要参与签名

Signature的计算规则如下：

```
base64(hmac-sha1(AccessKeySecret,
    + Expires + "\n"
    + CanonicalizedParams
    + CanonicalizedResource))
```

名称	描述
CanonicalizedResource	格式为 “/BucketName/ChannelName”
CanonicalizedParams	按照param key字典序拼接”key:value\n”,将所有的参数拼起来，如果参数个数为0，那么这一项为空。参数中不包含SecurityToken、OSSAccessKeyId和Expire以及Signature。每一个param key只能出现一次。

8.3 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。 有效值：enabled、disabled	是

细节分析

- 当没有客户端向该LiveChannel推流时，调用PutLiveChannel重新创建LiveChannel也可以达到修改Status的目的。
- 当有客户端向该LiveChannel推流时，无法调用PutLiveChannel重新创建LiveChannel，只能通过本接口修改LiveChannel的状态为disabled。

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
```

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

名称	类型	描述	是否必需
FragDuration	字符串	当Type为HLS时，指定每个ts文件的时长（单位：秒）。 子节点：无 父节点: Target 默认值：5 取值范围：[1, 100]	否
FragCount	字符串	当Type为HLS时，指定m3u8文件中包含ts文件的个数。 子节点：无 父节点：Target 默认值: 3 取值范围：[1, 100]	否
PlaylistName	字符串	当Type为HLS时，指定生成的m3u8文件的名称，必须以”.m3u8”结尾，长度范围为[6, 128]。 子节点：无 父节点：Target 默认值：playlist.m3u8 取值范围：[6, 128]	否
Snapshot	容器	保存高频截图操作Snapshot 选项的容器。 子节点：RoleName，DestBucket，NotifyTopic，Interval，PornRec 父节点：Snapshot	否
RoleName	字符串	用于高频截图操作的角色名称，要求有DestBucket的写权限和向NotifyTopic发消息的权限。 子节点：无 父节点：Snapshot	否
DestBucket	字符串	保存高频截图目标Bucket，要求与当前Bucket是同一个Owner。 子节点：无 父节点：Snapshot	否

名称	类型	描述	是否必需
NotifyTopic	字符串	用于通知用户高频截图操作结果的MNS的Topic。 子节点：无 父节点：Snapshot	否
Interval	数字	高频截图的间隔长度（单位：秒）。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。 子节点：无 父节点：Snapshot 取值范围：[1, 100]	否

细节分析

- ChannelName必须符合ObjectName的命名规范，另外，ChannelName不能包含”/“。
- 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。
- 转储类型为HLS时，OSS会在生成每个ts文件后，更新m3u8文件；m3u8文件中最多包含最近的FragCount个ts文件。
- 转储类型为HLS时，写入当前ts文件的音视频数据时长达到FragDuration指定的时长后，OSS会在收到下一个关键帧的时候切换到下一个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	类型	描述
CreateLiveChannelResult	容器	保存CreateLiveChannel请求结果的容器。 子节点：PublishUrls,PlayUrls 父节点：无
PublishUrls	容器	保存推流地址的容器。 子节点：Url 父节点：CreateLiveChannelResult
Url	字符串	推流地址。 子节点：无 父节点：PublishUrls

名称	类型	描述
PlayUrls	容器	保存推流地址的容器。 子节点：Url 父节点：CreateLiveChannelResult
Url	字符串	播放地址。 子节点：无 父节点：PlayUrls

细节分析

- 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。
- 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>
```

8.5 GetVodPlaylist

GetVodPlaylist接口用来查询指定 LiveChannel 推流生成的，指定时间段内的播放列表。

请求语法

```
GET /ChannelName?vod&endTime=EndTime&startTime=StartTime HTTP/1.1
Date: GMT date
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue
```

请求参数

名称	描述	是否必需
ChannelName	指定 LiveChannel 名称，该LiveChannel 必须存在。	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。 **注意:** EndTime 必须大于 StartTime，且时间跨度不能大于 1 天。	是

示例

请求示例

```
GET /test-channel?vod&endTime=1472020226&startTime=1472020031 HTTP/1.1
Date: Thu, 25 Aug 2016 07:13:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:ABIigvnLtCHK+7fMHLeRlOUnzv0=
```

返回示例

```
HTTP/1.1 200
content-length: 312
```

```
server: AliyunOSS
connection: close
etag: "9C6104DD9CF1A0C4D0CFD21F43905D59"
x-oss-request-id: 57BE9A96B92475920B002359
date: Thu, 25 Aug 2016 07:13:26 GMT
Content-Type: application/x-mpegURL

#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:13
#EXTINF:7.120,
1543895706266.ts
#EXTINF:5.840,
1543895706323.ts
#EXTINF:6.400,
1543895706356.ts
#EXTINF:5.520,
1543895706389.ts
#EXTINF:5.240,
1543895706428.ts
#EXTINF:13.320,
1543895706468.ts
#EXTINF:5.960,
1543895706538.ts
#EXTINF:6.520,
1543895706561.ts
#EXT-X-ENDLIST
```

8.6 PostVodPlaylist

PostVodPlaylist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

名称	描述	是否必需
EndTime	指定查询ts文件的终止时间，格式为Unix timestamp。	是

细节分析

- EndTime必须大于StartTime，且时间跨度不能大于1天。
- OSS会查询指定时间范围内的所有该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 YJjHKOKWDWINLKXv:ABIigvnLtCHK+7fMHLeRlOUnzv0=
```

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
```

8.7 GetLiveChannelStat

GetLiveChannelStat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	类型	描述
LiveChannelStat	容器	保存GetLiveChannelStat返回结果的容器。 子节点：Status，ConnectedTime，Video，Audio 父节点：无
Status	枚举字符串	LiveChannel当前的推流状态描述。 子节点：无 父节点：LiveChannelStat 有效值：`Disabled`，`Live`，`Idle`
ConnnectedTime	字符串	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。 子节点：无 父节点：LiveChannelStat
RemoteAddr	字符串	当Status为Live时，表示当前推流客户端的ip地址。 子节点：无 父节点：LiveChannelStat
Video	容器	当Status为Live时，保存视频流信息的容器。 子节点：Width，Heigth，FrameRate，Bandwidth，Codec 父节点：LiveChannelStat
Width	字符串	表示当前视频流的画面宽度（单位：像素）。 子节点：无 父节点：Video
Heigth	字符串	表示当前视频流的画面高度（单位：像素）。 子节点：无 父节点：Video

名称	类型	描述
FrameRate	字符串	表示当前视频流的帧率。 子节点：无 父节点：Video
Bandwidth	字符串	表示当前视频流的码率（单位：B/s）。 子节点：无 父节点：Video
Codec	枚举字符串	表示当前视频流的编码格式。 子节点：无 父节点：Video
Audio	容器	当Status为Live时，保存音频流信息的容器。 子节点：SampleRate，Bandwidth，Codec 父节点：LiveChannelStat
SampleRate	字符串	表示当前音频流的采样率。 子节点：无 父节点：Audio
Bandwidth	字符串	表示当前音频流的码率（单位：B/s）。 子节点：无 父节点：Audio
Codec	枚举字符串	表示当前音频流的编码格式。 子节点：无 父节点：Audio

细节分析

- 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。
- 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>
```

8.8 GetLiveChannelInfo

GetLiveChannelInfo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	类型	描述
LiveChannelConfiguration	容器	保存GetLiveChannelInfo返回结果的容器。 子节点：Description、Status、Target 父节点：无
Description	字符串	LiveChannel的描述信息。 子节点：无 父节点：LiveChannelConfiguration
Status	枚举字符串	LiveChannel的状态信息。 子节点：无 父节点：LiveChannelConfiguration 有效值：`enabled`、`disabled`
Target	容器	保存LiveChannel转储配置的容器。 子节点：Type、FragDuration、FragCount、PlaylistName 父节点：LiveChannelConfiguration
Type	枚举字符串	当Type为HLS时，指定推流时转储文件类型。 子节点：无 父节点：Target 有效值：HLS

名称	类型	描述
FragDuration	字符串	当Type为HLS时，指定每个ts文件的时长（单位：秒）。 子节点：无 父节点: Target
FragCount	字符串	当Type为HLS时，指定m3u8文件中包含ts文件的个数。 子节点：无 父节点：Target
PlaylistName	字符串	当Type为HLS时，指定生成的m3u8文件的名称。 子节点：无 父节点：Target

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 YJjHKOKWDWINLKXv: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>
```

8.9 GetLiveChannelHistory

GetLiveChannelHistory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	类型	描述
LiveChannelHistory	容器	保存GetLiveChannelHistory返回结果的容器。 子节点：LiveRecord 父节点：无
LiveRecord	容器	保存一次推流记录信息的容器。 子节点：StartTime，EndTime，RemoteAddr 父节点：LiveChannelHistory
StartTime	字符串	推流开始时间，使用ISO8601格式表示。 子节点：无 父节点：LiveRecord
EndTime	字符串	推流结束时间，使用ISO8601格式表示。 子节点：无 父节点：LiveRecord
RemoteAddr	字符串	推流客户端的ip地址。 子节点：无 父节点：LiveRecord

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:pqgDBP8JXTXAytBoXpvNoZfo68k=
```

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>
```

8.10 ListLiveChannel

ListLiveChannel接口用于罗列指定的LiveChannel。

请求语法

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

```
Authorization: SignatureValue
```

请求参数

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。 默认值：100	否
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	类型	描述
ListLiveChannelResult	容器	保存ListLiveChannel请求结果的容器。 子节点：Prefix，Marker，MaxKeys，IsTruncated，NextMarker，LiveChannel 父节点：无
Prefix	字符串	本次查询结果的开始前缀。 子节点：无 父节点：ListLiveChannelResult
Marker	字符串	本次ListLiveChannel的起点。 子节点：无 父节点：ListLiveChannelResult
MaxKeys	字符串	响应请求内返回结果的最大数目。 子节点：无 父节点：ListLiveChannelResult

名称	类型	描述
IsTruncated	字符串	指明是否所有的结果都已经返回；“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。 子节点：无 父节点：ListLiveChannelResult
NextMarker	字符串	如果本次没有返回全部结果，响应请求中将包含NextMarker元素，用于标明接下来请求的Marker值。 子节点：无 父节点：ListLiveChannelResult
LiveChannel	容器	保存返回每个LiveChannel信息的容器。 子节点：Name，Description，Status，LastModified，PublishUrls，PlayUrls 父节点：ListLiveChannelResult
Name	字符串	LiveChannel的名称。 子节点：无 父节点：LiveChannel
Description	字符串	LiveChannel的描述。 子节点：无 父节点：LiveChannel
Status	枚举字符串	LiveChannel的状态。 子节点：无 父节点：LiveChannel 有效值：`disabled`，`enabled`
LastModified	字符串	LiveChannel配置的最后修改时间，使用ISO8601格式表示。 子节点：无 父节点：LiveChannel

名称	类型	描述
PublishUrls	容器	保存LiveChannel对应的推流地址的容器。 子节点：Url 父节点：LiveChannel
Url	字符串	LiveChannel对应的推流地址。 子节点：无 父节点：PublishUrls
PlayUrls	容器	保存LiveChannel对应的播放地址的容器。 子节点：Url 父节点：LiveChannel
Url	字符串	LiveChannel对应的播放地址。 子节点：无 父节点：PlayUrls

示例

请求示例

```
GET /?live&max-keys=1 HTTP/1.1
Date: Thu, 25 Aug 2016 07:50:09 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:TaX+tlc/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>
```

8.11 DeleteLiveChannel

DeleteLiveChannel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

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

示例

请求示例

```
DELETE /test-channel?live HTTP/1.1
Date: Thu, 25 Aug 2016 07:32:26 GMT
Host: test-bucket.oss-cn-hangzhou.aliyuncs.com
Authorization: OSS YJjHKOKWDWINLKXv:ZbfvQ3XwmYEE8O9CX8kwVQYNbzQ=
```

返回示例

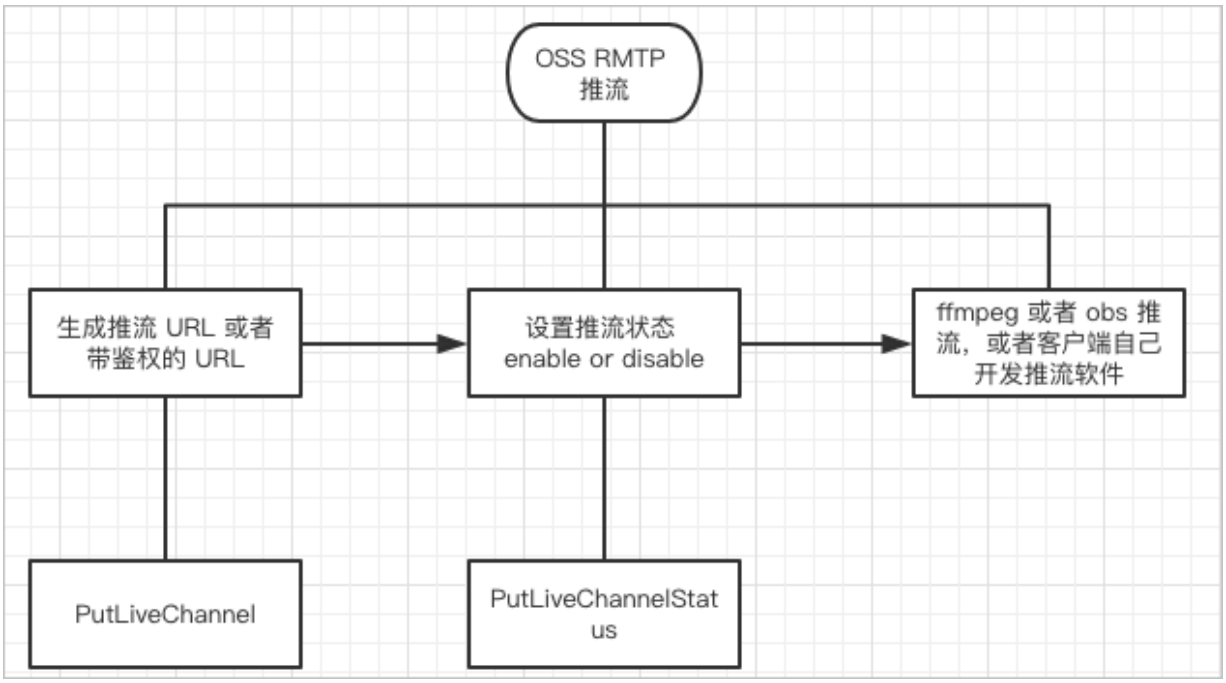
```
HTTP/1.1 204
content-length: 0
server: AliyunOSS
connection: close
x-oss-request-id: 57BE9F0AB92475920B0023E0
date: Thu, 25 Aug 2016 07:32:26 GMT
```

8.12 LiveChannel常见问题

本文主要介绍 LiveChannel 中遇到的常见问题及解决方案。

OSS livechannel 推流过程

下图为 OSS livechannel 推流过程图解，了解这个过程，有助于排查 livechannel 推流过程中遇到的问题。



更多详情请参考：

- 生成推流 *URL*
- 设置推流状态

案例：录制 **M3u8** 缺失

问题分析：默认录制成品的 m3u8 只有最后 3 片，遵循的是 HLS 协议的默认规则。

解决方案：用 `PostVodPlaylist` 接口将指定时间范围内的 *`ts`* 文件汇聚到一个 m3u8 索引内来解决。

注意:

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

案例：录制 **m3u8** 文件失败

问题分析：录制的 m3u8 文件成功推流到来 OSS 才算录制成功。

解决方案：可以在客户端抓包查看是否有 publish succees 的标志，有这个标志才表示和 OSS 音频包交互成功。所以发现客户端推流有记录，但是就是没有录制视频的情况，可以抓包分析一下。

案例：客户端无法推流到 OSS

使用 ffmpeg 推流不成功：

```
ffmpeg -re -i 0_20180525105430445.aac -acodec aac -strict -2 -f
flv rtmp://xxx.oss-cn-beijing.aliyuncs.com/live/test_1000?Expires
=1540458859&OSSAccessKeyId=LTAlujianb6C9z&Signature=qwh31xQsan
mao6ygCFJgovNIg%3D&playlistName=playlist.m3u8
```

解决方案：

- 使用 ffmpeg 推流不成功的时候建议用最原始的命令推流，不要加一些复杂的参数。
- 推流 URL 在有 & 符号时请将整个 URL 用 双引号（""）囊括起来。例如，ffmpeg -re -i 0_20180525105430445.aac -acodec aac -strict -2 -f flv "rtmp://xxx.oss-cn-beijing.aliyuncs.com/live/test_1000?Expires=1540458859&OSSAccessKeyId=LTAlujianb6C9z&Signature=qwh31xQsanmao6ygCFJgovNIg%3D&playlistName=playlist.m3u8"。
- 尝试更换成 obs 推流测试，确认是否因 ffmpeg 问题导致的推流失败；

案例：录制 M3u8 文件卡顿

转储类型为 HLS 时，写入当前 *ts* 文件的音视频数据时长达到`FragDuration`指定的时长后，OSS 会在收到下一个关键帧的时候切换到下一个 *ts* 文件。如果`max(2*FragDuration, 60s)`后仍未收到下一个关键帧，OSS 强制切换文件，此时可能引起播放时卡顿。

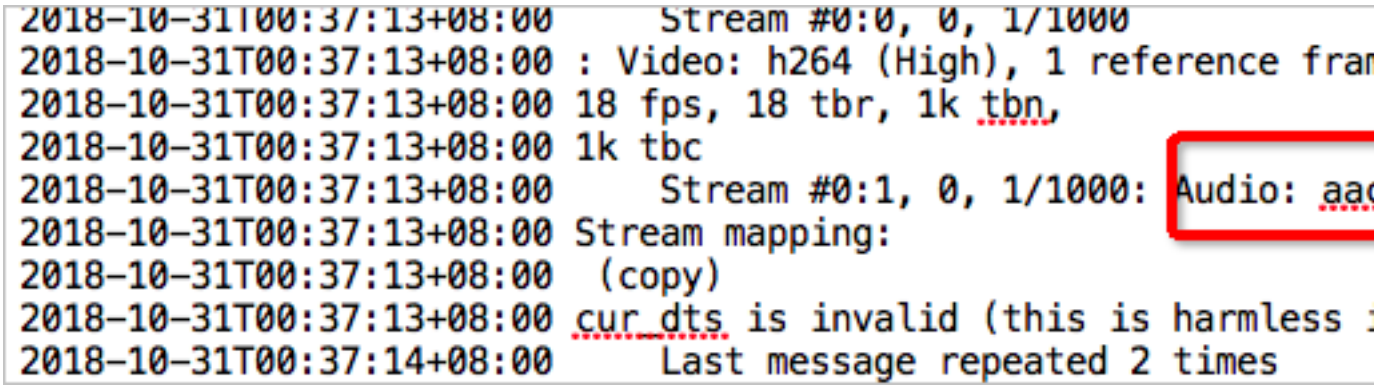
案例：录制 M3u8 文件没有音频或视频

以下几种情况会出现音频或者视频没有录制的情况：

- `AVC header`或者`AAC header`没有发送，可以通过抓包查看。
- `RTMP message`长度小于2，或者`sequence header`非常小。
- 音频`Message`超过缓冲区大小。
- `codec_ ctx`是解码上下文的关键信息，如果携带的音视频数据异常，也会导致录制失败。

案例：ffmpeg 推流到 OSS 录制没有音频

- 直接查看 ffmpeg 记录的日志，确定客户端是否有发送`aac_header`。
- 在客户端抓个 RTMP 的包，查看是否发送了`aac_header`。


2018-10-31T00:37:13+08:00 Stream #0:0, 0, 1/1000
2018-10-31T00:37:13+08:00 : Video: h264 (High), 1 reference fra
2018-10-31T00:37:13+08:00 18 fps, 18 tbr, 1k tbn,
2018-10-31T00:37:13+08:00 1k tbc
2018-10-31T00:37:13+08:00 Stream #0:1, 0, 1/1000: Audio: aa
2018-10-31T00:37:13+08:00 Stream mapping:
2018-10-31T00:37:13+08:00 (copy)
2018-10-31T00:37:13+08:00 cur_dts is invalid (this is harmless
2018-10-31T00:37:14+08:00 Last message repeated 2 times