阿里云
对象存储 OSS

API 参考

文档版本：20181106

法律声明

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

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

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
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
Put Bucket	创建Bucket
Put Bucket ACL	设置Bucket访问权限
Put Bucket Logging	开启Bucket日志
Put Bucket Website	设置Bucket为静态网站托管模式
Put Bucket Referer	设置Bucket的防盗链规则
Put Bucket Lifecycle	设置Bucket中Object的生命周期规则
Get Bucket Acl	获得Bucket访问权限
Get Bucket Location	获得Bucket所属的数据中心位置信息
Get Bucket Logging	查看Bucket的访问日志配置情况
Get Bucket Website	查看Bucket的静态网站托管状态
Get Bucket Referer	查看Bucket的防盗链规则
Get Bucket Lifecycle	查看Bucket中Object的生命周期规则
Delete Bucket	删除Bucket
Delete Bucket Logging	关闭Bucket访问日志记录功能
Delete Bucket Website	关闭Bucket的静态网站托管模式
Delete Bucket Lifecycle	删除Bucket中Object的生命周期规则
Get Bucket(List Object)	获得Bucket中所有Object的信息
Get Bucket Info	获取Bucket信息

关于**Object**的操作

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

关于**Multipart Upload**的操作

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

跨域资源共享**(CORS)**

API	描述
Put Bucket cors	在指定Bucket设定一个CORS的规则
Get Bucket cors	获取指定的Bucket目前的CORS规则

API	描述
Delete Bucket cors	关闭指定Bucket对应的CORS功能并清空所有规则
Option Object	跨域访问preflight请求

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 关于Bucket的操作

4.1 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所在的数据中心确定后，该Bucket下的所有Object将一直存放在对应的地区。更多内容参见访问域名和数据中心。

请求语法

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

细节分析

- 可以通过Put请求中的`x-oss-acl`头来设置Bucket访问权限。目前Bucket有三种访问权限：public-read-write，public-read和private。
- 如果请求的Bucket已经存在，返回409 Conflict。错误码：BucketAlreadyExists。
- 如果想创建的Bucket不符合命名规范，返回400 Bad Request消息。错误码：InvalidBucketName。
- 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。
- 同一用户在同一地域内最多可创建30个bucket。如果超过30个，则返回400 Bad Request消息。错误码：TooManyBuckets。
- 创建的Bucket，如果没有指定访问权限，则默认使用 Private 权限。
- 创建的Bucket，可以指定Bucket的存储类型，可选值为`Standard`、`IA`和`Archive`。
- 创建Bucket，可以指定Bucket的数据容灾类型（DataRedundancyType），取值为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/KyRt8dOPfo
8=
<?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
```

4.2 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>
```

4.3 PutBucketLogging

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

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

说明：

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

请求语法

```
PUT /?logging HTTP/1.1
Date: GMT Date
Content-Length：ContentLength
Content-Type: application/xml
Authorization: SignatureValue
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
    <LoggingEnabled>
        <TargetBucket>TargetBucket</TargetBucket>
        <TargetPrefix>TargetPrefix</TargetPrefix>
    </LoggingEnabled>
</BucketLoggingStatus>
```

请求元素(Request Elements)

表 4-1: 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	指定存放访问日志的Bucket。 子元素：无 父元素：BucketLoggingStatus.LoggingEnabled
TargetPrefix	字符	否	指定最终被保存的访问日志文件前缀。类型：子元素：None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。一个实际的用于存储OSS访问日志的Object名称例子如下：

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

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

LOG文件格式

表 4-2: LOG文件格式

名称	例子	含义
Remote IP	119.140.142.11	请求发起的IP地址（Proxy代理或用户防火墙可能会屏蔽该字段）
Reserved	-	保留字段
Reserved	-	保留字段
Time	[02/May/2012:00:00:04 +0800]	OSS收到请求的时间
Request-URI	“GET /aliyun-logo.png HTTP/1.1“	用户请求的URI(包括query-string)
HTTP Status	200	OSS返回的HTTP状态码
SentBytes	5576	用户从OSS下载的流量
RequestTime (ms)	71	完成本次请求的时间（毫秒）
Referer	`http://www.aliyun.com/product/oss`	请求的HTTP Referer
User-Agent	curl/7.15.5	HTTP的User-Agent头
HostName	oss-example.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

名称	例子	含义
Requester Aliyun ID	1657136103983691	请求者的阿里云ID；匿名访问为“-”
Operation	GetObject	请求类型
Bucket	oss-example	请求访问的Bucket名字
Key	/aliyun-logo.png	用户请求的Key
ObjectSize	5576	Object大小
Server Cost Time (ms)	17	OSS服务器处理本次请求所花的时间（毫秒）
Error Code	NoSuchBucket	OSS返回的错误码
Request Length	302	用户请求的长度（Byte）
UserID	1657136103983691	Bucket拥有者ID
Delta DataSize	280	Bucket大小的变化量；若没有变化为“-”
Sync Request	-	是否是CDN回源请求；若不是为“-”
Reserved	-	保留字段

细节分析

- 源Bucket和目标Bucket必须属于同一个用户。
- 上面所示的请求语法中，“BucketName”表示要开启访问日志记录的bucket；“TargetBucket”表示访问日志记录要存入的bucket；“TargetPrefix”表示存储访问日志记录的object名字前缀，可以为空。
- 源bucket和目标bucket可以是同一个Bucket，也可以是不同的Bucket；用户也可以将多个的源bucket的LOG都保存在同一个目标bucket内（建议指定不同的TargetPrefix）。
- 当关闭一个Bucket的访问日志记录功能时，只要发送一个空的BucketLoggingStatus即可，具体方法可以参考下面的请求示例。
- 所有PUT Bucket Logging请求必须带签名，即不支持匿名访问。
- 如果PUT Bucket Logging请求发起者不是源bucket（请求示例中的BucketName）的拥有者，OSS返回403错误码。
- 如果源bucket不存在，OSS返回错误码：NoSuchBucket。

- 如果PUT Bucket Logging请求发起者不是目标bucket（请求示例中的TargetBucket）的拥有者，OSS返回403；如果目标bucket不存在，OSS返回错误码：InvalidTargetBucketForLogging。
- 源Bucket和目标Bucket必须属于同一个数据中心，否则返回400错误，错误码为：InvalidTargetBucketForLogging。
- PUT Bucket Logging请求中的XML不合法，返回错误码：MalformedXML。
- 源bucket和目标bucket可以是同一个Bucket；用户也可以将不同的源bucket的LOG都保存在同一个目标bucket内（注意要指定不同的TargetPrefix）。
- 源Bucket被删除时，对应的Logging规则也将被删除。
- OSS以小时为单位生成bucket访问的Log文件，但并不表示这个小时的所有请求都记录在这个小时的LOG文件内，也有可能出现在上一个或者下一个LOG文件中。
- OSS生成的Log文件命名规则中的“UniqueString”仅仅是OSS为其生成的UUID，用于唯一标识该文件。
- OSS生成一个bucket访问的Log文件，算作一次PUT操作，并记录其占用的空间，但不会记录产生的流量。LOG生成后，用户可以按照普通的Object来操作这些LOG文件。
- OSS会忽略掉所有以“x-”开头的query-string参数，但这个query-string会被记录在访问LOG中。如果你想从海量的访问日志中，标示一个特殊的请求，可以在URL中添加一个“x-”开头的query-string参数。如：

  `http://oss-example.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。
- 如果用户上传了Content-MD5请求头，OSS会计算body的Content-MD5并检查一致性，如果不一致，将返回InvalidDigest错误码。

示例

开启**bucket**访问日志的请求示例：

```
PUT /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
```

```
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: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
```

4.4 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。

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	有条件，如果RedirectType为Mirror则必须指定

		`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	
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	有条件，当RedirectType不为External或者AliCDN时需要指定

		在RedirectType为External或者AliCDN时生效。 父节点：Redirect	
HttpRedirectCode	HTTP状态码	跳转时返回的状态码，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
```

4.5 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 **4-3:** 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
```

4.6 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 **4-4:** 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.

名称	类型	是否必需	描述
			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
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

名称	类型	是否必需	描述
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
```

4.7 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 4-5: 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
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。

名称	类型	是否必需	描述
			默认值：无 可选值：url

响应元素(Response Elements)

表 4-6: 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
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

名称	类型	描述
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
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>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>00220120222</ID>
            <DisplayName>user-example</DisplayName>
        </Owner>
    </Contents>
```

```
</ListBucketResult>
```

请求示例(含**Prefix**参数) :

```
GET /?prefix=fun HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/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>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>00220120222</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>fun/movie/007.avi</Key>
        <LastModified>2012-02-24T08:43:27.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>00220120222</ID>
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
    <Contents>
        <Key>fun/test.jpg</Key>
        <LastModified>2012-02-24T08:42:32.000Z</LastModified>
        <ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
        <Type>Normal</Type>
        <Size>344606</Size>
        <StorageClass>Standard</StorageClass>
        <Owner>
            <ID>00220120222</ID>
```

```
            <DisplayName>user_example</DisplayName>
        </Owner>
    </Contents>
</ListBucketResult>
```

请求示例(含**prefix**和**delimiter**参数)：

```
GET /?prefix=fun/&delimiter=/ HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS 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>
```

4.8 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 4-7: 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 >
```

4.9 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 **4-8:** 响应元素

名称	类型	描述
BucketInfo	容器	保存Bucket信息内容的容器 子节点：Bucket节点 父节点：无
Bucket	容器	保存Bucket具体信息的容器 父节点：BucketInfo节点
CreationDate	时间	Bucket创建时间。时间格式 2013-07-31T10:56:21.000Z 父节点：BucketInfo.Bucket
ExtranetEndpoint	字符串	Bucket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

名称	类型	描述
		父节点：BucketInfo.Bucket
Name	字符串	Bucket名字 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

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>
```

4.10 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 **4-9:** 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
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>
```

4.11 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。	有条件，父节点RoutingRule指定时必须指定。

		父节点：RoutingRule	
KeyPrefixEquals	字符串	只有匹配此前缀的object才能匹配此规则。 父节点：Condition	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。）	有条件，父节点Redirect指定时必须指定

		父节点：Redirect	
PassQueryString	bool型	执行跳转或者镜像回源时，是否要携带发起请求的请求参数，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	否

		父节点：Redirect	
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。	有条件，当父节点Set指定时必须指定

		父节点：Set	
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>
```

4.12 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 **4-10:** 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

名称	类型	描述
		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>
```

4.13 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>
```

4.14 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
```

4.15 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
```

4.16 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
```

4.17 DeleteBucketLifecycle

通过**DeleteBucketLifecycle**来删除指定Bucket的生命周期配置。

请求语法

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

细节分析

- 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或Lifecycle不存在，返回404 Not Found错误，错误码：NoSuchBucket或NoSuchLifecycle。
- 只有Bucket的拥有者才能删除Bucket的Lifecycle配置。如果试图操作一个不属于你的Bucket，OSS返回403 Forbidden错误，错误码：AccessDenied。

示例

请求示例：

```
DELETE /?lifecycle HTTP/1.1
Host: 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
```

5 关于Object操作

5.1 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 5-1: 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	字符串	过期时间；更详细描述请参照*RFC2616*。 默认值：无 **说明：** OSS不会对这个值进行限制和验证。

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 **说明：** 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

细节分析

- 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。
- 如果试图添加的Object的同名文件已经存在，并且有访问权限。新添加的文件将覆盖原来的文件，成功返回200 OK。
- 如果在PutObject的时候，携带以x-oss-meta-为前缀的参数，则视为user meta，比如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的user meta总大小不能超过8k。
- 如果Header不是*chunked encoding*编码方式，且没有加入Content length参数，会返回411 Length Required错误。错误码：MissingContentLength。
- 如果设定了长度，但是没有发送消息Body，或者发送的body大小小于给定大小，服务器会一直等待，直到time out，返回400 Bad Request消息。错误码：RequestTimeout。
- 如果试图添加的Object所在的Bucket不存在，返回404 Not Found错误。错误码：NoSuchBucket。
- 如果试图添加的Object所在的Bucket没有访问权限，返回403 Forbidden错误。错误码：AccessDenied。
- 如果添加文件长度超过5G，返回错误消息400 Bad Request。错误码：InvalidArgument。
- 如果传入的Object key长度大于1023字节，返回400 Bad Request。错误码：InvalidObjectName。
- PUT一个Object的时候，OSS支持5个 HTTP *RFC2616*协议规定的Header 字段：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type。如果上

传Object时设置了这些Header，则这个Object被下载时，相应的Header值会被自动设置成上传时的值。

- 如果上传Object时指定了x-oss-server-side-encryption Header，则必须设置其值为AES256，否则会返回400和相应错误提示：InvalidEncryptionAlgorithmError。指定该Header后，在响应头中也会返回该Header，OSS会对上传的Object进行加密编码存储，当这个Object被下载时，响应头中会包含x-oss-server-side-encryption，值被设置成该Object的加密算法。

常见问题

- **Content-Md5**计算方式错误

  以上传的内容为 `0123456789` 为例，计算这个字符串的Content-Md5。

以上传的内容为“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='`

示例

请求示例：

```
PUT /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
Expires: Fri, 28 Feb 2012 05:38:42 GMT
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Fri, 24 Feb 2012 06:03:28 GMT
Content-Type: image/jpg
Content-Length: 344606
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=
```

```
[344606 bytes of object data]
```

返回示例：

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

5.2 CopyObject

CopyObject接口用于在Bucket内或同地域的Bucket之间拷贝文件。

该接口可以发送一个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；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；否则返回304 HTTP错误码（预处理失败）。 默认值：无

名称	类型	描述
x-oss-copy-source-if-unmodified-since	字符串	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误。 默认值：无
x-oss-copy-source-if-modified-since	字符串	如果源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。 说明： 如果拷贝操作的源Object地址和目标Object地址相同，则无论**x-oss-metadata-directive**为何值，都会直接替换源Object的元信息。
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） 说明： • 如果指定了**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。

名称	类型	描述
		• 若拷贝操作中未指定**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都是未进行过服务器端加密编码的数据。
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

响应元素(Response Elements)

表 5-2: 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必须属于同一个地域（数据中心）。
  - 不能拷贝通过追加上传方式产生的Object。
  - 如果源Object为符号链接，只拷贝符号链接，不拷贝符号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Header（**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-**开头的，所以要加到签名字符串中。

示例

请求示例：

```
PUT /copy_oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 07:18:48 GMT
x-oss-copy-source: /oss-example/oss.jpg
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:gmnwPKuu20LQEjd+iPkL259A+
n0=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Content-Type: application/xml
Content-Length: 193
Connection: keep-alive
Date: Fri, 24 Feb 2012 07:18:48 GMT
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
 <LastModified>Fri, 24 Feb 2012 07:18:48 GMT</LastModified>
 <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
</CopyObjectResult>
```

5.3 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(Request Parameters)

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，前提条件是用户发送的GET请求必须携带签名。这些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

请求Header

表 5-3: 请求Header

名称	类型	描述
Range	字符串	指定文件传输的范围。如，设定 bytes=0-9，表示传送第0到第9这10个字符。 默认值：无
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
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

名称	类型	描述
		时间格式：GMT时间，例如Fri, 13 Nov 2015 14:47:53 GMT
If-Match	字符串	如果传入期望的ETag和object的 ETag匹配，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误 默认值：无
If-None-Match	字符串	如果传入的ETag值和Object的ETag不匹配，则正常传输文件，并返回200 OK；否则返回304 Not Modified 默认值：无

细节分析

- GetObject通过range参数可以支持断点续传，对于比较大的Object建议使用该功能。
- 如果在请求头中使用Range参数；则返回消息中会包含整个文件的长度和此次返回的范围，例如：Content-Range: bytes 0-9/44，表示整个文件长度为44，此次返回的范围为0-9。如果不符合范围规范，则传送整个文件，并且不在结果中提及Content-Range。
- 如果“If-Modified-Since”元素中设定的时间不符合规范，直接返回文件，并返回200 OK。
- If-Modified-Since和If-Unmodified-Since可以同时存在，If-Match和If-None-Match也可以同时存在。
- 如果包含If-Unmodified-Since并且不符合或者包含If-Match并且不符合，返回412 precondition failed
- 如果包含If-Modified-Since并且不符合或者包含If-None-Match并且不符合，返回304 Not Modified
- 如果文件不存在返回404 Not Found错误。错误码：NoSuchKey。
- 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。
- 在自定义OSS返回请求中的一些Header时，只有请求处理成功（即返回码为200时），OSS才会将请求的header设置成用户GET请求参数中指定的值。
- 若该Object为进行服务器端熵编码加密存储的，则在GET请求时会自动解密返回给用户，并且在响应头中，会返回x-oss-server-side-encryption，其值表明该Object的服务器端加密算法。
- 需要将返回内容进行 GZIP压缩传输的用户，需要在请求的Header中显示方式加入 Accept-Encoding:gzip，OSS会根据文件的Content-Type和文件大小，判断是否返回给用户经过GZIP 压缩的数据。如果采用了GZIP压缩则不会附带etag 信息。目前OSS支持GZIP压缩的Content-Type为HTML、Javascript、CSS、XML、RSS、Json，文件大小需不小于1k。

- 如果文件类型为符号链接，返回目标文件的内容。并且， 响应头中`Content-Length`、`ETag`、`Content-Md5` 均为目标文件的元信息；`Last-Modified`是目标文件和符号链接的最大值；其他均为符号链接的元信息。
- 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。
- 对于Archive归档类型，Object下载需要提交Restore请求，并等待Restore完成；只有在Object的Restore操作完成且超时前，Object才能被下载：
  - 如果没有提交Restore请求，或者上一次提交Restore已经超时， 则返回403错，错误码为：InvalidObjectState。
  - 或者已经提交Restore请求，但数据的Restore操作还没有完成， 则返回403错，错误码为：InvalidObjectState。
  - 只有Restore完成，且没有超时，数据才能直接下载。

示例

请求示例：

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

Range请求示例：

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

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+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]
```

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]
```

符号链接的请求示例：

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

```
Authorization:OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM
=
```

返回示例：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 08 Nov 2016 03:17:58 GMT
Content-Type: application/octet-stream
Content-Length: 20
Connection: keep-alive
x-oss-request-id: 582143E6D3436A212ADCC87D
Accept-Ranges: bytes
ETag: "8086265EFC0211ED1F9A2F09BF462227"
Last-Modified: Tue, 08 Nov 2016 03:17:58 GMT
x-oss-object-type: Symlink
Content-MD5: gIYmXvwCEe0fmi8Jv0YiJw==
```

Archive类型Object的Restore操作已经完成时的请求示例：

```
GET /oss.jpg HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 09:38:30 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+
NIUs=
```

返回示例

```
HTTP/1.1 200 OK
x-oss-request-id: 58F723894529F18D7F000053
x-oss-object-type: Normal
x-oss-restore: ongoing-request="false", expiry-date="Sun, 16 Apr 2017
08:12:33 GMT"
Date: Sat, 15 Apr 2017 09:38:30 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
[354606 bytes of object data]
```

5.4 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 **说明：** 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`

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。

名称	类型	描述
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。
- 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。
- 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。
- 当Position值为0，且没有同名Object存在，那么Append Object可以和Put Object请求一样，设置诸如x-oss-server-side-encryption之类的请求Header。这一点和Initiate Multipart Upload是一样的。如果在Position为0的请求时，加入了正确的x-oss-server-side-encryption头，那么后续的

Append Object响应头部也会包含x-oss-server-side-encryption头，其值表明加密算法。后续如果需要更改meta，可以使用Copy Object请求。

- 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。

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
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
```

5.5 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，OSS也返回状态码204（ No Content）。
- 如果Bucket不存在，返回404 Not Found。
- 如果文件类型为符号链接，只删除符号链接自身。

示例

请求示例：

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

返回示例：

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

```
Server: AliyunOSS
```

5.6 DeleteMultipleObjects

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

Delete Multiple Objects操作支持一次请求内最多删除1000个Object，并提供两种返回模式：详细(verbose)模式和简单(quiet)模式。

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

请求语法

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

请求参数(Request Parameters)

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

名称	描述
encoding-type	指定对返回的Key进行编码，目前支持url编码。Key使用UTF-8字符，但xml 1.0标准不支持解析一些控制字符，比如ascii值从0到10的字符。对于Key中包含xml 1.0标准不支持的控制字符，可以通过指定encoding-type对返回的Key进行编码。 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Delete Multiple Object请求的容器。 子节点：一个或多个Object元素，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，那返回的结果会对Key进行编码。 父节点：容器

细节分析

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

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:+z3gBfnFAxBcBDgx27Y/
jEfbfu8=
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>false</Quiet>
  <Object>
    <Key>multipart.data</Key>
  </Object>
  <Object>
    <Key>test.jpg</Key>
  </Object>
  <Object>
    <Key>demo.jpg</Key>
  </Object>
</Delete>
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 78320852-7eee-b697-75e1-b6db0f4849e7
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length: 244
Content-Type: application/xml
Connection: keep-alive
```

```
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <Deleted>
       <Key>multipart.data</Key>
    </Deleted>
    <Deleted>
       <Key>test.jpg</Key>
    </Deleted>
    <Deleted>
       <Key>demo.jpg</Key>
    </Deleted>
</DeleteResult>
```

请求示例II：

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

5.7 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	如果指定的时间早于实际修改时间，则返回200 OK和Object Meta；否则返回304 not modified 默认值：无
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

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
```

5.8 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
```

5.9 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
```

5.10 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)

名称	类型	描述
		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>
```

5.11 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
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	有条件的

名称	类型	描述	必须
		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。	
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
success_action_status	字符串	未指定success_action_redirect表	

名称	类型	描述	必须
		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-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	可选

名称	类型	描述	必须
		算签名时，与使用普通AccessKeyId签名方式一致。 默认值：无	

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。
Cache-Control, **Content-Type**, **Content-Disposition**, **Content-Encoding**, **Expires**	HTTP请求头。 该condition支持精确匹配和starts-with匹配方式。
key	上传文件的object名称。 该condition支持精确匹配和starts-with匹配方式。
success_action_redirect	上传成功后的跳转URL地址。 该condition支持精确匹配和starts-with匹配方式。
success_action_status	未指定success_action_redirect时，上传成功后的返回状态码。 该condition支持精确匹配和starts-with匹配方式。
x-oss-meta-*	用户指定的user meta。 该condition支持精确匹配和starts-with匹配方式。

如果Post请求中包含其他的表单域，可以将这些额外的表单域加入到policy的conditions中，conditions不涉及的表单域将不会进行合法性检查。

Conditions匹配方式

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

转义字符

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

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

Post Signature

对于验证的Post请求，HTML表单中必须包含policy和Signature信息。policy控制请求中那些值是允许的。计算Signature的具体流程为：

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

示例 **Demo**

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

5.12 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。

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

构造CallBack参数

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

字段	含义	选项
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3MB。 • 支持同时配置最多5个url，以”;”分割。OSS会依次发送请求直到第一个返回成功为止。 • 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`	必选项
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	可选项

字段	含义	选项
	url并将解析出的host填充到callbackHost中	
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。	必选项
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。 • 如果为application/x-www-form-urlencoded，则callbackBody中的变量将会被经过url编码的值替换掉，如果为application/json，则会按照json格式替换其中的变量。	可选项

示例json串如下

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

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

```
imageInfo.width=${imageInfo.width}&imageInfo.format=${imageInfo.format
}&my_var=${x:my_var}"
}
```

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

系统变量	含义
bucket	bucket
object	object
etag	文件的etag，即返回给用户的etag字段
size	object大小，CompleteMultipartUpload时为整个object的大小
mimeType	资源类型，如jpeg图片的资源类型为image/jpeg
imageInfo.height	图片高度
imageInfo.width	图片宽度
imageInfo.format	图片格式，如jpg、png等

自定义参数

用户可以通过callback-var参数来配置自定义参数。

自定义参数是一个Key-Value的Map，用户可以配置自己需要的参数到这个Map。在OSS发起POST回调请求的时候，会将这些参数和上一节所述的系统参数一起放在POST请求的body中以方便接收回调方获取。

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

 说明：

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

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

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

```
}
```

构造回调请求

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

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

说明：

在使用POST请求上传Object的时候只能使用这种方式来指定回调参数。

这三种方式只能同时使用其中一种，否则OSS会返回InvalidArgument错误。

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

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

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

- 如果需要在POST上传Object的时候附带回调参数会稍微复杂一点，callback参数要使用独立的表单域来附加，如下面的示例：

```
--9431149156168
Content-Disposition: form-data; name="callback"
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，举个例子，如果用户的自定义参数的json为

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

那么POST请求的表单域应该如下：

```
--9431149156168
Content-Disposition: form-data; name="callback"
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": "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYW
xsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2
FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJjYWxsYmFja0
JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkIn0="},
    ["starts-with", "$key", "user/eric/"],
  ]
```

```
}
```

发起回调请求

如果文件上传成功，OSS会根据用户的请求中的callback参数和自定义参数（callback-var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
```

返回回调结果

比如应用服务器端返回的回应请求为：

```
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"}
```

返回上传结果

再给客户端的内容为：

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

需要注意的是，如果类似CompleteMultipartUpload这样的请求，在返回请求本身body中存在内容（如XML格式的信息），使用上传回调功能后会覆盖原有的body的内容如`{"a":"b"}`，希望对此处做好判断处理。

回调签名

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

- 生成签名

  签名在OSS端发生，采用RSA非对称方式签名，私钥加密的过程为：

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

  

  说明：

  其中private_key为私钥，只有oss知晓，path为回调请求的资源路径，query_string为查询字符串，body为回调的消息体，所以签名过程由以下几步组成：

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

  如下例：

  ```
  POST /index.php?id=1&index=2 HTTP/1.0
  Host: 121.43.113.8
  Connection: close
  Content-Length: 18
  authorization: kKQeGTRccDKyHB3H9vF+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头保存的是公钥的url地址的base64编码，因此需要对其做base64解码后获取到公钥，即

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

   这里需要注意，用户需要校验`x-oss-pub-key-url`头的值必须以`http://gosspublic.alicdn.com/`或者`https://gosspublic.alicdn.com/`开头，目的是为了保证这个publickey是由OSS颁发的。
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
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()
```

其它语言实现的应用服务器如下：

Java版本：

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

PHP版本：

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

Python版本：

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

C#版本：

- 下载地址：点击这里
- 运行方法：解压后参看 `README.md`。

.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

特别须知

- 如果传入的callback或者callback-var不合法，则会返回400错误，错误码为”InvalidArgument”，不合法的情况包括以下几类：
  - PutObject()和CompleteMultipartUpload()接口中url和header同时传入callback(x-oss-callback)或者callback-var(x-oss-callback-var)参数
  - callback或者callback-var(PostObject()由于没有callback-var参数，因此没有此限制，下同)参数过长（超过5KB）
  - callback或者callback-var没有经过base64编码
  - callback或者callback-var经过base64解码后不是合法的json格式
  - callback参数解析后callbackUrl字段包含的url超过限制（5个），或者url中传入的port不合法，比如

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

  - callback参数解析后callbackBody字段为空
  - callback参数解析后callbackBodyType字段的值不是”application/x-www-form-urlencoded”或者”application/json”

- callback参数解析后callbackBody字段中变量的格式不合法，合法的格式为${var}
- callback-var参数解析后不是预期的json格式，预期的格式应该为`{"x:var1":"value1","x:var2":"value2"...}`

- 如果回调失败，则返回203，错误码为”CallbackFailed”，回调失败只是表示OSS没有收到预期的回调响应，不代表应用服务器没有收到回调请求（比如应用服务器返回的内容不是json格式），另外，此时文件已经成功上传到了OSS
- 应用服务器返回OSS的响应必须带有Content-Length的Header，Body大小不要超过1MB。

Callback支持的地域

Callback目前支持的地域如下：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（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）。

5.13 PutSymlink

PutSymlink可用于针对OSS上的TargetObject创建符号链接，用户可以通过该符号链接访问TargetObject。

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	符号链接指向的目标文件。 合法值：命名规范同Object。

细节分析

- TargetObjectName同ObjectName一样，需要URL encode。
- 符号链接的目标文件类型不能为符号链接。
- 创建符号链接时，
  - 不检查目标文件是否存在

— 不检查目标文件类型是否合法

— 不检查目标文件是否有权限访问

以上检查，都推迟到GetObject等需要访问目标文件的API。

- 如果试图添加的文件已经存在，并且有访问权限。新添加的文件将覆盖原来的文件，成功返回200 OK。
- 如果在PutSymlink的时候，携带以x-oss-meta-为前缀的参数，则视为user meta，比如x-oss-meta-location。一个Object可以有多个类似的参数，但所有的user meta总大小不能超过8k。
- 如果Bucket的类型为Archive，则不能调用该接口，否则返回400错误，错误码为OperationNotSupported。

示例

请求示例：

```
PUT /link-to-oss.jpg?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
Content-Disposition: attachment;filename=oss_download.jpg
Date: Tue, 08 Nov 2016 02:00:25 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=
x-oss-symlink-target: oss.jpg
```

返回示例：

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

5.14 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"
```

5.15 RestoreObject

RestoreObject接口用于服务端执行解冻任务。

只针对归档类型的Object读取，需要调用RestoreObject接口让服务端执行解冻任务。如果一个Object是标准或者低频访问类型，不要调用该接口。

归档类型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操作后，Object将处于解冻中的状态，服务端执行解冻。
3. 待服务端执行完成解冻任务后，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1天，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，之后，Object又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。在此期间内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，再次解冻的首次读取数据会收取数据取回费用。

请求语法

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

细节分析

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

示例

首次提交**restore**的请求示例：

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

再次调用，且**restore**没有完成时：

```
POST /oss.jpg?restore HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
```

```
Date: Sat, 15 Apr 2017 07:45:29 GMT
Authorization: OSS 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>
```

再次调用，且**restore**已经完成时：

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

5.16 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>
   <KeepAllColumns>false|true</KeepAllColumns>
  </CSV>
 <OutputRawData>false|true</OutputRawData>
      <EnablePayloadCrc>true</EnablePayloadCrc>
  <OutputHeader>false</OutputHeader>
  </OutputSerialization>
  <Options>
 <SkipPartialDataRecord>false</SkipPartialDataRecord>
  </Options>
</SelectRequest>
```

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

名称	类型	描述
CSV（InputSerialization）	容器	输入CSV的格式参数（可选） 子节点：FileHeaderInfo、RecordDelimiter、FieldDelimiter、QuoteCharacter、CommentCharacter、 Range 父节点：InputSerialization
CSV(OutputSerialization)	容器	输出CSV的格式参数（可选） 子节点： RecordDelimiter、FieldDelimiter、KeepAllColumns 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）
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。

名称	类型	描述
		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）
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

名称	类型	描述
		父节点：CSV（输出）
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
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。
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>
   <KeepAllColumns>false</KeepAllColumns>
 </CSV>
 <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

名称	类型	描述
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
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 {
        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'

应用场景	SQL语句
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
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

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

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

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

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。