
阿里云开放存储服务

OSS

目 录

前 言	3
1. 阿里云存储服务简介	4
2. 基本概念.....	5
2.1 Object.....	5
2.2 Bucket.....	5
2.3 Access Key ID、Access Key Secret.....	5
2.4 Service	6
3. OSS 功能简介	7
3.1 OSS 基本功能	7
3.2 Object 外链地址的构成规则	7
3.3 OSS 防盗链	8
3.4 访问日志记录(Server Access Logging).....	8
3.5 服务器端加密编码	11
4. 访问控制.....	13
4.1 用户签名验证（Authentication）	13
4.2 在 Head 中包含签名.....	13
4.3 在 URL 中包含签名	18
4.4 Bucket 权限控制.....	19
5. 开放接口规范.....	21
5.1 公共 HTTP 头定义	22
5.1.1 公共请求头（Common Request Headers）	22
5.1.2 公共响应头（Common Response Headers）	22
5.2 关于 Service 的操作	23
5.2.1 GetService (ListBucket).....	23
5.3 关于 Bucket 的操作.....	26
5.3.1 Delete Bucket	26
5.3.2 Get Bucket (List Object)	28
5.3.3 Get Bucket Acl.....	36

5.3.4	Put Bucket	38
5.3.5	Put Bucket Acl	40
5.4	关于 Object 操作	42
5.4.1	Copy Object.....	42
5.4.2	Delete Object	46
5.4.3	Delete Multiple Objects	47
5.4.4	Get Object.....	51
5.4.5	Head Object	55
5.4.6	Put Object	57
5.5	关于 Multipart Upload 的操作	60
5.5.1	Initiate Multipart Upload.....	60
5.5.2	Upload Part.....	63
5.5.3	Complete Multipart Upload.....	65
5.5.4	Abort Multipart Upload.....	69
5.5.5	List Multipart Uploads	70
5.5.6	List Parts.....	74
6.	OSS 的错误响应	78
6.1	OSS 的错误响应格式	78
6.2	OSS 的错误码	79
6.3	OSS 不支持分块传输编码	81
6.4	OSS 不支持的操作.....	82
6.5	OSS 操作支持但参数不支持的操作.....	84

前 言

本文档是阿里云存储服务（OSS）的开发帮助指南，描述了 OSS 中的基本概念、提供的服务以及可用的 API。

1. 阿里云存储服务简介

阿里云存储服务（OpenStorageService，简称 OSS），是阿里云对外提供的海量，安全，低成本，高可靠的云存储服务。用户可以通过简单的 REST 接口，在任何时间、任何地点、任何互联网设备上上传和下载数据，也可以使用 WEB 页面对数据进行管理。同时，OSS 提供 Java、Python、PHP、C#语言的 SDK，简化用户的编程。基于 OSS，用户可以搭建出各种多媒体分享网站、网盘、个人和企业数据备份等基于大规模数据的服务。

公网的 OSS 访问地址：<http://oss.aliyuncs.com>

阿里云主机的内网 OSS 访问地址¹：<http://oss-internal.aliyuncs.com>

OSS 的 web 控制台地址：<http://oss.aliyun.com/>

¹ 阿里云主机通过内网域名访问 OSS，会拥有比访问公网域名更快的响应时间、更高的传输速度；同时，上传、下载流量都是免费的。

2. 基本概念

2.1 Object

在 OSS 中，用户操作的基本数据单元是 Object。单个 Object 最大允许存储 5TB 的数据。Object 包含 key、meta 和 data。其中，key 是 Object 的名字；meta 是用户对该 object 的描述，由一系列 name-value 对组成；data 是 Object 的数据。

■ Object 命名规范

- 使用 UTF-8 编码
- 长度必须在 1-1023 字节之间
- 不能以 “/” 或者 “\” 字符开头

2.2 Bucket

Bucket 是 OSS 上的命名空间，也是计费、权限控制、日志记录等高级功能的管理实体；Bucket 名称在整个 OSS 服务中具有全局唯一性，且不能修改；存储在 OSS 上的每个 Object 必须都包含在某个 Bucket 中。一个应用，例如图片分享网站，可以对应一个或多个 Bucket。一个用户最多可创建 10 个 Bucket，但每个 Bucket 中存放的 Object 的数量和大小总和没有限制，用户不需要考虑数据的可扩展性。

■ Bucket 命名规范

- 只能包括小写字母，数字，短横线（-）
- 必须以小写字母或者数字开头
- 长度必须在 3-63 字节之间

2.3 Access Key ID、Access Key Secret

用户注册 OSS 时，系统会给用户分配一对 Access Key ID 和 Access Key Secret，

称为 ID 对，用于标识用户，为访问 OSS 做签名验证。

2.4 Service

OSS 提供给用户的虚拟存储空间，在这个虚拟空间中，每个用户可拥有一个或多个 Bucket。

3. OSS 功能简介

3.1 OSS 基本功能

OSS 为用户提供数据存储服务，用户可以通过以下操作来处理 OSS 上的数据：

- 创建、查看、罗列、删除 Bucket
- 修改、获取 Bucket 的访问权限
- 上传、查看、罗列、删除、批量删除 Object
- 对于大文件支持分片上传（Multi-Part Upload）
- 访问时支持 If-Modified-Since 和 If-Match 等 HTTP 参数

3.2 Object 外链地址的构成规则

如果一个 bucket 设置成公开读权限（详见下一章：访问控制），意味着你允许其他用户来访问属于你的 object。你的 object 的外链地址构成规则如下：


```
http://<你的bucket 名字>.oss.aliyuncs.com/<你的object 名字>
```

例如，在一个名为 oss-example 的 bucket 中，有一个名为"aliyun-log.png"的 object（content-type 为 image/png）。那么这个 object 的外链 URL 为：

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

构成规则的示意图如下：

http://oss-example.oss.



bu **host**

用户可以直接该 URL 链接放入 HTML 中使用：

```


```

3.3 OSS 防盗链

OSS 是按使用收费的服务，为了防止用户在 OSS 上的数据被其他人盗链，OSS 支持基于 HTTP header 中表头字段 `referer` 的防盗链方法。目前，只有通过 OSS 的控制台 (<http://oss.aliyun.com>) 可以对一个 bucket 设置 `referer` 字段的白名单和是否允许 `referer` 字段为空的请求访问。例如，对于一个名为 *oss-example* 的 bucket，设置其 `referer` 白名单为 *http://www.aliyun.com*。则所有 `referer` 为 *http://www.aliyun.com* 的请求才能访问 *oss-example* 这个 bucket 中的 Object。

细节分析：

- 1) 用户只有通过 URL 签名或者匿名访问 Object 时，才会做防盗链验证。请求的 Header 中有 “Authorization” 字段的，不会做防盗链验证。
- 2) 一个 bucket 可以支持多个 `referer` 参数，这些参数之间由 “,” 号分隔。
- 3) Referer 参数支持通配符 “*” 和 “?”。
- 4) 用户可以设置是否允许 `referer` 字段为空的请求访问。
- 5) 白名单为空时，不会检查 `referer` 字段是否为空（不然所有的请求都会被拒绝）。
- 6) 白名单不为空，且设置了不允许 `referer` 字段为空的规则；则只有 `referer` 属于白名单的请求被允许，其他请求（包括 `referer` 为空的请求）会被拒绝。
- 7) 如果白名单不为空，但设置了允许 `referer` 字段为空的规则；则 `referer` 为空的请求和符合白名单的请求会被允许；其他请求都会被拒绝。
- 8) Bucket 的三种权限（`private`，`public-read`，`public-read-write`）都会检查 `referer` 字段。

 注意：OSS 更多的防盗链的规则正在开发中。

3.4 访问日志记录²(Server Access Logging)

OSS 为用户提供自动保存访问日志记录功能。Bucket 的拥有者可以通过 OSS

² OSS 提供 Bucket 访问日志的目的是方便 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。

控制台 (<http://oss.aliyun.com>), 为其所拥有的 bucket 开启访问日志记录功能。当一个 bucket (源 Bucket, Source Bucket) 开启访问日志记录功能后, OSS 自动将访问这个 bucket 的请求日志, 以小时为单位, 按照固定的命名规则, 生成一个 Object 写入用户指定的 bucket (目标 Bucket, Target Bucket)。

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 (从左至右, 以空格分隔):

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)
Referrer	http://oss.aliyun.com	请求的 HTTP Referrer
User-Agent	curl/7.15.5	HTTP 的 User-Agent 头

HostName	oss-example.oss.aliyuncs.com	请求访问域名
Request ID	505B01695037C2AF032593A4	用于唯一标示该请求的 UUID
LoggingFlag	true	是否开启了访问日志功能
Reserved	-	保留字段
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“-”

细节分析：

- 1) 源 Bucket 和目标 Bucket 必须属于同一个用户。
- 2) “*TargetPrefix*” 表示存储访问日志记录的 object 名字前缀，可以为空。
- 3) 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）。
- 4) 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。
- 5) 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。
- 6) 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。

- 7) OSS 会忽略掉所有以 “x-” 开头的 query-string 参数，但这个 query-string 会被记录在访问 LOG 中。如果你想从海量的访问日志中，标示一个特殊的请求，可以在 URL 中添加一个 “x-” 开头的 query-string 参数。如：

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

OSS 处理上面两个请求，结果是一样的。但是在访问 LOG 中，你可以通过搜索 “x-user=admin”，很方便地定位出经过标记的这个请求。

- 8) 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。
- 9) 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。

3.5 服务器端加密编码

OSS 支持在服务器端对用户上传的数据进行加密编码（Server-Side Encryption）：用户上传数据时，OSS 对收到的用户数据进行加密编码，然后再将编码得到的数据永久保存下来；用户下载数据时，OSS 自动对保存的编码数据进行解码并把原始数据返回给用户，并在返回的 HTTP 请求 Header 中声明该数据进行了服务器端加密编码。换句话说，下载一个进行服务器端加密编码的 Object 和下载一个普通的 Object 没有任何区别，因为 OSS 会为用户管理整个编解码过程。

目前，OSS 的服务器端加密编码是 Object 的一个属性。用户创建一个 Object 的时候，只需要在 Put Object 的请求中携带 x-oss-server-side-encryption 的 HTTP Header，并指定其值为 AES256，即可以实现该 Object 的服务器端加密编码存储。目前支持服务器端加密编码的操作包括：

- Put Object
- Copy Object
- Initiate Multipart Upload

以 Put Object 请求为例，声明服务器端加密编码的请求如下：

```
PUT /ObjectName HTTP/1.1
Content-Length: ContentLength
```

```
Content-Type: ContentType
Host: BucketName.oss.aliyuncs.com
x-oss-server-side-encryption: AES256
Date: GMT Date
Authorization: SignatureValue
```

```
[object data]
```

Get 一个进行了服务器端加密编码的 Object 时, OSS 返回的 HTTP 请求例子:

```
HTTP/1.1 200 OK
x-oss-request-id: 3a89276f-2e2d-7965-3ff9-51c875b99c41
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpeg
Content-Length: 344606
x-oss-server-side-encryption: AES256
Server: AliyunOSS
```

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

细节分析:

- 1) Object Group 不支持服务器端加密编码。
- 2) 除了 Put Object、Copy Object 和 Initiate Multipart Upload 以外, 其他 OSS 收到的请求中如果出现 x-oss-server-side-encryption 头, OSS 会直接返回 HTTP 状态码: 400; 并在消息体内注明错误码是: InvalidArgument。
- 3) 目前, OSS 只支持 AES256 加密编码算法, 如果用户指定 x-oss-server-side-encryption 头为其他值, OSS 会直接返回 HTTP 状态码: 400; 并在消息体内注明错误码是: InvalidEncryptionAlgorithmError。
- 4) 通过服务器端加密编码存储的 Object, 在下述 API 请求中 OSS 会返回 x-oss-server-side-encryption 头, 其值为服务器端使用的熵编码加密算法:
 - Put Object
 - Copy Object
 - Initiate Multipart Upload
 - Upload Part
 - Complete Multipart Upload
 - Get Object

4. 访问控制

4.1 用户签名验证 (Authentication)

OSS 通过使用 Access Key ID/ Access Key Secret 对称加密的方法来验证某个请求的发送者身份。Access Key ID 用于标示用户，Access Key Secret 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 Access Key Secret 必须保密，只有用户和 OSS 知道。每个 Access Key 对 (Access Key ID 和 Access Key Secret) 都有 active/inactive 两种状态。

- active 表明用户可以用此 ID 对签名验证请求
- inactive 表明用户暂停此 ID 对签名验证的功能

一个用户可以同时拥有 0 至 2 个 active 或者 inactive 的 ID 对。用户可以登录 <http://oss.aliyun.com/>，在 OSS 管理控制台申请新增或删除 ID 对。

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

4.2 在 Head 中包含签名

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

Authorization 字段计算方法如下：

"Authorization: OSS " + Access Key Id + ":" + Signature

```
Signature = base64(hmac-sha1(VERB + "\n"
    + CONTENT-MD5 + "\n"
    + CONTENT-TYPE + "\n"
    + DATE + "\n"
    + CanonicalizedOSSHeaders
    + CanonicalizedResource))
```

- CONTENT-MD5 表示请求内容数据的 MD5 值
- CONTENT-TYPE 表示请求内容的类型
- DATE 表示此次操作的时间，且必须为 HTTP1.1 中支持的 GMT 格式
- CanonicalizedOSSHeaders 表示 http 中的 object meta 组合
- CanonicalizedResource 表示用户想要访问的 OSS 资源

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

构建 CanonicalizedOSSHeaders 的方法：

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

- 1) 将所有以“x-oss-”为前缀的 HTTP 请求头的名字转换成小写字母。如‘X-OSS-Meta-Name: TaoBao’转换成‘x-oss-meta-name: TaoBao’。
- 2) 将上一步得到的所有 HTTP 请求头按照字典序进行升序排列。
- 3) 如果有相同名字的请求头，则根据标准 RFC 2616, 4.2 章进行合并（两个值之间只用逗号分隔）。如有两个名为‘x-oss-meta-name’的请求头，对应的值分别为‘TaoBao’和‘Alipay’，则合并后为：‘x-oss-meta-name:TaoBao,Alipay’。
- 4) 删除请求头和内容之间分隔符两端出现的任何空格。如‘x-oss-meta-name: TaoBao,Alipay’转换成：‘x-oss-meta-name:TaoBao,Alipay’。
- 5) 将所有的头和内容用‘\n’分隔符分隔拼成最后的 CanonicalizedOSSHeader。

构建 CanonicalizedResource 的方法：

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

- 1) 将 CanonicalizedResource 置成空字符串 (“”);

- 2) 放入要访问的 OSS 资源: “ */BucketName/ObjectName* ” (无 *ObjectName* 则不填)
- 3) 如果请求的资源包括子资源(sub-resource)³, 那么将所有的子资源按照字典序, 从小到大排列并以 ‘&’ 为分隔符生成子资源字符串。在 *CanonicalizedResource* 字符串尾添加 “?” 和子资源字符串。此时的 *CanonicalizedResource* 例子如: */BucketName/ObjectName?acl &uploadId=UploadId*
- 4) 如果用户请求在查询字符串(query string)中指定了要重写(override)返回请求的 header⁴, 那么将这些查询字符串及其请求值按照字典序, 从小到大排列, 以 ‘&’ 为分隔符, 按参数的字典序添加到 *CanonicalizedResource* 中。此时的 *CanonicalizedResource* 例子:
/BucketName/ObjectName?acl&response-content-type=ContentType & uploadId =UploadId

例如: 想签名以下信息:

```
PUT /nelson HTTP/1.0
Content-Md5: c8fdb181845a4ca6b8fec737b3581d76
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.oss.aliyuncs.com
X-OSS-Meta-Author: foo@bar.com
X-OSS-Magic: abracadabra
```

假如 AccessID 是: “44CF9590006BF252F707”

Access Key Secret 是 “OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV”,

可用以下方法签名 (Signature):

python 示例代码:

```
import base64
import hmac
import sha

h = hmac.new("OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
             "PUT\nc8fdb181845a4ca6b8fec737b3581d76\ntext/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nelson", sha)

base64.encodestring(h.digest()).strip()
```

签名 (Signature) 计算结果应该为 “dZpCvvKgxiFw6wvMHHj5g3W6STM=”, 然后加上 Authorization 头来组成最后需要发送的消息:

```
PUT /nelson HTTP/1.0
Authorization: OSS 44CF9590006BF252F707: dZpCvvKgxiFw6wvMHHj5g3W6STM=
Content-Md5: c8fdb181845a4ca6b8fec737b3581d76
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
```

³ OSS 目前支持的子资源包括: *acl*, *group*, *uploadId*, *partNumber*, *uploads*, *logging*

⁴ OSS 目前支持的 *override* 查询字符串包括: *response-content-type*, *response-content-language*, *response-expires*, *response-cache-control*, *response-content-disposition*, *response-content-encoding*

Host: oss-example.oss.aliyuncs.com

X-OSS-Meta-Author: foo@bar.com

X-OSS-Magic: abracadabra

在计算签名头的时候请遵循如下规则：

- 1) 用来签名的字符串必须为 UTF-8 格式。含有中文字符的签名字符串必须先进行 UTF-8 编码，再与 Access Key Secret 计算最终签名。
- 2) 签名的方法用 RFC 2104 (<http://www.ietf.org/rfc/rfc2104.txt>) 中定义的 HMAC-SHA1 方法，其中 Key 为 Access Key Secret。
- 3) content-type 和 content-md5 在请求中不是必须的，但是如果请求需要签名验证，空值的话以换行符 “\n” 代替。
- 4) 在所有非 HTTP 标准定义的 header 中，只有以 “x-oss-” 开头的 header，需要加入签名字符串；其他非 HTTP 标准 header 将被 OSS 忽略（如上例中的 x-oss-magic）。
- 5) 以 “x-oss-” 开头的 head 在签名验证前需要符合以下规范：
 - head 的名字需要变成小写。
 - head 按字典序从小到大排序。
 - 分割 head name 和 value 的冒号前后不能有空格。
 - 每个 Head 之后都有一个换行符 “\n”，如果没有 Head，CanonicalizedOSSHeaders 就设置为空。

细节分析：

- 1) 如果传入的 AccessID 不存在或 inactive，返回 403 Forbidden。错误码：InvalidAccessKeyId。
- 2) 若用户请求头中 Authorization 值的格式不对，返回 400 Bad Request。错误码：InvalidArgument。
- 3) OSS 所有的请求都**必须**使用 HTTP 1.1 协议规定的 [GMT 时间格式](#)。其中，日期的格式有三种：

<i>date1 = 2DIGIT SP month SP 4DIGIT; day month year (e.g., 02 Jun 1982)</i>
<i>date2 = 2DIGIT "-" month "-" 2DIGIT; day-month-year (e.g., 02-Jun-82)</i>
<i>date3 = month SP (2DIGIT / (SP 1DIGIT)); month day (e.g., Jun 2) 【注意“2”前面有两个空格】</i>

上述这三种日期格式中，“天”所占位数都是“2 DIGIT”。因此，“Jun 2”、“2 Jun 1982”和“2-Jun-82”都是非法日期格式。

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

返回示例：

```
<?xml version="1.0" ?>
<Error>
  <Code>
    SignatureDoesNotMatch
  </Code>
  <Message>
    The request signature we calculated does not match the signature you provided. Check your key and
    signing method.
  </Message>
  <StringToSignBytes>
    47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31 31 20 30 37 3a 35 39 3a 32 35 20 47
    4d 54 0a 2f 75 73 72 65 61 6c 74 65 73 74 3f 61 63 6c
  </StringToSignBytes>
  <RequestId>
    1E446260FF9B10C2
  </RequestId>
  <HostId>
    oss.aliyuncs.com
  </HostId>
  <SignatureProvided>
    y5H7yzPsA/tP4+0tH1HHvPEwUv8=
  </SignatureProvided>
  <StringToSign>
    GET
    Wed, 11 May 2011 07:59:25 GMT
    /oss-example?acl
  </StringToSign>
```

```
<OSSAccessKeyId>
    AKIAIVAKMSMOY7VOMRWQ
</OSSAccessKeyId>
</Error>
```

4.3 在 URL 中包含签名

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

URL 中包含签名示例：

```
http://oss-example.oss.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=44CF9590006BF252F707&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

在 URL 中实现签名，必须至少包含 Signature, Expires, OSSAccessKeyId 三个参数。Expires 这个参数的值是一个 UNIX 时间（自 UTC 时间 1970 年 1 月 1 号开始的秒数，详见 [wiki](#)），用于标识该 URL 的超时时间。如果 OSS 接收到这个 URL 请求的时候晚于签名中包含的 Expires 参数时，则返回请求超时的错误码。例如：当前时间是 1141889060，开发者希望创建一个 60 秒后自动失效的 URL，则可以设置 Expires 时间为 1141889120。

所有的 OSS 支持的请求和各种 Head 参数，在 URL 中进行签名的方法和上节介绍的签名算法基本一样。主要区别如下：

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

URL 中添加签名的 python 示例代码为：

```
import base64
import hmac
```

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

细节分析:

- 1) 使用在 URL 中签名的方式, 会将你授权的数据在过期时间以内暴露在互联网上, 请预先评估使用风险。
- 2) PUT 和 GET 请求都支持在 URL 中签名。
- 3) 在 URL 中添加签名时, Signature, Expires, OSSAccessKeyId 顺序可以交换, 但是如果 Signature, Expires, OSSAccessKeyId 缺少其中的一个或者多个, 返回 403 Forbidden。错误码: AccessDenied。
- 4) 如果访问的当前时间晚于请求中设定的 Expires 时间, 返回 403 Forbidden。错误码: AccessDenied。
- 5) 如果传入请求时间, 必须在 OSS 服务器当前时间之后的 15 分钟以内。否则返回 403 Forbidden。错误码: RequestTimeTooSkewed。
- 6) 如果 Expires 时间格式错误, 返回 403 Forbidden。错误码: AccessDenied。
- 7) 如果 URL 中包含参数 Signature, Expires, OSSAccessKeyId 中的一个或者多个, 并且 Head 中也包含签名消息, 返回消息 400 Bad Request。错误码: InvalidArgument。
- 8) 生成签名字符串时, 除 Date 被替换成 Expires 参数外, 仍然包含 content-type、content-md5 等上节中定义的 Header。(请求中虽然仍然有 Data 这个请求头, 但不需要将 Data 加入签名字符串中)

4.4 Bucket 权限控制

OSS 提供 Bucket 级别的权限访问控制, Bucket 目前有三种访问权限: public-read-write, public-read 和 private, 它们的含义如下:

- **public-read-write:** 任何人(包括匿名访问)都可以对该 bucket 中的

object 进行 PUT, Get 和 Delete 操作；所有这些操作产生的费用由该 bucket 的创建者承担，请慎用该权限。

■ **public-read:** 只有该 bucket 的创建者可以对该 bucket 内的 Object 进行写操作（包括 Put 和 Delete Object）；任何人（包括匿名访问）可以对该 bucket 中的 object 进行读操作（Get Object）。

■ **private:** 只有该 bucket 的创建者可以对该 bucket 内的 Object 进行读写操作（包括 Put、Delete 和 Get Object）；其他人无法访问该 Bucket 内的 Object。

用户新创建一个新 Bucket 时，如果不指定 Bucket 权限，OSS 会自动为该 Bucket 设置 private 权限。对于一个已经存在的 Bucket，只有它的创建者可以通过 OSS 的 Put Bucket Acl 接口修改该 Bucket 的权限。

相关阅读:

- [Put Bucket Acl](#)
- [Get Bucket Acl](#)

5. 开放接口规范

本章主要介绍 OSS 的开放接口。开发者在发送请求给 OSS 时，既可以使用带签名认证的请求，也可以使用匿名访问。当签名验证错误或者是访问没有权限的资源时，OSS 返回的错误码请参考下一章，本章就不在举例了。

5.1 公共 HTTP 头定义

5.1.1 公共请求头（Common Request Headers）

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

名称	描述
Authorization	用于验证请求合法性的认证信息。 类型：字符串 默认值：无 使用场景：非匿名请求
Content-Length	RFC 2616中定义的HTTP请求内容长度。 类型：字符串 默认值：无 使用场景：需要向OSS提交数据的请求
Content-Type	RFC 2616中定义的HTTP请求内容类型。 类型：字符串 默认值：无 使用场景：需要向 OSS 提交数据的请求
Date	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT 类型：字符串 默认值：无
Host	访问Host值，格式为：<bucketname>.oss.aliyuncs.com。 类型：字符串 默认值：无

5.1.2 公共响应头（Common Response Headers）

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

名称	描述
Content-Length	RFC 2616中定义的HTTP请求内容长度。 类型：字符串 默认值：无 使用场景：需要向 OSS 提交数据的请求

Connection	<p>标明客户端和OSS服务器之间的链接状态。</p> <p>类型：枚举</p> <p>有效值：open close</p> <p>默认值：无</p>
Date	<p>HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT</p> <p>类型：字符串</p> <p>默认值：无</p>
ETag	<p>ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。</p> <p>类型：字符串</p> <p>默认值：无</p>
Server	<p>生成Response的服务器。</p> <p>类型：字符串</p> <p>默认值：AliyunOSS</p>
x-oss-request-id	<p>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。</p> <p>类型：字符串</p> <p>默认值：无</p>

5.2 关于 Service 的操作

5.2.1 GetService (ListBucket)

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

请求语法：

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

响应元素(Response Elements)

名称	描述
Bucket	保存bucket信息的容器。

	类型：容器 子节点：Name, CreationDate 父节点：ListAllMyBucketsResult.Buckets
Buckets	保存多个Bucket信息的容器。 类型：容器 子节点：Bucket 父节点：ListAllMyBucketsResult
CreateDate	Bucket创建时间 类型：时间 (格式：yyyy-mm-ddThh:mm:ss.timezone, e.g., 2011-12-01T12:27:13.000Z) 父节点：ListAllMyBucketsResult.Buckets.Bucket
DisplayName	Bucket拥有者的名称 (目前和ID一致)。 类型：字符串 父节点：ListAllMyBucketsResult.Owner
ID	Bucket拥有者的用户ID。 类型：字符串 父节点：ListAllMyBucketsResult.Owner
ListAllMyBucketsResult	保存Get Service请求结果的容器。 类型：容器 子节点：Owner, Buckets 父节点：None
Name	Bucket名称。 类型：字符串 父节点：ListAllMyBucketsResult.Buckets.Bucket
Owner	用于存放Bucket拥有者信息的容器。 类型：容器 父节点：ListAllMyBucketsResult

细节分析：

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

请求示例：

```
GET / HTTP/1.1
Host: oss.aliyuncs.com
Date: Fri, 24 Feb 2012 02:58:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:MiorP5BDFDhKAn44wDnkSSv2Z94=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 60633d3c-1293-0d72-7739-759423f02d36
Date: Fri, 24 Feb 2012 02:58:28 GMT
Content-type: application/xml
Content-Length: 685
Connection: close
Server: AliyunOSS
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://doc.oss.aliyuncs.com">
  <Owner>
    <ID>00220120222</ID>
    <DisplayName>oss_doc</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>multipart_upload</Name>
      <CreationDate>2012-02-22T08:25:07.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>my_oss</Name>
      <CreationDate>2012-02-24T02:53:26.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

相关阅读:

- [Get Bucket \(List Object\)](#)
- [Get Object](#)

5.3 关于 Bucket 的操作

5.3.1 Delete Bucket

Delete Bucket 用于删除某个 Bucket。

请求语法：

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

细节分析：

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

请求示例：

```
DELETE / HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Fri, 24 Feb 2012 05:31:04 GMT
Authorization: OSS qn6qrrxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

返回示例：

```
HTTP/1.1 204 No Content
x-oss-request-id: 7faf664d-0cad-852e-4b38-2ac2232e7e7f
Date: Fri, 24 Feb 2012 05:31:04 GMT
Connection: close
Content-Length: 0
Server: AliyunOSS
```

相关阅读：

-
- [Put Bucket](#)
 - [Delete Object](#)

5.3.2 Get Bucket (List Object)

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

请求语法:

```
GET / HTTP/1.1
Host: BucketName.oss.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。

名称	描述
delimiter	是一个用于对 Object 名字进行分组的字符。所有名字包含指定的前缀且第一次出现 delimiter 字符之间的 object 作为一组元素——CommonPrefixes。 数据类型: 字符串 默认值: 无
marker	设定结果从 marker 之后按字母排序的第一个开始返回。 数据类型: 字符串 默认值: 无
max-keys	限定此次返回 object 的最大数, 如果不设定, 默认为 1000, max-keys 取值不能大于 1000。 数据类型: 字符串 默认值: 100
prefix	限定返回的 object key 必须以 prefix 作为前缀。注意使用 prefix 查询时, 返回的 key 中仍会包含 prefix。 数据类型: 字符串 默认值: 无

响应元素(Response Elements)

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

	<p>object 名称的集合。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>
Delimiter	<p>是一个用于对 Object 名字进行分组的字符。所有名字包含指定的前缀且第一次出现 delimiter 字符之间的 object 作为一组元素——CommonPrefixes。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>
DisplayName	<p>Object 拥有者的名字。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents.Owner</p>
ETag	<p>ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents</p>
ID	<p>Bucket拥有者的用户ID。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents.Owner</p>
IsTruncated	<p>指明是否所有的结果都已经返回；“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。</p> <p>类型：枚举字符串</p> <p>有效值：true false</p> <p>父节点：ListBucketResult</p>
Key	<p>Object 的 Key.</p> <p>类型：字符串</p> <p>父节点：ListBucketResult.Contents</p>
LastModified	<p>Object 最后被修改的时间。</p> <p>类型：时间</p> <p>父节点：ListBucketResult.Contents</p>
ListBucketResult	<p>保存 Get Bucket 请求结果的容器。</p> <p>类型：容器</p> <p>子节点：Name, Prefix, Marker, MaxKeys, Delimiter, IsTruncated, Nextmarker, Contents</p> <p>父节点：None</p>
Marker	<p>标明这次 Get Bucket (List Object) 的起点。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>
MaxKeys	<p>响应请求内返回结果的最大数目。</p> <p>类型：字符串</p> <p>父节点：ListBucketResult</p>
Name	<p>Bucket 名字</p> <p>类型：字符串</p>

	父节点: ListBucketResult
Owner	保存 Bucket 所有者信息的容器。 类型: 容器 子节点: DisplayName, ID 父节点: ListBucketResult
Prefix	本次查询结果的开始前缀。 类型: 字符串 父节点: ListBucketResult
Size	Object 的字节数。 类型: 字符串 父节点: ListBucketResult.Contents
StorageClass	Object 的存储类型, 目前只能是 “Standard” 类型: 字符串 父节点: ListBucketResult.Contents

细节分析:

- 1) Object 中用户自定义的 meta, 在 GetBucket 请求时不会返回。
- 2) List 的结果中会标识出哪些 keys 是 Object Group。
- 3) 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。
- 4) 如果没有访问该 Bucket 的权限, 返回 403 Forbidden 错误, 错误码: AccessDenied。
- 5) 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。
- 6) 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。
- 7) 若 prefix, marker, delimiter 参数不符合长度要求, 返回 400 Bad Request。错误码: InvalidArgument。
- 8) prefix, marker 用来实现分页显示效果, 参数的长度必须小于 1024 字节。
- 9) 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

请求示例：

```
GET / HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0ykboO4M=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 248c6483-2a95-622e-3022-eb65d8aad5f
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1866
Connection: close
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss.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.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:BC+oQIXVR2/ZghT7cGa0ykboO4M=
```

返回示例:

```
HTTP/1.1 200 OK
x-oss-request-id: 25cb535f-1feb-1e90-2f22-12176bcb563e
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1464
Connection: close
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss.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>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
    <Type>Normal</Type>
    <Size>344606</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>00220120222</ID>
      <DisplayName>user_example</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>fun/movie/007.avi</Key>
```

```

    <LastModified>2012-02-24T08:43:27.000Z</LastModified>
    <ETag>&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.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrn7xHk3sgysx7I8U9I9IY1vY=

```

返回示例:

```

HTTP/1.1 200 OK
x-oss-request-id: 0b05f9b1-539e-a858-0a81-9ca13d8a8011
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: close
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss.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>
```

相关阅读:

- [Put Bucket](#)
- [Put Object](#)
- [Get Object](#)

5.3.3 Get Bucket Acl

Get Bucket ACL 用来获取某个 Bucket 的访问权限。

请求语法：

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

响应元素(Response Elements)

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

细节分析：

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

请求示例：

```
GET /?acl HTTP/1.1
```

Host: oss-example.oss.aliyuncs.com

Date: Fri, 24 Feb 2012 04:11:23 GMT

Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAI4XZ+Wwlfnm0FmgbrQ0=

返回示例:

HTTP/1.1 200 OK

x-oss-request-id: 6f720c98-40fe-6de0-047b-e7fb08c4059b

Date: Fri, 24 Feb 2012 04:11:23 GMT

Content-Length: 253

Content-Type: application/xml

Connection: close

Server: AliyunOSS

<?xml version="1.0" ?>

<AccessControlPolicy>

<Owner>

<ID>00220120222</ID>

<DisplayName>user_example</DisplayName>

</Owner>

<AccessControlList>

<Grant>public-read</Grant>

</AccessControlList>

</AccessControlPolicy>

相关阅读:

- [Put Bucket](#)
- [Get Bucket Acl](#)

5.3.4 Put Bucket

PutBucket 用于创建 Bucket（不支持匿名访问）。

请求语法：

```
PUT / HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

细节分析：

- 1) 如果请求的 Bucket 已经存在，并且请求者是所有者，同样返回 200 OK 成功。
- 2) 如果请求的 Bucket 已经存在，但是不是请求者所拥有的，返回 409 Conflict。
错误码：BucketAlreadyExists。
- 3) 如果想创建的 Bucket 不符合命名规范，返回 400 Bad Request 消息。错误码：InvalidBucketName。
- 4) 如果用户发起 PUT Bucket 请求的时候，没有传入用户验证信息，返回 403 Forbidden 消息。错误码：AccessDenied。
- 5) 如果 PutBucket 的时候发现已经超过 bucket 最大创建数——10 时，返回 400 Bad Request 消息。错误码：TooManyBuckets。
- 6) 创建的 Bucket，如果没有指定访问权限，则默认使用“Private”权限。

请求示例：

```
PUT / HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Fri, 24 Feb 2012 03:15:40 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:77Dvh5wQgIjWjwO/KyRt8dOPfo8=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 7c9e8b71-3c6a-1b7d-2361-093f1af5f5e9
Date: Fri, 24 Feb 2012 03:15:40 GMT
Location: /oss-example
Content-Length: 0
Connection: close
```

相关阅读:

- [Get Bucket \(List Object\)](#)
- [Delete Bucket](#)
- [Put Object](#)

5.3.5 Put Bucket Acl

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

请求语法：

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

细节分析：

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

请求示例：

```
PUT / HTTP/1.1
x-oss-acl: public-read
Host: oss-example.oss.aliyuncs.com
Date: Fri, 24 Feb 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 248c6483-2a95-622e-3022-ebe65d8aad5f
Date: Fri, 24 Feb 2012 03:21:12 GMT
Content-Length: 0
Connection: close
```

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: 4e63c87a-71dc-87f7-11b5-583a600e0038
Date: Fri, 24 Feb 2012 03:55:00 GMT
Content-Length: 309
Content-Type: text/xml; charset=UTF-8
Connection: close
Server: AliyunOSS

<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
  <Code>
    InvalidArgument
  </Code>
  <Message>
    <Message>
      <ArgumentValue>
        error-acl
      </ArgumentValue>
      <ArgumentName>
        x-oss-acl
      </ArgumentName>
      <RequestId>
        4e63c87a-71dc-87f7-11b5-583a600e0038
      </RequestId>
      <HostId>
        oss.aliyuncs.com
      </HostId>
    </Error>
```

相关阅读：

- [Put Bucket](#)
- [Get Bucket Acl](#)

5.4 关于 Object 操作

5.4.1 Copy Object

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

请求语法:

```
PUT //ObjectName HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
x-oss-copy-source: /SourceBucketName/SourceObjectName
```

请求 Header:

名称	描述
x-oss-copy-source	复制源地址（必须有可读权限） 类型：字符串 默认值：无
x-oss-copy-source-if-match	如果源Object的ETAG值和用户提供的ETAG相等，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。 类型：字符串 默认值：无
x-oss-copy-source-if-none-match	如果源Object自从用户指定的时间以后就没有被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。 类型：字符串 默认值：无
x-oss-copy-source-if-unmodified-since	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误。 类型：字符串 默认值：无
x-oss-copy-source-if-modified-since	如果源Object自从用户指定的时间以后被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。 类型：字符串 默认值：无

x-oss-metadata-directive	<p>有效值为COPY和REPLACE。如果该值设为COPY，则新的Object的meta都从源Object复制过来；如果设为REPLACE，则忽视所有源Object的meta值，而采用用户这次请求中指定的meta值；其他值则返回400 HTTP错误码。注意该值为COPY时，源Object的x-oss-server-side-encryption的meta值不会进行拷贝。</p> <p>类型：字符串</p> <p>默认值：COPY</p> <p>有效值：COPY REPLACE</p>
x-oss-server-side-encryption	<p>指定oss创建目标object时的服务器端熵编码加密算法</p> <p>类型：字符串</p> <p>有效值：AES256</p>

响应元素(Response Elements):

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

细节分析:

- 1) 可以通过拷贝操作来实现修改已有 Object 的 meta 信息。
- 2) 如果拷贝操作的源 Object 地址和目标 Object 地址相同，则无论 x-oss-metadata-directive 为何值，都会直接替换源 Object 的 meta 信息。
- 3) OSS 支持拷贝操作的四个预判断 Header 任意个同时出现，相应逻辑参见 Get Object 操作的细节分析。
- 4) 拷贝操作需要请求者对源 Object 有读权限。
- 5) 拷贝操作的计费统计会对源 Object 所在的 Bucket 增加一次 Get 请求次数，并对目标 Object 所在的 Bucket 增加一次 Put 请求次数，以及相应的新增存储空间。
- 6) 拷贝操作涉及到的请求头，都是以“x-oss-”开头的，所以要加入签名字符串中。

-
- 7) 拷贝操作同样支持 Object Group 的拷贝。
 - 8) 若在拷贝操作中指定了 `x-oss-server-side-encryption` 请求头，并且请求值合法（为 AES256），则无论源 Object 是否进行过服务器端加密编码，拷贝之后的目标 Object 都会进行服务器端加密编码。并且拷贝操作的响应头中会包含 `x-oss-server-side-encryption`，值被设置成目标 Object 的加密算法。在这个目标 Object 被下载时，响应头中也会包含 `x-oss-server-side-encryption`，值被设置成该 Object 的加密算法；若拷贝操作中未指定 `x-oss-server-side-encryption` 请求头，则无论源 Object 是否进行过服务器端加密编码，拷贝之后的目标 Object 都是未进行过服务器端加密编码加密的数据。
 - 9) 拷贝操作中 `x-oss-metadata-directive` 请求头为 `COPY`（默认值）时，并不拷贝源 Object 的 `x-oss-server-side-encryption` 值，即目标 Object 是否进行服务器端加密编码只根据 `COPY` 操作是否指定了 `x-oss-server-side-encryption` 请求头来决定。
 - 10) Object group 不支持熵编码加密。源 Object 是 Object group 时，若在拷贝操作中指定了 `x-oss-server-side-encryption` 请求头，会返回 400 和错误提示：`InvalidArgument`。
 - 11) 若在拷贝操作中指定了 `x-oss-server-side-encryption` 请求头，并且请求值非 AES256，则返回 400 和相应的错误提示：`InvalidEncryptionAlgorithmError`。

请求示例：

```
PUT /copy_oss.jpg HTTP/1.1
Host: oss-example.oss.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: 3dfb2597-72a0-b3f7-320f-8b6627a96e68
Content-Type: application/xml
Content-Length: 193
Connection: close
Date: Fri, 24 Feb 2012 07:18:48 GMT
Server: AliyunOSS
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult xmlns="http://doc.oss.aliyuncs.com">
  <LastModified>Fri, 24 Feb 2012 07:18:48 GMT</LastModified>
  <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
</CopyObjectResult>
```

相关阅读:

- [Put Object](#)
- [Get Object](#)
- [Delete Object](#)

5.4.2 Delete Object

DeleteObject 用于删除某个 Object。

请求语法：

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

细节分析：

- 1) DeleteObject 要求对该 Object 要有写权限。
- 2) 如果要删除的 Object 不存在，OSS 也返回状态码 204（No Content）。
- 3) 如果 Bucket 不存在，返回 404 Not Found。

请求示例：

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

返回示例：

```
HTTP/1.1 204 NoContent
x-oss-request-id: 1a61ecd1-5de8-4e2e-20b5-c66e135bc379
Date: Fri, 24 Feb 2012 07:45:28 GMT
Content-Length: 0
Connection: close
Server: AliyunOSS
```

相关阅读：

- [Put Object](#)
- [Get Object](#)
- [Delete Multiple Objects](#)

5.4.3 Delete Multiple Objects

Delete Multiple Object 操作支持用户通过一个 HTTP 请求删除同一个 Bucket 中的多个 Object。Delete Multiple Object 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.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>
```

请求元素(Request Elements)：

名称	描述
Delete	保存Delete Multiple Object请求的容器。 类型：容器 子节点：一个或多个 <i>Object</i> 元素，可选的 <i>Quiet</i> 元素 父节点： None.
Key	被删除Object的名字。 类型：字符串 父节点： Object
Object	保存一个Object信息的容器。

	类型：容器 子节点：key 父节点：Delete.
Quiet	打开“安静”响应模式的开关。 类型：枚举字符串 有效值：true false 默认值：false 父节点：Delete

响应元素(Response Elements)：

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

细节分析：

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

请求示例 I:

```
POST /?delete HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length: 151
Content-MD5: fae2e404736a78a0067b62d80b1cc7d8
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: +z3gBfnFAxBcBDgx27Y/jEjbfu8=

<?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: close
Server: AliyunOSS

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

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

请求示例 II:

```
POST /?delete HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length: 151
Content-MD5: fae2e404736a78a0067b62d80b1cc7d8
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: WuV0Jks8RyGSNQrBca64kEEsJDs=

<?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: 501ad9bb-1383-771d-0ee9-59a810bd5fde
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length: 0
Connection: close
Server: AliyunOSS
```

相关阅读:

- [Delete Object](#)

5.4.4 Get Object

用于获取某个 Object，此操作要求用户对该 Object 有读权限。

请求语法：

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

请求参数：

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

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

请求 Header：

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

	类型：字符串 默认值：无
If-Modified-Since	如果指定的时间早于实际修改时间，则正常传送文件，并返回200 OK；否则返回304 not modified 类型：字符串 默认值：无
If-Unmodified-Since	如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误 类型：字符串 默认值：无
If-Match	如果传入期望的ETag和object的 ETag匹配，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误 类型：字符串 默认值：无
If-None-Match	如果传入的ETag值和Object的ETag不匹配，则正常传输文件,并返回200 OK；否则返回304 Not Modified 类型：字符串 默认值：无

细节分析：

- 1) GetObject 通过 range 参数可以支持断点续传,对于比较大的 Object 建议使用该功能。
- 2) 如果在请求头中使用 Range 参数；则返回消息中会包含整个文件的长度和此次返回的范围，例如：Content-Range: bytes 0-9/44，表示整个文件长度为 44，此次返回的范围为 0-9。如果不符合范围规范，则传送整个文件，并且不在结果中提及 Content-Range。
- 3) 如果“If-Modified-Since”元素中设定的时间不符合规范，直接返回文件，并返回 200 OK。
- 4) If-Modified-Since 和 If-Unmodified-Since 可以同时存在，If-Match 和 If-None-Match 也可以同时存在。
- 5) 如果包含 If-Unmodified-Since 并且不符合或者包含 If-Match 并且不符合，返回 412 precondition failed
- 6) 如果包含 If-Modified-Since 并且不符合或者包含 If-None-Match 并且不符合，返回 304 Not Modified
- 7) 如果文件不存在返回 404 Not Found 错误。错误码：NoSuchKey。

-
- 8) OSS 不支持在匿名访问的 GET 请求中, 通过请求参数来自定义返回请求的 header。
 - 9) 在自定义 OSS 返回请求中的一些 Header 时, 只有请求处理成功 (即返回码为 200 时), OSS 才会将请求的 header 设置成用户 GET 请求参数中指定的值。
 - 10) 若该 Object 为进行服务器端熵编码加密存储的, 则在 GET 请求时会自动解密返回给用户, 并且在响应头中, 会返回 x-oss-server-side-encryption, 其值表明该 Object 的服务器端加密算法。

请求示例:

```
GET /oss.jpg HTTP/1.1
Host: oss-example.oss.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
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpeg
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.aliyuncs.com
Date: Fri, 28 Feb 2012 05:38:42 GMT
Range: bytes=100-900
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM=
```

返回示例:

```
HTTP/1.1 206 Partial Content
x-oss-request-id: 28f6508f-15ea-8224-234e-c0ce40734b89
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（URL 签名方式）：

```
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.aliyuncs.com:
Date: Fri, 24 Feb 2012 06:09:48 GMT
```

返回示例：

```
HTTP/1.1 200 OK
x-oss-request-id: 1144d124-055c-4052-2c65-a1e3439d41c1
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: close
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]
```

相关阅读：

- [Get Object](#)
- [Delete Object](#)

5.4.5 Head Object

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

请求语法：

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

请求 Header：

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

细节分析：

- 1) 不论正常返回 200 OK 还是非正常返回，Head Object 都不返回消息体。
- 2) HeadObject 支持在头中设定 If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match 的查询条件。具体规则请参见 GetObject 中对应的选项。如果没有修改，返回 304 Not Modified。
- 3) 如果用户在 PutObject 的时候传入以 x-oss-meta-为开头的 user meta，比如 x-oss-meta-location，返回消息时，返回这些 user meta。

-
- 4) 如果文件不存在返回 404 Not Found 错误。
 - 5) 若该 Object 为进行服务器端熵编码加密存储的，则在 Head 请求响应头中，会返回 x-oss-server-side-encryption，其值表明该 Object 的服务器端加密算法。

请求示例：

```
HEAD /oss.jpg HTTP/1.1
Host: oss-example.oss.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: 06d4be30-2216-9264-757a-8f8b19b254bb
Date: Fri, 24 Feb 2012 07:32:52 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
Content-Type: image/jpeg
Connection: close
Server: AliyunOSS
```

相关阅读：

- [Put Object](#)
- [Get Object](#)

5.4.6 Put Object

Put Object 用于上传文件。

请求语法：

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

请求 Header：

名称	描述
Cache-Control	指定该Object被下载时的网页的缓存行为；更详细描述请参照RFC2616。 类型：字符串 默认值：无
Content-Disposition	指定该Object被下载时的名称；更详细描述请参照RFC2616。 类型：字符串 默认值：无
Content-Encoding	指定该Object被下载时的内容编码格式；更详细描述请参照RFC2616。 类型：字符串 默认值：无
Expires	过期时间（milliseconds）；更详细描述请参照RFC2616。 类型：整数 默认值：无
x-oss-server-side-encryption	指定oss创建object时的服务器端加密编码算法。 类型：字符串 合法值：AES256

细节分析：

- 1) Put Object 请求处理成功后，OSS 会将收到文件的 MD5 值放在返回给用户的请求头“ETag”中，以使用户检查 OSS 上的数据和要上传的数据内容一致。
- 2) 如果请求头中的“Content-Length”值小于实际请求体（body）中传输的数据长度，OSS 仍将成功创建文件；但 Object 大小只等于“Content-Length”中定义的大小，其他数据将被丢弃。

-
- 3) 如果试图添加的 Object 的同名文件已经存在，并且有访问权限。新添加的文件将覆盖原来的文件，成功返回 200 OK。
 - 4) 如果在 PutObject 的时候，携带以 x-oss-meta-为前缀的参数，则视为 user meta，比如 x-oss-meta-location。一个 Object 可以有多个类似的参数，但所有的 user meta 总大小不能超过 2k。
 - 5) 如果 Head 中没有加入 Content length 参数，会返回 411 Length Required 错误。错误码：MissingContentLength。
 - 6) 如果设定了长度，但是没有发送消息 Body，或者发送的 body 大小小于给定大小，服务器会一直等待，直到 time out，返回 400 Bad Request 消息。错误码：RequestTimeout。此时 OSS 上的这个文件内容是用户已经上传完的数据。
 - 7) 如果试图添加的 Object 所在的 Bucket 不存在，返回 404 Not Found 错误。错误码：NoSuchBucket。
 - 8) 如果试图添加的 Object 所在的 Bucket 没有访问权限，返回 403 Forbidden 错误。错误码：AccessDenied。
 - 9) 如果添加文件长度超过 5G，返回错误消息 400 Bad Request。错误码：InvalidArgument。
 - 10) 如果传入的 Object key 长度大于 1023，返回 400 Bad Request。错误码：InvalidObjectName。
 - 11) PUT 一个 Object 的时候，OSS 支持 4 个 HTTP RFC 2616 协议规定的 Header 字段：Cache-Control、Expires、Content-Encoding、Content-Disposition。如果上传 Object 时设置了这些 Header，则这个 Object 被下载时，相应的 Header 值会被自动设置成上传时的值。
 - 12) 如果上传 Object 时指定了 x-oss-server-side-encryption Header，则必须设置其值为 AES256，否则会返回 400 和相应错误提示：InvalidEncryptionAlgorithmError。指定该 Header 后，在响应头中也会返回该 Header，OSS 会对上传的 Object 进行加密编码存储，当这个 Object 被下载时，响应头中会包含 x-oss-server-side-encryption，值被设置成该 Object 的加密算法。

请求示例:

```
PUT /oss.jpg HTTP/1.1
Host: oss-example.oss.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
x-oss-request-id: 61d2042d-1b68-6708-5906-33d81921362e
Date: Fri, 24 Feb 2012 06:03:28 GMT
ETag: 5B3C1A2E053D763E1B002CC607C5A0FE
Connection: close
Content-Length: 0
Server: AliyunOSS
```

相关阅读:

- [Get Object](#)
- [Delete Object](#)

5.5 关于 Multipart Upload 的操作

除了通过 PUT Object 接口上传文件到 OSS 以外，OSS 还提供了另外一种上传模式——Multipart Upload。用户可以在如下的应用场景内（但不仅限于此），使用 Multipart Upload 上传模式，如：

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

5.5.1 Initiate Multipart Upload

使用 Multipart Upload 模式传输数据前，必须先调用该接口来通知 OSS 初始化一个 Multipart Upload 事件。该接口会返回一个 OSS 服务器创建的全局唯一的 Upload ID，用于标识本次 Multipart Upload 事件。用户可以根据这个 ID 来发起相关的操作，如中止 Multipart Upload、查询 Multipart Upload 等。

请求语法：

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

请求 Header：

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

	类型：字符串 默认值：无
Expires	过期时间（milliseconds）；更详细描述请参照RFC2616。 类型：整数 默认值：无
x-oss-server-side-encryption	指定上传该Object每个part时使用的服务器端加密编码算法，OSS会对上传的每个part采用服务器端加密编码进行存储。 类型：字符串 合法值：AES256

响应元素(Response Elements)：

名称	描述
Bucket	初始化一个Multipart Upload事件的Bucket名称。 类型：字符串 父节点：InitiateMultipartUploadResult
InitiateMultipartUploadResult	保存Initiate Multipart Upload请求结果的容器。 类型：容器 子节点：Bucket, Key, UploadId 父节点：None
Key	初始化一个Multipart Upload事件的Object名称。 类型：字符串 父节点：InitiateMultipartUploadResult
UploadId	唯一标示此次Multipart Upload事件的ID。 类型：字符串 父节点：InitiateMultipartUploadResult

细节分析：

- 1) 该操作计算认证签名的时候，需要加“?uploads”到 CanonicalizedResource 中。
- 2) 初始化 Multipart Upload 请求，支持如下标准的 HTTP 请求头：Cache-Control, Content-Disposition, Content-Encoding, Content-Type, Expires, 以及“x-oss-meta-”开头的用户自定义 Headers。具体含义请参见 PUT Object 接口。
- 3) 初始化 Multipart Upload 请求，并不会影响已经存在的同名 object。
- 4) 服务器收到初始化 Multipart Upload 请求后，会返回一个 XML 格式的请求体。该请求体内有三个元素：Bucket, Key 和 UploadID。请记录下其中的

UploadID，以用于后续的 Multipart 相关操作。

- 5) 初始化 Multipart Upload 请求时，若设置了 `x-oss-server-side-encryption` Header，则在响应头中会返回该 Header，并且在上传的每个 part 时，服务端会自动对每个 part 进行熵编码加密存储，目前 OSS 服务器端只支持 256 位高级加密标准（AES256），指定其他值会返回 400 和相应的错误提示：`InvalidEncryptionAlgorithmError`；在上传每个 part 时不必再添加 `x-oss-server-side-encryption` 请求头，若指定该请求头则 OSS 会返回 400 和相应的错误提示：`InvalidArgument`。

请求示例：

```
POST /multipart.data?uploads HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:/cluRFtRwMTZpC2hTj4F67AGdM4=
```

返回示例：

```
HTTP/1.1 200 OK
Content-Length: 230
Server: AliyunOSS
Connection: close
x-oss-request-id: 42c25703-7503-fbd8-670a-bda01eaec618
Date: Wed, 22 Feb 2012 08:32:21 GMT
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns="http://doc.oss.aliyuncs.com">
  <Bucket> multipart_upload</Bucket>
  <Key>multipart.data</Key>
  <UploadId>0004B9894A22E5B1888A1E29F8236E2D</UploadId>
</InitiateMultipartUploadResult>
```

相关阅读：

- [List Multipart Uploads](#)
- [Abort Multipart Upload](#)
- [Complete Multipart Upload](#)

5.5.2 Upload Part

在初始化一个 Multipart Upload 之后,可以根据指定的 Object 名和 Upload ID 来分块(Part)上传数据。每一个上传的 Part 都有一个标识它的号码(part number, 范围是 1~10,000)。对于同一个 Upload ID, 该号码不但唯一标识这一块数据, 也标识了这块数据在整个文件内的相对位置。如果你用同一个 part 号码, 上传了新的数据, 那么 OSS 上已有的这个号码的 Part 数据将被覆盖。除了最后一块 Part 以外, 其他的 part 最小为 5MB; 最后一块 Part 没有大小限制。

请求语法:

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

细节分析:

- 1) 调用该接口上传 Part 数据前, 必须调用 Initiate Multipart Upload 接口, 获取一个 OSS 服务器颁发的 Upload ID。
- 2) Multipart Upload 要求除最后一个 Part 以外, 其他的 Part 大小都要大于 5MB。但是 Upload Part 接口并不会立即校验上传 Part 的大小 (因为不知道是否为最后一块); 只有当 Complete Multipart Upload 的时候才会校验。
- 3) OSS 会将服务器端收到 Part 数据的 MD5 值放在 ETag 头内返回给用户。为了保证数据在网络传输过程中不出现错误, 强烈推荐用户在收到 OSS 的返回请求后, 用该 MD5 值验证上传数据的正确性。
- 4) Part 号码的范围是 1~10000。如果超出这个范围, OSS 将返回 InvalidArgument 的错误码。
- 5) 若调用 Initiate Multipart Upload 接口时, 指定了 x-oss-server-side-encryption 请求头, 则会对上传的 Part 进行加密编码, 并在 Upload Part 响应头中返回 x-oss-server-side-encryption 头, 其值表明该 Part 的服务器端加密算法, 具体见 Initiate Multipart Upload 接口。

请求示例:

```
PUT /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E36 HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53ofjbyc:J/ICfXEvPmmSW86bBAfMmUmWjI=

[6291456 bytes data]
```

返回示例:

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

相关阅读:

- [Initiate Multipart Upload](#)
- [List Parts](#)
- [Complete Multipart Upload](#)

5.5.3 Complete Multipart Upload

在将所有数据 Part 都上传完成后，必须调用 Complete Multipart Upload API 来完成整个文件的 Multipart Upload。在执行该操作时，用户必须提供所有有效的数据 Part 的列表(包括 part 号码和 ETag);OSS 收到用户提交的 Part 列表后，会逐一验证每个数据 Part 的有效性。当所有的数据 Part 验证通过后，OSS 将把这些数据 part 组合成一个完整的 Object。

请求语法:

```
POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Content-Length: Size
Authorization: Signature

<CompleteMultipartUpload>
  <Part>
    <PartNumber>PartNumber</PartNumber>
    <ETag>ETag</ETag>
  </Part>
  ...
</CompleteMultipartUpload>
```

请求元素(Request Elements):

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

	父节点: Part
--	-----------

响应元素(Response Elements):

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

细节分析:

- 1) Complete Multipart Upload 时, 会确认除最后一块以外所有块的大小都大于 5MB, 并检查用户提交的 Partlist 中的每一个 Part 号码和 Etag。所以在上传 Part 时, 客户端除了需要记录 Part 号码外, 还需要记录每次上传 Part 成功后, 服务器返回的 ETag 值。
- 2) OSS 处理 Complete Multipart Upload 请求时, 会持续一定的时间。在这段时间内, 如果客户端和 OSS 之间的链接断掉, OSS 仍会继续将请求做完。
- 3) 用户提交的 Part List 中, Part 号码可以是不连续的。例如第一块的 Part 号码是 1; 第二块的 Part 号码是 5。
- 4) OSS 处理 Complete Multipart Upload 请求成功后, 该 Upload ID 就会变成无效。
- 5) 同一个 Object 可以同时拥有不同的 Upload Id, 当 Complete 一个 Upload ID

后，该 Object 的其他 Upload ID 不受影响。

- 6) 若调用 Initiate Multipart Upload 接口时，指定了 x-oss-server-side-encryption 请求头，则在 Complete Multipart Upload 的响应头中，会返回 x-oss-server-side-encryption，其值表明该 Object 的服务器端加密算法。

请求示例：

```
POST /multipart.data? uploadId=0004B9B2D2F7815C432C9057C03134D4 HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 1056
Date: Fri, 24 Feb 2012 10:19:18 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:8VwFhFUWmVecK6jQlHIXMK/zMT0=

<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
  </Part>
  <Part>
    <PartNumber>5</PartNumber>
    <ETag>"8EFDA8BE206636A695359836FE0A0E0A"</ETag>
  </Part>
  <Part>
    <PartNumber>8</PartNumber>
    <ETag>"8C315065167132444177411FDA149B92"</ETag>
  </Part>
</CompleteMultipartUpload>
```

返回示例：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Content-Length: 329
Content-Type: Application/xml
Connection: close
x-oss-request-id: 594f0751-3b1e-168f-4501-4ac71d217d6e
Date: Fri, 24 Feb 2012 10:19:18 GMT

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns="http://doc.oss.aliyuncs.com">
  <Location>http://storage.aliyun-inc.com/multipart_upload/multipart.data</Location>
  <Bucket>multipart_upload</Bucket>
  <Key>multipart.data</Key>
```

```
<ETag> &quot;B864DB6A936D376F9F8D3ED3BBE540DD-3&quot; </ETag>
</CompleteMultipartUploadResult>
```

相关阅读:

- [**Initiate Multipart Upload**](#)
- [**List Parts**](#)

5.5.4 Abort Multipart Upload

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

请求语法：

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

细节分析：

- 1) 中止一个 Multipart Upload 事件时，如果其所属的某些 Part 仍然在上传，那么这次中止操作将无法删除这些 Part。所以如果存在并发访问的情况，为了彻底释放 OSS 上的空间，需要调用几次 Abort Multipart Upload 接口。
- 2) 如果输入的 Upload Id 不存在，OSS 会返回 404 错误，错误码为：NoSuchUpload。

请求示例：

```
Delete /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/ICfXEvPmmSW86bBAfMmUmWjI=
```

返回示例：

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

相关阅读：

- [Initiate Multipart Upload](#)
- [List Multipart Uploads](#)

5.5.5 List Multipart Uploads

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

请求语法：

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

请求参数：

名称	描述
delimiter	是一个用于对 Object 名字进行分组的字符。所有名字包含指定的前缀且第一次出现 delimiter 字符之间的 object 作为一组元素——CommonPrefixes。 类型：字符串
max-uploads	限定此次返回 Multipart Uploads 事件的最大数目，如果不设定，默认为 1000，max-keys 取值不能大于 1000。 类型：字符串
key-marker	与 upload-id-marker 参数一同使用来指定返回结果的起始位置。 <ul style="list-style-type: none">● 如果 upload-id-marker 参数未设置，查询结果中包含：所有 Object 名字的字典序大于 key-marker 参数值的 Multipart 事件。● 如果 upload-id-marker 参数被设置，查询结果中包含：所有 Object 名字的字典序大于 key-marker 参数值的 Multipart 事件和 Object 名字等于 key-marker 参数值，但是 Upload ID 比 upload-id-marker 参数值大的 Multipart Uploads 事件。 类型：字符串
prefix	限定返回的 object key 必须以 prefix 作为前缀。注意使用 prefix 查询时，返回的 key 中仍会包含 prefix。 类型：字符串
upload-id-marker	与 key-marker 参数一同使用来指定返回结果的起始位置。 <ul style="list-style-type: none">● 如果 key-marker 参数未设置，则 OSS 忽略 upload-id-marker 参数。● 如果 key-marker 参数被设置，查询结果中包含：所有 Object 名字的字典序大于 key-marker 参数值的 Multipart 事件和 Object 名字等于 key-marker 参数值，但是 Upload

	ID比upload-id-marker参数值大的Multipart Uploads事件。 类型：字符串
--	--

响应元素(Response Elements)：

名称	描述
ListMultipartUploadsResult	保存List Multipart Upload请求结果的容器。 类型：容器 子节点： <i>Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MaxUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload</i> 父节点：None
Bucket	Bucket名称。 类型：字符串 父节点：ListMultipartUploadsResult
KeyMarker	列表的起始Object位置。 类型：字符串 父节点：ListMultipartUploadsResult
UploadIdMarker	列表的起始UploadID位置。 类型：字符串 父节点：ListMultipartUploadsResult
NextKeyMarker	如果本次没有返回全部结果，响应请求中将包含NextKeyMarker元素，用于标明接下来请求的KeyMarker值。 类型：字符串 父节点：ListMultipartUploadsResult
NextUploadMarker	如果本次没有返回全部结果，响应请求中将包含NextUploadMarker元素，用于标明接下来请求的UploadMarker值。 类型：字符串 父节点：ListMultipartUploadsResult
MaxUploads	返回的最大Upload数目。 类型：整数 父节点：ListMultipartUploadsResult
IsTruncated	标明是否本次返回的Multipart Upload结果列表被截断。“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。 类型：枚举字符串 有效值：false true 默认值：false 父节点：ListMultipartUploadsResult
Upload	保存Multipart Upload事件信息的容器。 类型：容器 子节点： <i>Key, UploadId, Initiated</i> 父节点：ListMultipartUploadsResult
Key	初始化Multipart Upload事件的Object名字。

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

细节分析：

- 1) “max-uploads” 参数最大值为 1000。
- 2) 在 OSS 的返回结果首先按照 Object 名字字典序升序排列;对于同一个 Object, 则按照时间序, 升序排列。
- 3) 可以灵活地使用 prefix 参数对 bucket 内的 object 进行分组管理（类似与文件夹的功能）。
- 4) List Multipart Uploads 请求支持5种请求参数：prefix,marker,delimiter, upload-id-marker 和 max-keys。通过这些参数的组合，可以设定查询 Multipart Uploads 事件的规则，获得期望的查询结果。

请求示例：

```
Get /?uploads HTTP/1.1
Host:oss-example.oss.aliyuncs.com
Date: Thu, 23 Feb 2012 06:14:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:JX75CtQqsmBBz+dcivn7kwBMvOY=
```

返回示例：

```
HTTP/1.1 200
Server: AliyunOSS
Connection: close
Content-length: 1839
Content-type: application/xml
x-oss-request-id: 58a41847-3d93-1905-20db-ba6f561ce67a
Date: Thu, 23 Feb 2012 06:14:27 GMT

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns="http://doc.oss.aliyuncs.com">
  <Bucket>oss-example</Bucket>
  <KeyMarker></KeyMarker>
```

```
<UploadIdMarker></UploadIdMarker>
<NextKeyMarker>oss.avi</NextKeyMarker>
<NextUploadIdMarker>0004B99B8E707874FC2D692FA5D77D3F</NextUploadIdMarker>
<Delimiter></Delimiter>
<Prefix></Prefix>
<MaxUploads>1000</MaxUploads>
<IsTruncated>false</IsTruncated>
<Upload>
  <Key>multipart.data</Key>
  <UploadId>0004B999EF518A1FE585B0C9360DC4C8</UploadId>
  <Initiated>2012-02-23T04:18:23.000Z</Initiated>
</Upload>
<Upload>
  <Key>multipart.data</Key>
  <UploadId>0004B999EF5A239BB9138C6227D69F95</UploadId>
  <Initiated>2012-02-23T04:18:23.000Z</Initiated>
</Upload>
<Upload>
  <Key>oss.avi</Key>
  <UploadId>0004B99B8E707874FC2D692FA5D77D3F</UploadId>
  <Initiated>2012-02-23T06:14:27.000Z</Initiated>
</Upload>
</ListMultipartUploadsResult>
```

相关阅读:

- [Initiate Multipart Upload](#)
- [List Parts](#)

5.5.6 List Parts

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

请求语法：

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

请求参数(Request Parameters)：

名称	描述
uploadId	Multipart Upload事件的ID。 类型：字符串 默认值：无
max-parts	规定在OSS响应中的最大Part数目。 类型：整数 默认值：1,000
part-number-marker	指定List的起始位置，只有Part Number数目大于该参数的Part会被列出。 类型：整数 默认值：无

响应元素(Response Elements)：

名称	描述
ListPartsResult	保存List Part请求结果的容器。 类型：容器 子节点：Bucket, Key, UploadId, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part 父节点：无
Bucket	Bucket名称。 类型：字符串 父节点：ListPartsResult
Key	Object名称。 类型：字符串 父节点：ListPartsResult
UploadId	Upload事件ID。 类型：字符串 父节点：ListPartsResult
PartNumberMarker	本次List结果的Part Number起始位置。

	类型：整数 父节点：ListPartsResult
NextPartNumberMarker	如果本次没有返回全部结果，响应请求中将包含NextPartNumberMarker元素，用于标明接下来请求的PartNumberMarker值。 类型：整数 父节点：ListPartsResult
MaxParts	返回请求中最大的Part数目。 类型：整数 父节点：ListPartsResult
IsTruncated	标明是否本次返回的List Part结果列表被截断。“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。 类型：枚举字符串 有效值：true false 父节点：ListPartsResult
Part	保存Part信息的容器。 类型：字符串 子节点：PartNumber, LastModified, ETag, Size 父节点：ListPartsResult
PartNumber	标示Part的数字。 类型：整数 父节点：ListPartsResult.Part
LastModified	Part上传的时间。 类型：日期 父节点：ListPartsResult.part
ETag	已上传Part内容的ETag。 类型：字符串 父节点：ListPartsResult.Part
Size	已上传Part大小。 类型：整数 父节点：ListPartsResult.Part

细节分析：

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

请求示例：

```
Get /multipart.data?uploadId=0004B999EF5A239BB9138C6227D69F95 HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Thu, 23 Feb 2012 07:13:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:4qOnUMc9UQWqkz8wDqD3lIsa9P8=
```

返回示例:

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

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns="http://doc.oss.aliyuncs.com">
  <Bucket>multipart_upload</Bucket>
  <Key>multipart.data</Key>
  <UploadId>0004B999EF5A239BB9138C6227D69F95</UploadId>
  <NextPartNumberMarker>5</NextPartNumberMarker>
  <MaxParts>1000</MaxParts>
  <IsTruncated>>false</IsTruncated>
  <Part>
    <PartNumber>1</PartNumber>
    <LastModified>2012-02-23T07:01:34.000Z</LastModified>
    <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
    <Size>6291456</Size>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2012-02-23T07:01:12.000Z</LastModified>
    <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
    <Size>6291456</Size>
  </Part>
  <Part>
    <PartNumber>5</PartNumber>
    <LastModified>2012-02-23T07:02:03.000Z</LastModified>
    <ETag>"7265F4D211B56873A381D321F586E4A9"</ETag>
    <Size>1024</Size>
  </Part>
</ListPartsResult>
```

相关阅读:

-
- [Initiate Multipart Upload](#)
 - [List Multipart Uploads](#)

6. OSS 的错误响应

当用户访问 OSS 出现错误时, OSS 会返回给用户相应的错误码和错误信息, 便于用户定位问题, 并做出适当的处理。

6.1 OSS 的错误响应格式

当用户访问 OSS 出错时, OSS 会返回给用户一个合适的 3xx, 4xx 或者 5xx 的 HTTP 状态码; 以及一个 application/xml 格式的消息体。

错误响应的消息体例子:

```
<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
  <Code>
    AccessDenied
  </Code>
  <Message>
    Query-string authentication requires the Signature, Expires and OSSAccessKeyId parameters
  </Message>
  <RequestId>
    1D842BC5425544BB
  </RequestId>
  <HostId>
    oss.aliyuncs.com
  </HostId>
</Error>
```

所有错误的消息体中都包括以下几个元素:

- **Code:** OSS 返回给用户的错误码。
- **Message:** OSS 给出的详细错误信息。
- **RequestId:** 用于唯一标识该次请求的 UUID; 当你无法解决问题时, 可以凭这个 RequestId 来请求 OSS 开发工程师的帮助。
- **HostId:** 用于标识访问的 OSS 集群 (目前统一为 oss.aliyuncs.com)

其他特殊的错误信息元素请参照每个请求的具体介绍。

6.2 OSS 的错误码

OSS 的错误码列表如下：

错误码	描述	HTTP 状态码
AccessDenied	拒绝访问	403
BucketAlreadyExists	Bucket 已经存在	409
BucketNotEmpty	Bucket 不为空	409
EntityTooLarge	实体过大	400
EntityTooSmall	实体过小	400
FileGroupTooLarge	文件组过大	400
FilePartNotExist	文件 Part 不存在	400
FilePartStale	文件 Part 过时	400
InvalidArgument	参数格式错误	400
InvalidAccessKeyId	Access Key ID 不存在	403
InvalidBucketName	无效的 Bucket 名字	400
InvalidDigest	无效的摘要	400
InvalidEncryptionAlgorithmError	指定的加密编码算法错误	400
InvalidObjectName	无效的 Object 名字	400
InvalidPart	无效的 Part	400
InvalidPartOrder	无效的 part 顺序	400
InvalidTargetBucketForLogging	Logging 操作中有无效的目标 bucket	400
InternalError	OSS 内部发生错误	500
MalformedXML	XML 格式非法	400
MethodNotAllowed	不支持的方法	405
MissingArgument	缺少参数	411
MissingContentLength	缺少内容长度	411
NoSuchBucket	Bucket 不存在	404

NoSuchKey	文件不存在	404
NoSuchUpload	Multipart Upload ID 不存在	404
NotImplemented	无法处理的方法	501
PreconditionFailed	预处理错误	412
RequestTimeTooSkewed	发起请求的时间和服务器时间超出 15 分钟	403
RequestTimeout	请求超时	400
SignatureDoesNotMatch	签名错误	403
TooManyBuckets	用户的 Bucket 数目超过限制	400

6.3 OSS 不支持分块传输编码

HTTP 协议有一种分块传输编码的机制(Chunked Transfer Encoding)，即一个 HTTP 消息可以分成多个部分进行传输。它对 HTTP 请求和 HTTP 响应都是适用的。出于安全的考虑，OSS 不支持这种分块传输编码方式。如果 OSS 收到这种分块传输编码方式的请求，回直接返回 HTTP 错误码：411。该错误码在 HTTP 协议中的含义是：“The request must be chunked or have a content length”。

6.4 OSS 不支持的操作

如果试图以OSS不支持的操作来访问某个资源,返回 405 Method Not Allowed 错误。

错误请求示例:

```
abc / HTTP/1.1
Host:oss-example. oss.aliyuncs.com
Date: date
Authorization: signatureValue
```

返回示例:

```
x-oss-request-id: 2403382433A2EDA8
Allow: GET, DELETE, HEAD, PUT
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 31 Mar 2011 10:01:52 GMT
Server: AliyunOSS

<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
  <Code>
    MethodNotAllowed
  </Code>
  <Message>
    The specified method is not allowed against this resource.
  </Message>
  <ResourceType>
    BUCKET
  </ResourceType>
  <Method>
    abc
  </Method>
  <RequestId>
    2403382433A2EDA8
  </RequestId>
  <HostId>
    oss.aliyuncs.com
  </HostId>
</Error>
```

▲ 注意：如果访问的资源是 `/bucket/`，`ResourceType` 应该是 `bucket`，如果访问的资源是 `/bucket/object`，`ResourceType` 应该是 `object`。

6.5 OSS 操作支持但参数不支持的操作

如果在 OSS 合法的操作中，添加了 OSS 不支持的参数（例如在 PUT 的时候，加入 If-Modified-Since 参数），OSS 会返回 501 Not Implemented 错误

错误请求示例：

```
PUT /my-image.jpg HTTP/1.1
Host:oss-example.oss.aliyuncs.com
Date: Wed, 28 May 2011 22:32:00 GMT
If-Modified-Since: Wed, 06 Apr 2011 10:02:46 GMT
```

返回示例：

```
501 Not Implemented
x-oss-request-id: 77E534EBF90372BE
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 28 Apr 2011 08:03:07 GMT
Connection: close
Server: AliyunOSS

<?xml version="1.0" ?>
<Error xmlns="http://doc.oss.aliyuncs.com">
  <Code>
    NotImplemented
  </Code>
  <Message>
    A header you provided implies functionality that is not implemented.
  </Message>
  <Header>
    If-Modified-Since
  </Header>
  <RequestId>
    77E534EBF90372BE
  </RequestId>
  <HostId>
    oss.aliyuncs.com
  </HostId>
</Error>
```