# 目 录

# 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、任何互联网设备上进行上传和下载数据，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/

# 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。

n Object命名规范

- Ø 使用UTF-8编码
- Ø 长度必须在1-1023字节之间
- Ø 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。

n Bucket命名规范

- Ø 只能包括小写字母，数字，短横线（-）
- Ø 必须以小写字母或者数字开头
- Ø 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到多个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//image/aliyun-logo.png*

构成规则的示意图如下：

用户可以直接该URL链接放入HTML中使用：

```
<img src="http://oss-example.oss.aliyuncs.com/aliyun-logo.png"/>
```

## 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 自定义域名绑定（CNAME）

OSS支持用户将自定义的域名绑定在属于自己的bucket上面，这个操作必须通过OSS控制台（http://oss.aliyun.com）来实现。按照中国《互联网管理条例》的要求，所有需要开通这项功能的用户，必须提供阿里云备案号，域名持有者身份证等有效资料，经由阿里云审批通过后才可以使用。在开通CNAME功能后，OSS将自动处理对该域名的访问请求。

**CNAME应用场景例子：**

Ø 用户A拥有一个域名为abc.com的网站；这个网站的所有图片存储在img.abc.com这个子域名下；
Ø 为了应对日益增长的图片流量压力，用户A在OSS上创建了一个名为abc-img的bucket，并将所有图片存在OSS上；
Ø 通过OSS控制台，提交将img.abc.com CNAME成 abc-img.oss.aliyuncs.com的申请，并提供相应的材料
Ø 通过阿里云审核后，在自己的域名服务器上，添加一条CNAME规则，将img.abc.com映射成abc-img.oss.aliyuncs.com，这样所有对img.abc.com的访问都将变成访问abc-img这个bucket。例如：一个对http://img.abc.com/logo.png的访问，实际上访问的是http://abc-img.oss.aliyuncs.com/logo.png。

## 3.5 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 |
| 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。

# 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。

n active表明用户可以用此ID对签名验证请求

n 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))
```

n CONTENT-MD5表示请求内容数据的MD5值

n CONTENT-TYPE表示请求内容的类型

n DATE表示此次操作的时间，且必须为HTTP1.1中支持的GMT格式

n CanonicalizedOSSHeaders表示 http中的object meta组合

n 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header[4]，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Access Key ID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”63mwfl+zYIOG6k95yxbgMruQ6QI=”，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: 63mwfl+zYIOG6k95yxbgMruQ6QI=
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
```

**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：

- Ø head的名字需要变成小写。
- Ø head按字典序自小到大排序。
- Ø 分割head name和value的冒号前后不能有空格。
- Ø 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Access Key ID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：

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

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Access Key ID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" encoding="UTF-8"?>
<Error xmlns="http://doc.oss.aliyuncs.com">
<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\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曝露在互联网上，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：

n `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。

n `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）。

n `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:**

Ø Put Bucket Acl

Ø Get Bucket Acl

Ø

# 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。<br>类型：字符串<br>默认值：无<br>使用场景：非匿名请求 |
| Content-Length | RFC 2616中定义的HTTP请求内容长度。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求 |
| Content-Type | RFC 2616中定义的HTTP请求内容类型。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求 |
| Date | HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT<br>类型：字符串<br>默认值：无 |
| Host | 访问Host值，格式为：<*bucketname*>*.oss.aliyuncs.com*。<br>类型：字符串<br>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。<br>类型：字符串<br>默认值：无<br>使用场景：需要向OSS提交数据的请求 |
| Connection | 标明客户端和OSS服务器之间的链接状态。<br>类型：枚举<br>有效值：open \| close<br>默认值：无 |
| Date | HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT<br>类型：字符串<br>默认值：无 |
| ETag | ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。<br>类型：字符串<br>默认值：无 |
| Server | 生成Response的服务器。<br>类型：字符串<br>默认值：AliyunOSS |
| x-oss-request-id | x-oss-request-id是由Aliyun OSS创建，并唯一标识这个response的UUID。如果在使用OSS服务时遇到问题，可以凭借该字段联系OSS工作人员，快速定位问题。<br>类型：字符串<br>默认值：无 |

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

Ø Get Bucket (List Object)

Ø 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 qn6qrrqxo2oawuk53otfjbyc: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：**

**Ø Put Bucket**

**Ø 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。<br>数据类型：字符串<br>默认值：无 |
| marker | 设定结果从marker之后按字母排序的第一个开始返回。<br>数据类型：字符串<br>默认值：无 |
| max-keys | 限定此次返回object的最大数，如果不设定，默认为1000，max-keys取值不能大于1000。<br>数据类型：字符串<br>默认值：100 |
| prefix | 限定返回的object key必须以prefix作为前缀。注意使用prefix查询时，返回的key中仍会包含prefix。<br>数据类型：字符串<br>默认值：无 |

**响应元素(Response Elements)**

| 名称 | 描述 |
| --- | --- |
| Contents | 保存每个返回Object meta的容器。<br>类型：容器<br>父节点：ListBucketResult |
| CommonPrefixes | 如果请求中指定了delimiter参数，则在OSS返回的响应中包含CommonPrefixes元素。该元素标明那些以delimiter结尾，并有共同前缀的object名称的集合。<br>类型：字符串<br>父节点：ListBucketResult |
| Delimiter | 是一个用于对Object名字进行分组的字符。所有名字包含指定的前缀且第一次出现delimiter字符之间的object作为一组元素——CommonPrefixes。<br>类型：字符串<br>父节点：ListBucketResult |
| DisplayName | Object 拥有者的名字。<br>类型：字符串<br>父节点：ListBucketResult.Contents.Owner |
| ETag | ETag (entity tag) 在每个Object生成的时候被创建，用于标示一个Object的内容。对于Put Object请求创建的Object，ETag值是其内容的MD5值；对于其他方式创建的Object，ETag值是其内容的UUID。ETag值可以用于检查Object内容是否发生变化。<br>类型：字符串<br>父节点：ListBucketResult.Contents |
| ID | Bucket拥有者的用户ID。<br>类型：字符串<br>父节点：ListBucketResult.Contents.Owner |

| | |
|---|---|
| IsTruncated | 指明是否所有的结果都已经返回；“true”标示所有结果都已经返回；“false”表示本次请求只返回了部分结果。<br>类型：枚举字符串<br>有效值：true \| false<br>父节点：ListBucketResult |
| Key | Object的Key.<br>类型：字符串<br>父节点：ListBucketResult.Contents |
| LastModified | Object最后被修改的时间。<br>类型：时间<br>父节点：ListBucketResult.Contents |
| ListBucketResult | 保存Get Bucket请求结果的容器.<br>类型：容器<br>子节点：Name, Prefix, Marker, MaxKeys, Delimiter, IsTruncated, Nextmarker, Contents<br>父节点：None |
| Marker | 标明这次Get Bucket（List Object）的起点。<br>类型：字符串<br>父节点：ListBucketResult |
| MaxKeys | 响应请求内返回结果的最大数目。<br>类型：字符串<br>父节点：ListBucketResult |
| Name | Bucket名字<br>类型：字符串<br>父节点：ListBucketResult |
| Owner | 保存Bucket拥有者信息的容器。<br>类型：容器<br>子节点：DisplayName, ID<br>父节点：ListBucketResult |
| Prefix | 本次查询结果的开始前缀。<br>类型：字符串<br>父节点：ListBucketResult |
| Size | Object的字节数。<br>类型：字符串<br>父节点：ListBucketResult.Contents |
| StorageClass | Object的存储类型，目前只能是“Standard”<br>类型：字符串<br>父节点：ListBucketResult.Contents |

**细节分析：**

1) Object中用户自定义的meta，在GetBucket请求时不会返回。
2) 如果访问的Bucket不存在，包括试图访问因为命名不规范无法创建的Bucket，返回404 Not Found错误，错误码：NoSuchBucket。
3) 如果没有访问该Bucket的权限，返回403 Forbidden错误，错误码：AccessDenied。
4) 如果因为max-keys的设定无法一次完成listing，返回结果会附加一个<NextMarker>，提示继续listing可以以此为marker。NextMarker中的值仍在list结果之中。
5) 在做条件查询时，即使marker实际在列表中不存在，返回也从符合marker字母排序的下一个开始打印。如果max-keys小于0或者大于1000，将返回400 Bad Request错误。错误码：InvalidArgument。
6) 若prefix，marker，delimiter参数不符合长度要求，返回400 Bad Request。错误码：InvalidArgument。
7) prefix，marker用来实现分页显示效果，参数的长度必须小于1024字节。
8) 如果把prefix设为某个文件夹名，就可以罗列以此prefix开头的文件，即该文件夹下递归的所有的文件和子文件夹。如果再把delimiter设置为 / 时，返回值就只罗列该文件夹下的文件，该文件夹下的子文件名返回在CommonPrefixes部分，子文件夹下递归的文件和文件夹不被显示。如一个bucket存在三个object ：

fun/test.jpg， fun/movie/001.avi， fun/movie/007.avi。

若设定prefix为”fun/” ，则返回三个object；如果增加设定delimiter为“/”，则返回文件”fun/test.jpg”和前缀”fun/movie/”；即实现了文件夹的逻辑。

**举例场景：**

在bucket“my_oss”内有4个object，名字分别为：

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

**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-ebe65d8aad5f
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>&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>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
        <ID>00220120222</ID>
        <DisplayName>user_example</DisplayName>
      </Owner>
    </Contents>
  </ListBucketResult>
```

请求示例(含prefix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:DNrnx7xHk3sgysx7I8U9I9IY1vY=
```

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>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
    <Type>Normal</Type>
    <Size>344606</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>00220120222</ID>
      <DisplayName>user_example</DisplayName>
    </Owner>
  </Contents>
  <CommonPrefixes>
    <Prefix>fun/movie/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

相关阅读：

Ø **Put Bucket**

Ø **Put Object**

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

**细节分析：**

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

**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+WwIfNm0FmgbrQ0=
```

**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-Tupe: 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：

Ø **Put Bucket**

Ø **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
Server: AliyunOSS
```

**相关阅读：**

Ø **Get Bucket (List Object)**

Ø **Delete Bucket**

Ø **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：

Ø **Put Bucket**

Ø **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）<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-match | 如果源Object的ETAG值和用户提供的ETAG相等，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-none-match | 如果源Object自从用户指定的时间以后就没有被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-unmodified-since | 如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误。<br>类型：字符串<br>默认值：无 |
| x-oss-copy-source-if-modified-since | 如果源Object自从用户指定的时间以后被修改过，则执行拷贝操作；否则返回412 HTTP错误码（预处理失败）。<br>类型：字符串<br>默认值：无 |
| x-oss-metadata-directive | 有效值为COPY和REPLACE。如果该值设为COPY，则新的Object的meta都从源Object复制过来；如果设为REPLACE，则忽视所有源Object的meta值，而采用用户这次请求中指定的meta值；其他值则返回400 HTTP错误码。<br>类型：字符串<br>默认值：COPY<br>有效值：COPY \| REPLACE |

**响应元素(Response Elements)：**

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

**细节分析：**

1）可以通过拷贝操作来实现修改已有Object的meta信息。

2）如果拷贝操作的源Object地址和目标Object地址相同，则无论x-oss-metadata-directive为何值，都会直接替换源Object的meta信息。

3）OSS支持拷贝操作的四个预判断Header任意个同时出现，相应逻辑参见Get Object操作的细节分析。

4）拷贝操作需要请求者对源Object有读权限。

5）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。

**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：**

Ø **Put Object**

Ø **Get Object**

Ø **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:zUglwRPGkbByZxm1+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：**

Ø **Put Object**

Ø **Get Object**

Ø **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：

l 详细模式：OSS返回的消息体中会包含每一个删除Object的结果。

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

**响应元素(Response Elements)：**

| 名称 | 描述 |
| --- | --- |
| Deleted | 保存被成功删除的Object的容器。<br>类型：容器<br>子节点：Key<br>父节点：DeleteResult |

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

**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/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: 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>
```

## 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:WuV0Jks8RyGSNQrBca64kEExJDs=

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
   <Quiet>true</Quiet>
   <Object>
      <Key>multipart.data</Key>
   </Object>
   <Object>
      <Key>test.jpg</Key>
   </Object>
   <Object>
      <Key>demo.jpg</Key>
   </Object>
</Delete>
```

## 返回示例：

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

Ø **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<br>类型：字符串<br>默认值：无 |
| response-content-language | 设置OSS返回请求的content-language头<br>类型：字符串<br>默认值：无 |
| response-expires | 设置OSS返回请求的expires头<br>类型：字符串<br>默认值：无 |
| response-cache-control | 设置OSS返回请求的cache-control头<br>类型：字符串<br>默认值：无 |
| response-content-disposition | 设置OSS返回请求的content-disposition头<br>类型：字符串<br>默认值：无 |
| response-content-encoding | 设置OSS返回请求的content-encoding头<br>类型：字符串<br>默认值：无 |

**请求Header：**

| 名称 | 描述 |
| --- | --- |
| Range | 指定文件传输的范围。如，设定 bytes=0-9，表示传送第0到第9这10个字符。<br>类型：字符串<br>默认值：无 |
| If-Modified-Since | 如果指定的时间早于实际修改时间，则正常传送文件，并返回200 OK；否则返回304 not modified<br>类型：字符串<br>默认值：无 |
| If-Unmodified-Since | 如果传入参数中的时间等于或者晚于文件实际修改时间，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误<br>类型：字符串<br>默认值：无 |
| If-Match | 如果传入期望的ETag和object的 ETag匹配，则正常传输文件，并返回200 OK；否则返回412 precondition failed错误<br>类型：字符串<br>默认值：无 |
| If-None-Match | 如果传入的ETag值和Object的ETag不匹配，则正常传输文件,并返回200 OK；否则返回304 Not Modified<br>类型：字符串<br>默认值：无 |

**细节分析：**

1) GetObject通过range参数可以支持断点续传，对于比较大的Object建议使用该功能。
2) 如果在请求头中使用Range参数；则返回消息中会包含整个文件的长度和此次返回的范围，例如：Content-Range: bytes 0-9/44，表示整个文件长度为44，此次返回的范围为0-9。如果不符合范围规范，则传送整个文件，并且不在结果中提及Content-Range。
3) 如果“If-Modified-Since”元素中设定的时间不符合规范，直接返回文件，并返回200 OK。
4) If-Modified-Since和If-Unmodified-Since可以同时存在，If-Match和If-None-Match也可以同时存在。
5) 如果包含If-Unmodified-Since并且不符合或者包含If-Match并且不符合，返回412 precondition failed
6) 如果包含If-Modified-Since并且不符合或者包含If-None-Match并且不符合，返回304 Not Modified
7) 如果文件不存在返回404 Not Found错误。错误码：NoSuchKey。
8) OSS不支持在匿名访问的GET请求中，通过请求参数来自定义返回请求的header。
9) 在自定义OSS返回请求中的一些Header时，只有请求处理成功（即返回码为200时），OSS才会将请求的header设置成用户GET请求参数中指定的值。

**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/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.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：

Ø **Get Object**

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

**细节分析：**

1) 不论正常返回200 OK还是非正常返回，Head Object都不返回消息体。

2) HeadObject支持在头中设定If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match的查询条件。具体规则请参见GetObject中对应的选项。如果没有修改，返回304 Not Modified。

3) 如果用户在PutObject的时候传入以x-oss-meta-为开头的user meta，比如x-oss-meta-location，返回消息时，返回这些user meta。

4) 如果文件不存在返回404 Not Found错误。

**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/jpg
Connection: close
```

```
Server: AliyunOSS
```

相关阅读：

Ø **Put Object**

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

**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，以便用户检查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。

**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：**

Ø **Get Object**

Ø **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：

l 需要支持断点上传。

l 上传超过100MB大小的文件。

l 网络条件较差，和OSS的服务器之间的链接经常断开。

l 需要流式地上传文件。

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

**响应元素(Response Elements)：**

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

| | |
|---|---|
| Key | 初始化一个Multipart Upload事件的Object名称。<br>类型：字符串<br>父节点：InitiateMultipartUploadResult |
| UploadId | 唯一标示此次Multipart Upload事件的ID。<br>类型：字符串<br>父节点：InitiateMultipartUploadResult |

**细节分析：**

1）该操作计算认证签名的时候，需要加“?uploads”到CanonicalizedResource中。

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

3）初始化Multipart Upload请求，并不会影响已经存在的同名object。

4）服务器收到初始化Multipart Upload请求后，会返回一个XML格式的请求体。该请求体内有三个元素：Bucket，Key和UploadID。请记录下其中的UploadID，以用于后续的Multipart相关操作。

**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：**

Ø **List Multipart Uploads**

Ø **Abort Multipart Upload**

Ø **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。

**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 qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMmUmWjI=

[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：**

Ø **Initiate Multipart Upload**

Ø **List Parts**

Ø **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。<br>类型：容器<br>子节点：一个或多个*Part*元素<br>父节点：无 |
| ETag | Part成功上传后，OSS返回的ETag值。<br>类型：字符串<br>父节点：Part |
| Part | 保存已经上传Part信息的容器。<br>类型：容器<br>子节点：ETag, PartNumber<br>父节点：InitiateMultipartUploadResult |
| PartNumber | Part数目。<br>类型：整数<br>父节点：Part |

**响应元素(Response Elements)：**

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

| | |
|---|---|
| Location | 新创建Object的URL。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult |
| Key | 新创建Object的名字。<br>类型：字符串<br>父节点：CompleteMultipartUploadResult |

**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。

**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:8VwFhFUWmVecK6jQlHlXMK/zMT0=

<CompleteMultipartUpload>
   <Part>
      <PartNumber>1</PartNumber>
      <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
   </Part>
   <Part>
      <PartNumber>5</PartNumber>
      <ETag>"8EFDA8BE206636A695359836FE0A0E0A"</ETag>
   </Part>
   <Part>
      <PartNumber>8</PartNumber>
      <ETag>"8C3150651671324441774 11FDA149B92"</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>&amp;quot;B864DB6A936D376F9F8D3ED3BBE540DD-3&amp;quot;</ETag>
</CompleteMultipartUploadResult>
```

**相关阅读：**

Ø **Initiate Multipart Upload**

Ø **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/lICfXEvPmmSW86bBAfMmUmWjI=
```

**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：**

Ø **Initiate Multipart Upload**

Ø **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。<br>类型：字符串 |
| max-uploads | 限定此次返回Multipart Uploads事件的最大数目，如果不设定，默认为1000，max-keys取值不能大于1000。<br>类型：字符串 |
| key-marker | 与upload-id-marker参数一同使用来指定返回结果的起始位置。<br>l 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。<br>l 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。<br>类型：字符串 |
| prefix | 限定返回的object key必须以prefix作为前缀。注意使用prefix查询时，返回的key中仍会包含prefix。<br>类型：字符串 |
| upload-id-marker | 与key-marker参数一同使用来指定返回结果的起始位置。<br>l 如果key-marker参数未设置，则OSS忽略upload-id-marker参数。<br>l 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。<br>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。<br>类型：容器<br>子节点：*Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MasUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload*<br>父节点：None |
| Bucket | Bucket名称。<br>类型：字符串<br>父节点：ListMultipartUploadsResult |
| KeyMarker | 列表的起始Object位置。<br>类型：字符串<br>父节点：ListMultipartUploadsResult |

| | |
|---|---|
| UploadIdMarker | 列表的起始UploadID位置。<br>类型：字符串<br>父节点：ListMultipartUploadsResult |
| NextKeyMarker | 如果本次没有返回全部结果，响应请求中将包含NextKeyMarker元素，用于标明接下来请求的KeyMarker值。<br>类型：字符串<br>父节点：ListMultipartUploadsResult |
| NextUploadMarker | 如果本次没有返回全部结果，响应请求中将包含NextUploadMarker元素，用于标明接下来请求的UploadMarker值。<br>类型：字符串<br>父节点：ListMultipartUploadsResult |
| MaxUploads | 返回的最大Upload数目。<br>类型：整数<br>父节点：ListMultipartUploadsResult |
| IsTruncated | 标明是否本次返回的Multipart Upload结果列表被截断。“true”表示本次没有返回全部结果；“false”表示本次已经返回了全部结果。<br>类型：枚举字符串<br>有效值：false \| true<br>默认值：false<br>父节点：ListMultipartUploadsResult |
| Upload | 保存Multipart Upload事件信息的容器。<br>类型：容器<br>子节点：*Key, UploadId, Initiated*<br>父节点：ListMultipartUploadsResult |
| Key | 初始化Multipart Upload事件的Object名字。<br>类型：字符串<br>父节点：Upload |
| UploadId | Multipart Upload事件的ID。<br>类型：字符串<br>父节点：Upload |
| Initiated | Multipart Upload事件初始化的时间.<br>类型：日期<br>父节点：Upload |

**细节分析：**

1）“max-uploads”参数最大值为1000。

2）在OSS的返回结果首先按照Object名字字典序升序排列；对于同一个Object，则按照时间序，升序排列。

3）可以灵活地使用prefix参数对bucket内的object进行分组管理（类似与文件夹的功能）。

4）List Multipart Uploads请求支持5种请求参数： prefix，marker，delimiter，upload-id-marker和max-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：

Ø **Initiate Multipart Upload**

Ø **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。<br>类型：字符串<br>默认值：无 |
| max-parts | 规定在OSS响应中的最大Part数目。<br>类型：整数<br>默认值：1,000 |
| part-number-marker | 指定List的起始位置，只有Part Number数目大于该参数的Part会被列出。<br>类型：整数<br>默认值：无 |

| 名称 | 描述 |
| --- | --- |
| uploadId | Upload ID identifying the multipart upload whose parts are being listed.<br>Type: String<br>Default: None |
| max-parts | Sets the maximum number of parts to return in the response body.<br>Type: Integer<br>Default: 1,000 |
| part-number-marker | Specifies the part after which listing should begin. Only parts with higher part numbers will be listed.<br>Type: Integer<br>Default: None |

**响应元素(Response Elements)：**

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

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

**细节分析：**

1）List Parts支持*max-parts*和*part-number-marker*两种请求参数。

2） *max-parts*参数最大值为1000；默认值也为1000。

3）在OSS的返回结果按照Part号码升序排列。

4）由于网络传输可能出错，所以不推荐用List Part出来的结果（Part Number和ETag值）来生成最后Complete Multipart的Part列表。

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

相关阅读：

Ø **Initiate Multipart Upload**

Ø **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：

n **Code**：OSS返回给用户的错误码。

n **Message**：OSS给出的详细错误信息。

n **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。

n **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 |
| 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>
```

---

[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] 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。

[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, reponse-cache-control, response-content-disposition, response-content-encoding