对象存储服务

# API 参考（安卡拉区域）

文档版本 01

发布日期 2024-04-15

华为云计算技术有限公司

### 商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

### 注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

## 华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

# 目 录



# 1 使用前必读

## 1.1 概述

欢迎使用对象存储服务OBS（Object Storage Service）。对象存储服务提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。适合企业备份/归档、视频点播、视频监控等多种数据存储场景。

您可以使用本文档提供的API对OBS进行相关操作，如创建、修改、删除桶，上传、下载、删除对象等。支持的全部操作请参见**API概览**。

在调用OBS API之前，请确保已经充分了解OBS相关概念。详细信息请参见**基本概念**。

## 1.2 调用说明

OBS提供了REST（Representational State Transfer）风格API，支持您通过HTTP/HTTPS请求调用，调用方法请参见**如何调用API**。

## 1.3 终端节点

**表 1-1** OBS 终端节点信息

| 区域名称 | 区域 | 终端节点（Endpoint） | 协议类型 |
| --- | --- | --- | --- |
| 土耳其-安卡拉-PUR | tr-central-201 | obs.tr-central-201.hc.vodafone.com.tr | HTTPS/HTTP |

OBS在每个区域都提供独立的二级域名，访问OBS服务既可以使用OBS提供的域名，也可以使用自定义域名。

# 1.4 基本概念

## 使用 OBS API 涉及的常用概念

- 账号

  用户的账号，账号对其所拥有的资源及云服务具有完全的访问权限，可以重置用户密码、分配用户权限等。

- 用户

  由账号在IAM中创建的用户，是云服务的使用人员，具有身份凭证（密码和访问密钥）。

  在控制台“我的凭证”下，您可以查看账号ID和用户ID，管理账号或IAM用户的访问密钥。

  通常在调用API的鉴权过程中，您需要用到账号和IAM用户的访问密钥。

- 桶

  桶是用于存放对象的容器，是OBS中最高等级的命名空间。每个对象都存放在一个桶中。例如，如果名为“picture.jpg”的对象存放在“photo”桶中，则可使用URL（http://photo.obs.*region*.*example*.com/picture.jpg）对该对象进行寻址。

- 对象

  对象在OBS中是最基本的实体。在一个桶中可以存放多个对象，OBS系统并不能区分对象的类型。在OBS系统中存储的对象是被序列化了的，因此它可能是一个文本文件或者一个视频文件。OBS支持的数据大小范围可以是0B到48.8TB（包含0B和48.8TB），PutObject接口上传对象时对象最大为5GB，超过5GB对象需要使用多段方式上传。

- 区域

  指云资源所在的物理位置，同一区域（region）内可用区间内网互通，不同区域间内网不互通。通过在不同地区创建云资源，可以将应用程序设计的更接近特定客户的要求，或满足不同地区的法律或其他要求。

  OBS中的桶归属于某个区域，这个区域是您在创桶时选定的，桶一旦创建，其归属区域信息不能改变。您可以根据地理位置、成本、满足法规要求等标准来选择桶的区域。可以选择的区域请参见**终端节点**。

# 2 API 概览

## 桶基础操作接口

**表 2-1** 桶基础操作接口

| 接口 | 说明 |
| --- | --- |
| 获取桶列表 | 查询自己创建的桶列表。 |
| 创建桶 | 创建一个新桶。可以在创建时添加不同的请求消息头来指定桶的区域、权限控制策略、等信息。 |
| 列举桶内对象 | 获取桶内对象列表。可以在创建时添加不同的请求消息头来获取符合指定前缀、标识符等要求的对象。 |
| 获取桶元数据 | 查询桶元数据是否存在。可以查询桶的区域、OBS服务版本号、CORS配置等信息。 |
| 获取桶区域位置 | 获取桶区域位置信息。 |
| 删除桶 | 删除指定的桶。删除之前需要确保桶内无对象。 |

## 桶高级配置接口

**表 2-2** 桶高级配置接口

| 接口 | 说明 |
| --- | --- |
| 设置桶策略 | 创建或者修改一个桶的策略。如果桶已经存在一个策略，那么当前请求中的策略将完全覆盖桶中现存的策略。 |
| 获取桶策略 | 获取指定桶的策略信息。 |
| 删除桶策略 | 删除一个指定桶上的策略。 |

| 接口 | 说明 |
| --- | --- |
| 设置桶ACL | 设置一个指定桶的ACL信息。通过ACL可以控制桶的读写权限。 |
| 获取桶ACL | 获取一个指定桶的ACL信息。 |
| 设置桶日志管理配置 | 开启或关闭桶的日志管理功能。开启后，桶的每次操作将会产生一条日志，并将多条日志打包成一个日志文件存放在指定的位置。 |
| 获取桶日志管理配置 | 获取指定桶的日志管理配置信息。 |
| 设置桶的生命周期配置 | 指定规则来实现定时删除桶中对象。 |
| 获取桶的生命周期配置 | 获取指定桶已配置的生命周期规则。 |
| 删除桶的生命周期配置 | 删除指定桶的生命周期配置信息。 |
| 设置桶的多版本状态 | 开启或暂停桶的多版本功能。开启后，可以检索和还原各个版本的对象，在意外操作或应用程序故障时快速恢复数据。 |
| 获取桶的多版本状态 | 获取指定桶的多版本功能状态。 |
| 设置桶的消息通知配置 | 设置桶的消息通知功能，安全、及时的了解发生在桶上的关键事件。 |
| 获取桶的消息通知配置 | 获取指定桶的消息通知配置信息。 |
| 设置桶的跨区域复制配置 | 设置桶的跨区域复制功能。通过激活跨区域复制，OBS可将新创建的对象及修改的对象从一个源桶复制到不同区域中的目标桶。 |
| 获取桶的跨区域复制配置 | 获取指定桶的跨区域复制配置信息。 |
| 删除桶的跨区域复制配置 | 删除指定桶的跨区域复制配置信息。 |
| 设置桶标签 | 添加标签至一个已存在的桶。为桶添加标签后，该桶上所有请求产生的话单里都会带上这些标签，从而可以针对话单报表做分类筛选，进行更详细的成本分析。 |
| 获取桶标签 | 获取指定桶的标签。 |
| 删除桶标签 | 删除指定桶的标签。 |
| 设置桶配额 | 设置桶的空间配额，用以限制桶的最大存储容量。 |
| 获取桶配额 | 获取桶的空间配额。 |
| 获取桶存量信息 | 获取桶中的对象个数及对象占用空间。 |
| 设置桶清单 | 为一个桶配置清单规则。桶清单可以用来帮助您管理桶内对象，它可以定期列举桶内对象，并将对象元数据的相关信息保存在CSV格式的文件中，上传到您指定的桶中。 |

| 接口 | 说明 |
| --- | --- |
| 获取桶清单 | 获取指定桶的某个清单规则。 |
| 列举桶清单 | 获取指定桶的所有清单规则。 |
| 删除桶清单 | 删除指定桶的某个清单规则。 |
| 设置桶的自定义域名 | 为桶设置自定义域名。设置成功之后，用户可以通过桶的自定义域名访问桶。 |
| 获取桶的自定义域名 | 查询桶已设置的自定义域名。 |
| 删除桶的自定义域名 | 删除桶已设置的自定义域名。 |
| 设置桶的跨集群复制配置 | 设置指定桶的跨集群复制配置。 |
| 获取桶的跨集群复制配置 | 获取指定桶的跨集群复制配置。 |
| 删除桶的跨集群复制配置 | 删除指定桶的跨集群复制配置。 |

## 静态网站托管接口

**表 2-3** 静态网站托管接口

| 接口 | 说明 |
| --- | --- |
| 设置桶的网站配置 | 创建或更新桶的网站配置信息。OBS允许在桶内保存静态的网页资源，如.html网页文件、flash文件、音视频文件等，当客户端通过桶的Website接入点访问这些对象资源时，浏览器可以直接解析出这些支持的网页资源，呈现给最终用户。 |
| 获取桶的网站配置 | 获取桶的网站配置信息。 |
| 删除桶的网站配置 | 删除桶的网站配置信息。 |
| 设置桶的CORS配置 | 设置桶的跨域资源共享配置信息。OBS允许在桶内保存静态的网页资源，在正确的使用下，OBS的桶可以成为网站资源。只有进行了适当的CORS配置，OBS中的网站才能响应另一个网站的跨域请求。 |
| 获取桶的CORS配置 | 获取桶的跨域资源共享配置信息。 |
| 删除桶的CORS配置 | 删除桶的跨域资源共享配置信息。 |
| OPTIONS桶 | 检测客户端是否具有对服务端进行操作的权限。通常用于跨域访问之前。 |
| OPTIONS对象 | 检测客户端是否具有对服务端进行操作的权限。通常用于跨域访问之前。 |

## 对象操作接口

**表 2-4** 对象操作接口

| 接口 | 说明 |
| --- | --- |
| PUT上传 | 上传简单对象到指定的桶。 |
| POST上传 | 基于表单上传对象到指定的桶。 |
| 复制对象 | 为OBS上已经存在的对象创建一个副本。 |
| 下载对象 | 下载对象。 |
| 获取对象元数据 | 获取对象的元数据信息。包括对象的过期时间、版本号、CORS配置等信息。 |
| 删除对象 | 删除指定的对象。也可以携带versionId删除指定版本的对象。 |
| 批量删除对象 | 将一个桶内的一部分对象一次性删除，删除后不可恢复。 |
| 追加写对象 | 在指定桶内的一个对象尾追加上传数据，不存在相同对象键值的对象则创建新对象。 |
| 设置对象ACL | 设置一个指定对象的ACL信息。通过ACL可以控制对象的读写权限。 |
| 获取对象ACL | 获取一个指定对象的ACL信息。 |
| 修改对象元数据 | 添加、修改或删除桶中已经上传的对象的元数据。 |
| 修改写对象 | 将指定并行文件系统内的一个对象从指定位置起修改为其他内容。 |
| 截断对象 | 将指定并行文件系统内的一个对象截断到指定大小。 |
| 重命名对象 | 将指定并行文件系统内的一个对象重命名为其他对象名。 |

## 多段操作接口

**表 2-5** 多段操作接口

| 接口 | 说明 |
| --- | --- |
| 列举桶中已初始化多段任务 | 查询一个桶中所有的初始化后还未合并以及未取消的多段上传任务。 |

| 接口 | 说明 |
| --- | --- |
| 初始化上传段任务 | 使用多段上传特性时，必须首先调用此接口初始化上传段任务，获取全局唯一的多段上传任务号，用于后续的上传段、合并段、列举段等操作。 |
| 上传段 | 为特定的任务上传段。 |
| 拷贝段 | 将已上传对象的一部分或全部拷贝为段。 |
| 列举已上传的段 | 查询一个任务所属的所有段信息。 |
| 合并段 | 将指定的段合并成一个完整的对象。 |
| 取消多段上传任务 | 取消一个多段上传的任务。 |

# 3 如何调用 API

## 3.1 构造请求

本节介绍REST API请求的组成。

### 请求 URI

OBS根据桶和对象及带的资源参数来确定具体的URI，当需要进行资源操作时，可以使用这个URI地址。

URI的一般格式为（方括号内为可选项）：

**protocol://[bucket.]domain[:port][/object][?param]**

**表 3-1** URI 中的参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| protocol | 请求使用的协议类型，如HTTP、HTTPS。HTTPS表示通过安全的HTTPS访问该资源，对象存储服务支持HTTP，HTTPS两种传输协议。 | 必选 |
| bucket | 请求使用的桶资源路径，在整个系统中唯一标识一个桶。 | 可选 |
| domain | 存放资源的服务器的域名或IP地址。 | 必选 |
| port | 请求使用的端口号。根据软件服务器的部署不同而不同。缺省时使用默认端口，各种传输协议都有默认的端口号，如HTTP的默认端口为80，HTTPS的默认端口为443。<br>OBS对象存储服务的HTTP方式访问端口为80，HTTPS方式访问端口为443。 | 可选 |
| object | 请求使用的对象资源路径。 | 可选 |
| param | 请求使用的桶和对象的具体资源，缺省默认为请求桶或对象自身资源。 | 可选 |

**须知**

除获取桶列表之外的所有接口，都应当包含桶名。OBS基于DNS解析性能和可靠性的考虑，要求凡是携带桶名的请求，在构造URL的时候都必须将桶名放在domain前面，形成三级域名形式，又称为虚拟主机访问域名。

例如，如果您有一个位于a1区域的名为test-bucket的桶，期望访问桶中一个名为test-object对象的acl，正确的访问URL为https://test-bucket.obs.a1.example.com/test-object?acl

## 请求方法

HTTP方法（也称为操作或动词），它告诉服务你正在请求什么类型的操作。

**表 3-2** 对象存储支持的 REST 请求方法

| 方法 | 说明 |
| --- | --- |
| GET | 请求服务器返回指定资源，如获取桶列表、下载对象等。 |
| PUT | 请求服务器更新指定资源，如创建桶、上传对象等。 |
| POST | 请求服务器新增资源或执行特殊操作，如初始化上传段任务、合并段等。 |
| DELETE | 请求服务器删除指定资源，如删除对象等。 |
| HEAD | 请求服务器返回指定资源的概要，如获取对象元数据等。 |
| OPTIONS | 请求服务器检查是否具有某个资源的操作权限，需要桶配置CORS。 |

## 请求消息头

可选的附加请求头字段，如指定的URI和HTTP方法所要求的字段。详细的公共请求消息头字段请参见表3-3。

**表 3-3** 公共请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Authorization | 请求消息中可带的签名信息。<br>类型：String<br>默认值：无。<br>条件：匿名请求不需要带，其他请求必选。 | 有条件必选 |
| Content-Length | RFC 2616中定义的消息（不包含消息头）长度。<br>类型：String<br>默认值：无。<br>条件：PUT操作可选，加载XML的操作必须带。 | 有条件必选 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-Type | 资源内容的类型，例如： text/plain。<br>类型：String<br>默认值：无。 | 否 |
| Date | 请求发起端的日期和时间，例如：Wed, 27 Jun 2018 13:39:15 +0000。<br>类型：String<br>默认值：无。<br>条件：如果是匿名请求或者消息头中带了x-obs-date字段，则可以不带该字段，其他情况下必选。 | 有条件必选 |
| Host | 表明主机地址。如 bucketname.obs.*region*.example.com。<br>类型：String<br>默认值：无。 | 是 |

### 请求消息体（可选）

请求消息体通常以结构化格式（如JSON或XML）发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。如果请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，也并不是每个接口都需要有请求消息体（或者说消息体为空），GET、DELETE操作类型的接口就不需要消息体，消息体具体内容需要根据具体接口而定。

### 发起请求

共有两种方式可以基于已构建好的请求消息发起请求，分别为：

- cURL

  cURL是一个命令行工具，用来执行各种URL操作和信息传输。cURL充当的是HTTP客户端，可以发送HTTP请求给服务端，并接收响应消息。cURL适用于接口调试。关于cURL详细信息请参见**https://curl.haxx.se/**。由于cURL无法计算签名，使用cURL时仅支持访问匿名的公共OBS资源。
- 编码

  通过编码调用接口，组装请求消息，并发送处理请求消息。

# 3.2 认证鉴权

## 3.2.1 用户签名验证

OBS通过AK/SK对请求进行签名，在向OBS发送请求时，客户端发送的每个消息头需要包含由SK、请求时间、请求类型等信息生成的签名信息。

- AK(Access Key ID)：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- SK(Secret Access Key)：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

用户可以在IAM服务中获取AK和SK，获取的方法请参见**获取访问密钥（AK/SK）**。

OBS根据应用场景，提供了**Header中携带签名**、**URL中携带签名**和**基于浏览器上传的表单中携带签名**3种签名计算方式。

以Header中携带签名为例，用户签名验证流程如**表3-4**所示。Header中携带签名方法的具体参数说明及代码示例，请参见**Header中携带签名**。

**表 3-4** OBS 签名计算和验证步骤

| 步骤 | | 示例 |
| --- | --- | --- |
| 签名计算 | 1. 构造HTTP消息 | PUT /object HTTP/1.1<br>Host: bucket.obs.region.example.com<br>Date: Tue, 04 Jun 2019 06:54:59 GMT<br>Content-Type: text/plain<br>Content-Length: 5913 |
| | 2. 按照签名规则计算**StringToSign** | StringToSign = HTTP-Verb + "\n" + Content-MD5 + "\n" + Content-Type + "\n" + Date + "\n" + CanonicalizedHeaders + CanonicalizedResource |
| | 3. 准备AK和SK | AK: ******<br>SK: ****** |
| | 4. 计算签名**Signature** | Signature = Base64( HMAC-SHA1( **SecretAccessKeyID**, UTF-8-Encoding-Of( **StringToSign** ) ) ) |
| | 5. 添加签名头域发送到OBS服务 | PUT /object HTTP/1.1<br>Host: bucket.obs.region.example.com<br>Date: Tue, 04 Jun 2019 06:54:59 GMT<br>Content-Type: text/plain<br>Content-Length: 5913<br>Authorization: OBS **AccessKeyID**:**Signature** |
| 签名验证 | 6. 接收HTTP消息 | PUT /object HTTP/1.1<br>Host: bucket.obs.region.example.com<br>Date: Tue, 04 Jun 2019 06:54:59 GMT<br>Content-Type: text/plain<br>Content-Length: 5913<br>Authorization: OBS **AccessKeyID**:**Signature** |
| | 7. 根据请求中的AK获取SK | 从头域Authorization中取出AK，去IAM取回用户的SK |

| 步骤 | | 示例 |
| --- | --- | --- |
| | 8. 按照签名规则计算**StringToSign** | StringToSign = HTTP-Verb + "\n" + Content-MD5 + "\n" + Content-Type + "\n" + Date + "\n" + CanonicalizedHeaders + CanonicalizedResource |
| | 9. 计算签名**Signature** | Signature = Base64( HMAC-SHA1( **SecretAccessKeyID**, UTF-8-Encoding-Of( **StringToSign** ) ) ) |
| | 10. 验证签名 | 验证头域Authorization中的**Signature**与服务端计算的**Signature**是否相等<br>相等：签名验证通过<br>不相等：签名验证失败 |

## 3.2.2 Header 中携带签名

OBS的所有API接口都可以通过在header中携带签名方式来进行身份认证，也是最常用的身份认证方式。

在Header中携带签名是指将通过HTTP消息中Authorization header头域携带签名信息，消息头域的格式为：

```
Authorization: OBS AccessKeyID:signature
```

签名的计算过程如下：

1. 构造请求字符串(StringToSign)。
2. 对第一步的结果进行UTF-8编码。
3. 使用SK对第二步的结果进行HMAC-SHA1签名计算。
4. 对第三步的结果进行Base64编码，得到签名。

请求字符串(StringToSign)按照如下规则进行构造，各个参数的含义如表3-5所示。

```
StringToSign =
   HTTP-Verb + "\n" +
   Content-MD5 + "\n" +
   Content-Type + "\n" +
   Date + "\n" +
   CanonicalizedHeaders + CanonicalizedResource
```

**表 3-5** 构造 StringToSign 所需参数说明

| 参数 | 描述 |
| --- | --- |
| HTTP-Verb | 指接口操作的方法，对REST接口而言，即为http请求操作的VERB，如："PUT"，"GET"，"DELETE"等字符串。 |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串，可以为空。具体请参见表3-10以及表下方的计算方法示例。 |

| 参数 | 描述 |
| --- | --- |
| Content-Type | 内容类型，用于指定消息类型，例如： text/plain。<br>当请求中不带该头域时，该参数按照空字符串处理，见**表3-6**。 |
| Date | 生成请求的时间，该时间格式遵循RFC 1123；该时间与当前服务器的时间超过15分钟时服务端返回403。<br>当有自定义字段x-obs-date时，该参数按照空字符串处理；见**表3-10**。<br>如果进行临时授权方式操作（如临时授权方式获取对象内容等操作）时，该参数不需要。 |
| CanonicalizedHeaders | HTTP请求头域中的OBS请求头字段，即以“x-obs-”作为前缀的头域，如“x-obs-date，x-obs-acl，x-obs-meta-*”。用户在调用API时，请根据自身需求，从调用的API支持的头域中选取。<br>1. 请求头字段中关键字的所有字符要转为小写（但内容值需要区分大小写），需要添加多个字段时，要将所有字段按照关键字的字典序从小到大进行排序。<br>2. 在添加请求头字段时，如果有重名的字段，则需要进行合并。如：x-obs-meta-name:name1和x-obs-meta-name:name2，则需要先将重名字段的值（这里是name1和name2）以逗号分隔，合并成x-obs-meta-name:name1,name2。<br>3. 头域中的请求头字段中的关键字不允许含有非ASCII码或不可识别字符；请求头字段中的值也不建议使用非ASCII码或不可识别字符，如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。<br>4. 当请求头字段中含有无意义空格或Tab键时，需要摒弃。例如：x-obs-meta-name: name（name前带有一个无意义空格），需要转换为：x-obs-meta-name:name<br>5. 每一个请求头字段最后都需要另起新行，见**表3-8** |

| 参数 | 描述 |
| --- | --- |
| CanonicalizedResource | 表示HTTP请求所指定的OBS资源，构造方式如下：<br><桶名+对象名>+[子资源1] + [子资源2] + ...<br>1. 桶名和对象名，例如：/bucket/object。如果没有对象名，如列举桶，则为"/bucket/"，如桶名也没有，则为“/”。<br>2. 如果有子资源，则将子资源添加进来，例如?acl，?logging。OBS支持各种子资源，包括：acl, append, attname, cors, customdomain, delete, deletebucket, inventory, length, lifecycle, location, logging, metadata, modify, name, notification, partNumber, policy, position, quota, rename, replication, response-cache-control, response-content-disposition, response-content-encoding, response-content-language, response-content-type, response-expires, storagePolicy, storageinfo, tagging, torrent, truncate, uploadId, uploads, versionId, versioning, versions, website, x-obs-security-token。<br>3. 如果有多个子资源，在包含这些子资源时，需要首先将这些子资源按照其关键字的字典序从小到大排列，并使用**“&”**拼接。<br>**说明**<br>• 子资源通常是唯一的，不建议请求的URL包含多个相同关键字的子资源（例如，key=value1&key=value2），如果存在这种情况，OBS服务端签名时只会计算第一个子资源且也只有第一个子资源的值会对实际业务产生作用；<br>• 以获取对象（GetObject）接口为例，假设桶名为bucket-test，对象名为object-test，对象的版本号为xxx，获取时需要重写Content-Type为text/plain，那么签名计算出的CanonicalizedResource为：/bucket-test/object-test?response-content-type=text/plain&versionId=xxx。 |

下面的几张表提供了一些生成StringToSign的例子。

**表 3-6** 获取对象

| 请求消息头 | StringToSign |
| --- | --- |
| GET /object.txt HTTP/1.1<br>Host: bucket.obs.*region*.example.com<br>Date: Sat, 12 Oct 2015 08:12:38 GMT | GET \n<br>\n<br>\n<br>Sat, 12 Oct 2015 08:12:38 GMT\n<br>/bucket/object.txt |

**表 3-7** 使用临时 AK/SK 和 securitytoken 上传对象

| 请求消息头 | StringToSign |
| --- | --- |
| PUT /object.txt HTTP/1.1<br>User-Agent: curl/7.15.5<br>Host: bucket.obs.*region*.example.com<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT<br>x-obs-security-token: YwkaRTbdY8g7q....<br>content-type: text/plain<br>Content-Length: 5913339 | PUT\n<br>\n<br>text/plain\n<br>\n<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT\n<br>x-obs-security-token:YwkaRTbdY8g7q....\n<br>/bucket/object.txt |

**表 3-8** 带请求头字段上传对象

| 请求消息头 | StringToSign |
| --- | --- |
| PUT /object.txt HTTP/1.1<br>User-Agent: curl/7.15.5<br>Host: bucket.obs.*region*.example.com<br>Date: Mon, 14 Oct 2015 12:08:34 GMT<br>x-obs-acl: public-read<br>content-type: text/plain<br>Content-Length: 5913339 | PUT\n<br>\n<br>text/plain\n<br>Mon, 14 Oct 2015 12:08:34 GMT\n<br>x-obs-acl:public-read\n<br>/bucket/object.txt |

**表 3-9** 获取对象 ACL

| 请求消息头 | StringToSign |
| --- | --- |
| GET /object.txt?acl HTTP/1.1<br>Host: bucket.obs.*region*.example.com<br>Date: Sat, 12 Oct 2015 08:12:38 GMT | GET \n<br>\n<br>\n<br>Sat, 12 Oct 2015 08:12:38 GMT\n<br>/bucket/object.txt?acl |

**表 3-10** 上传对象且携带 Content-MD5 头域

| 请求消息头 | StringToSign |
| --- | --- |
| PUT /object.txt HTTP/1.1<br>Host: bucket.obs.*region*.example.com<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT<br>Content-MD5: I5pU0r4+sgO9Emgl1KMQUg==<br>Content-Length: 5913339 | PUT\n<br>I5pU0r4+sgO9Emgl1KMQUg==\n<br>\n<br>\n<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT\n<br>/bucket/object.txt |

**表 3-11** 使用自定义域名方式上传对象

| 请求消息头 | StringToSign |
| --- | --- |
| PUT /object.txt HTTP/1.1<br>Host: obs.ccc.com<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT<br>Content-MD5: I5pU0r4+sgO9Emgl1KMQUg==<br>Content-Length: 5913339 | PUT\n<br>I5pU0r4+sgO9Emgl1KMQUg==\n<br>\n<br>\n<br>x-obs-date:Tue, 15 Oct 2015 07:20:09 GMT\n<br>/obs.ccc.com/object.txt |

## Java 中 Content-MD5 的计算方法示例

```
import java.security.MessageDigest;
import sun.misc.BASE64Encoder;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;

public class Md5{
    public static void main(String[] args) {
        try {
            String exampleString = "blog";
            MessageDigest messageDigest = MessageDigest.getInstance("MD5");
            BASE64Encoder encoder = new BASE64Encoder();
            String contentMd5 = encoder.encode(messageDigest.digest(exampleString.getBytes("utf-8")));
            System.out.println("Content-MD5:" + contentMd5);
        } catch (NoSuchAlgorithmException | UnsupportedEncodingException e)
        {
            e.printStackTrace();
        }
    }
}
```

根据请求字符串(StringToSign)和用户SK使用如下算法生成Signature，生成过程使用HMAC算法(hash-based authentication code algorithm)。

```
Signature = Base64( HMAC-SHA1( YourSecretAccessKeyID, UTF-8-Encoding-Of( StringToSign ) ) )
```

例如在某区域创建桶名为newbucketname2的私有桶，客户端请求格式为：

```
PUT / HTTP/1.1
Host: newbucketname2.obs.region.example.com
Content-Length: length
```

```
Date: Fri, 06 Jul 2018 03:45:51 GMT
x-obs-acl:private
Authorization: OBS UDSIAMSTUBTEST000254:ydH8ffpcbS6YpeOMcEZfn0wE90c=
<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
    <Location>region</Location>
</CreateBucketConfiguration>
```

## Java 中签名的计算方法

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.TreeMap;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class SignDemo {

    private static final String SIGN_SEP = "\n";

    private static final String OBS_PREFIX = "x-obs-";

    private static final String DEFAULT_ENCODING = "UTF-8";

    private static final List<String> SUB_RESOURCES = Collections.unmodifiableList(Arrays.asList(
        "CDNNotifyConfiguration", "acl", "append", "attname",  "cors", "customdomain", "delete",
       "deletebucket", "inventory", "length", "lifecycle", "location", "logging",
       "metadata", "mirrorBackToSource", "modify", "name", "notification", "obscompresspolicy",
       "partNumber", "policy", "position", "quota","rename", "replication", "response-cache-control",
       "response-content-disposition","response-content-encoding", "response-content-language", "response-
content-type",
       "response-expires","storagePolicy", "storageinfo", "tagging", "torrent", "truncate",
       "uploadId", "uploads", "versionId", "versioning", "versions", "website",
       "x-obs-security-token"));

    private String ak;

    private String sk;

     public String urlEncode(String input) throws UnsupportedEncodingException {
        return URLEncoder.encode(input, DEFAULT_ENCODING)
            .replaceAll("%7E", "~") //for browser
            .replaceAll("%2F", "/")
            .replaceAll("%20", "+");
    }

    private String join(List<?> items, String delimiter) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < items.size(); i++) {
            String item = items.get(i).toString();
            sb.append(item);
            if (i < items.size() - 1) {
                sb.append(delimiter);
            }
        }
        return sb.toString();
    }

    private boolean isValid(String input) {
```

```
        return input != null && !input.equals("");
    }

    public String hmacSha1(String input) throws NoSuchAlgorithmException, InvalidKeyException,
UnsupportedEncodingException {
        SecretKeySpec signingKey = new SecretKeySpec(this.sk.getBytes(DEFAULT_ENCODING), "HmacSHA1");
        Mac mac = Mac.getInstance("HmacSHA1");
        mac.init(signingKey);
        return Base64.getEncoder().encodeToString(mac.doFinal(input.getBytes(DEFAULT_ENCODING)));
    }

    private String stringToSign(String httpMethod, Map<String, String[]> headers, Map<String, String>
queries,
        String bucketName, String objectName) throws Exception{
        String contentMd5 = "";
        String contentType = "";
        String date = "";

        TreeMap<String, String> canonicalizedHeaders = new TreeMap<String, String>();

        String key;
        List<String> temp = new ArrayList<String>();
        for(Map.Entry<String, String[]> entry : headers.entrySet()) {
            key = entry.getKey();
            if(key == null || entry.getValue() == null || entry.getValue().length == 0) {
                continue;
            }

            key = key.trim().toLowerCase(Locale.ENGLISH);
            if(key.equals("content-md5")) {
                contentMd5 = entry.getValue()[0];
                continue;
            }

            if(key.equals("content-type")) {
                contentType = entry.getValue()[0];
                continue;
            }

            if(key.equals("date")) {
                date = entry.getValue()[0];
                continue;
            }

            if(key.startsWith(OBS_PREFIX)) {
                for(String value : entry.getValue()) {
                    if(value != null) {
                        temp.add(value.trim());
                    }
                }
                canonicalizedHeaders.put(key, this.join(temp, ","));
                temp.clear();
            }
        }

        if(canonicalizedHeaders.containsKey("x-obs-date")) {
            date = "";
        }

        // handle method/content-md5/content-type/date
        StringBuilder stringToSign = new StringBuilder();
        stringToSign.append(httpMethod).append(SIGN_SEP)
            .append(contentMd5).append(SIGN_SEP)
            .append(contentType).append(SIGN_SEP)
            .append(date).append(SIGN_SEP);

        // handle canonicalizedHeaders
        for(Map.Entry<String, String> entry : canonicalizedHeaders.entrySet()) {
            stringToSign.append(entry.getKey()).append(":").append(entry.getValue()).append(SIGN_SEP);
```

```
        }

        // handle CanonicalizedResource
        stringToSign.append("/");
        if(this.isValid(bucketName)) {
            stringToSign.append(bucketName).append("/");
            if(this.isValid(objectName)) {
                stringToSign.append(this.urlEncode(objectName));
            }
        }

        TreeMap<String, String> canonicalizedResource = new TreeMap<String, String>();
        for(Map.Entry<String, String> entry : queries.entrySet()) {
            key = entry.getKey();
            if(key == null) {
                continue;
            }

            if(SUB_RESOURCES.contains(key)) {
                canonicalizedResource.put(key, entry.getValue());
            }
        }

        if(canonicalizedResource.size() > 0) {
            stringToSign.append("?");
            for(Map.Entry<String, String> entry : canonicalizedResource.entrySet()) {
                stringToSign.append(entry.getKey());
                if(this.isValid(entry.getValue())) {
                    stringToSign.append("=").append(entry.getValue());
                }
                stringToSign.append("&");
            }
            stringToSign.deleteCharAt(stringToSign.length()-1);
        }

        //    System.out.println(String.format("StringToSign:%s%s", SIGN_SEP, stringToSign.toString()));

        return stringToSign.toString();
    }

    public String headerSignature(String httpMethod, Map<String, String[]> headers, Map<String, String>
queries,
        String bucketName, String objectName) throws Exception {

        //1. stringToSign
        String stringToSign = this.stringToSign(httpMethod, headers, queries, bucketName, objectName);

        //2. signature
        return String.format("OBS %s:%s", this.ak, this.hmacSha1(stringToSign));
    }

    public String querySignature(String httpMethod, Map<String, String[]> headers, Map<String, String>
queries,
        String bucketName, String objectName, long expires) throws Exception {
        if(headers.containsKey("x-obs-date")) {
            headers.put("x-obs-date", new String[] {String.valueOf(expires)});
        } else {
            headers.put("date", new String[] {String.valueOf(expires)});
        }
        //1. stringToSign
        String stringToSign = this.stringToSign(httpMethod, headers, queries, bucketName, objectName);

        //2. signature
        return this.urlEncode(this.hmacSha1(stringToSign));
    }

    public static void main(String[] args) throws Exception {
        SignDemo demo = new SignDemo();
```

```
        /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和YOUR_SK。*/
        demo.ak = System.getenv("YOUR_AK");
        demo.sk = System.getenv("YOUR_SK");

        String bucketName = "bucket-test";
        String objectName = "hello.jpg";
        Map<String, String[]> headers = new HashMap<String, String[]>();
        headers.put("date", new String[] {"Sat, 12 Oct 2015 08:12:38 GMT"});
        headers.put("x-obs-acl", new String[] {"public-read"});
        headers.put("x-obs-meta-key1", new String[] {"value1"});
        headers.put("x-obs-meta-key2", new String[] {"value2", "value3"});
        Map<String, String> queries = new HashMap<String, String>();
        queries.put("acl", null);

        System.out.println(demo.headerSignature("PUT", headers, queries, bucketName, objectName));
    }

}
```

签名计算的样例结果为（按照执行时间的不同变化）：
ydH8ffpcbS6YpeOMcEZfn0wE90c=

### Python 中签名的计算方法

```
import os
import sys
import hashlib
import hmac
import binascii
from datetime import datetime
IS_PYTHON2 = sys.version_info.major == 2 or sys.version < '3'

# 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
# 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和YOUR_SK。
yourSecretAccessKeyID = os.getenv('YOUR_SK')
httpMethod = "PUT"
contentType = "application/xml"
# "date" is the time when the request was actually generated
date = datetime.utcnow().strftime('%a, %d %b %Y %H:%M:%S GMT')
canonicalizedHeaders = "x-obs-acl:private\n"
CanonicalizedResource = "/newbucketname2"
canonical_string = httpMethod + "\n" + "\n" + contentType + "\n" + date + "\n" + canonicalizedHeaders + CanonicalizedResource
if IS_PYTHON2:
    hashed = hmac.new(yourSecretAccessKeyID, canonical_string, hashlib.sha1)
    encode_canonical = binascii.b2a_base64(hashed.digest())[:-1]
else:
    hashed = hmac.new(yourSecretAccessKeyID.encode('UTF-8'), canonical_string.encode('UTF-8'), hashlib.sha1)
    encode_canonical = binascii.b2a_base64(hashed.digest())[:-1].decode('UTF-8')

print(encode_canonical)
```

签名计算的样例结果为（按照执行时间的不同变化）：
ydH8ffpcbS6YpeOMcEZfn0wE90c=

## 3.2.3 URL 中携带签名

URL中携带签名：OBS服务支持用户构造一个特定操作的URL，这个URL中会包含用户AK、签名、有效期、资源等信息，任何拿到这个URL的人均可执行这个操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带

签名信息的下载对象的URL，拿到相应URL的人能下载这个对象，但该URL只在Expires指定的失效时间内有效。URL中携带签名主要用于在不提供给其他人Secret Access Key的情况下，让其他人能用预签发的URL来进行身份认证，并执行预定义的操作。

URL中携带签名请求的消息格式如下：

```
GET /ObjectKey?AccessKeyId=AccessKeyID&Expires=ExpiresValue&Signature=signature HTTP/1.1
Host: bucketname.obs.region.example.com
```

URL中使用临时AK，SK和securitytoken下载对象消息格式如下：

```
GET /ObjectKey?AccessKeyId=AccessKeyID&Expires=ExpiresValue&Signature=signature&x-obs-security-token=securitytoken HTTP/1.1
Host: bucketname.obs.region.example.com
```

参数具体意义如表3-12所示。

**表 3-12** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| AccessKeyId | 签发者的AK信息。OBS根据AK确定签发者的身份，并认为URL就是签发者在访问。<br>类型：String | 是 |
| Expires | 临时授权失效的时间；UTC时间，1970年1月1日零时之后的指定的Expires时间内有效（以秒为单位）。<br>类型：String | 是 |
| Signature | 根据用户SK、Expires等参数计算出的签名信息。<br>类型：String | 是 |
| x-obs-security-token | 使用临时AK/SK鉴权时，临时AK/SK和securitytoken必须同时使用，请求头中需要添加“x-obs-security-token”字段； | 否 |

签名的计算过程如下：

1. 构造请求字符串(StringToSign)。
2. 对第一步的结果进行UTF-8编码。
3. 使用SK对第二步的结果进行HMAC-SHA1签名计算。
4. 对第三步的结果进行Base64编码。
5. 对第四步的结果进行URL编码，得到签名。

请求字符串(StringToSign)按照如下规则进行构造，各个参数的含义如表3-13所示：

```
StringToSign =
    HTTP-Verb + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedHeaders +   CanonicalizedResource;
```

**表 3-13** 构造 StringToSign 所需参数说明

| 参数 | 描述 |
| --- | --- |
| HTTP-Verb | 指接口操作的方法，对REST接口而言，即为http请求操作的VERB，如："PUT"，"GET"，"DELETE"等字符串。 |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串，可以为空。 |
| Content-Type | 内容类型，用于指定消息类型，例如： text/plain。<br>当请求中不带该头域时，该参数按照空字符串处理。 |
| Expires | 临时授权的失效时间，即请求消息参数Expires的值ExpiresValue。 |
| CanonicalizedHeaders | HTTP请求头域中的OBS请求头字段，即以“x-obs-”作为前缀的头域，如“x-obs-date，x-obs-acl，x-obs-meta-*”。<br>1. 请求头字段中关键字的所有字符要转为小写，需要添加多个字段时，要将所有字段按照关键字的字典序从小到大进行排序。<br>2. 在添加请求头字段时，如果有重名的字段，则需要进行合并。如：x-obs-meta-name:name1和x-obs-meta-name:name2，则需要先将重名字段的值（这里是name1和name2）以逗号分隔，合并成x-obs-meta-name:name1,name2。<br>3. 头域中的请求头字段中的关键字不允许含有非ASCII码或不可识别字符；请求头字段中的值也不建议使用非ASCII码或不可识别字符，如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。<br>4. 当请求头字段中含有无意义空格或Tab键时，需要摒弃。例如：x-obs-meta-name: name（name前带有一个无意义空格），需要转换为：x-obs-meta-name:name<br>5. 每一个请求头字段最后都需要另起新行。 |

| 参数 | 描述 |
| --- | --- |
| CanonicalizedResource | 表示HTTP请求所指定的OBS资源，构造方式如下：<br><桶名+对象名>+[子资源]+ [子资源2] + ...<br>1. 桶名和对象名，例如：/bucket/object。如果没有对象名，如列举桶，则为"/bucket/"，如桶名也没有，则为“/”。<br>2. 如果有子资源，则将子资源添加进来，例如?acl，?logging。OBS支持各种子资源，包括：acl, append, attname, cors, customdomain, delete, deletebucket, inventory, length, lifecycle, location, logging, metadata, modify, name, notification, partNumber, policy, position, quota, rename, replication, response-cache-control, response-content-disposition, response-content-encoding, response-content-language, response-content-type, response-expires, storagePolicy, storageinfo, tagging, torrent, truncate, uploadId, uploads, versionId, versioning, versions, website, x-obs-security-token。<br>3. 如果有多个子资源，在包含这些子资源时，需要首先将这些子资源按照其关键字的字典序从小到大排列，并使用**“&”**拼接。<br>**说明**<br>● 子资源通常是唯一的，不建议请求的URL包含多个相同关键字的子资源（例如，key=value1&key=value2），如果存在这种情况，OBS服务端签名时只会计算第一个子资源且也只有第一个子资源的值会对实际业务产生作用；<br>● 以获取对象（GetObject）接口为例，假设桶名为bucket-test，对象名为object-test，对象的版本号为xxx，获取时需要重写Content-Type为text/plain，那么签名计算出的CanonicalizedResource为：/bucket-test/object-test?response-content-type=text/plain&versionId=xxx。 |

根据请求字符串(StringToSign)和用户SK使用如下算法生成Signature，生成过程使用HMAC算法(hash-based authentication code algorithm)。

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKeyID, UTF-8-Encoding-
Of( StringToSign ) ) ) )
```

URL中的Signature计算方法和Header中携带的Authorization签名计算方法有两处不同：

- URL中签名在Base64编码后还要经过URL编码。
- StringToSign中的Expires和原来Authorization消息中的消息头Date对应。

使用URL携带签名方式为浏览器生成预定义的URL实例：

**表 3-14** 下载对象在 URL 中携带签名的请求及 StringToSign

| 请求消息头 | StringToSign |
| --- | --- |
| GET /objectkey?<br>AccessKeyId=MFyfvK41ba2giqM7Uio6PznpdUKGpownRZlmVmHc&Expires=1532779451&Signature=0Akylf43Bm3mD1bh2rM3dmVp1Bo%3D HTTP/1.1<br>Host:<br>examplebucket.obs.*region*.example.com | GET \n<br>\n<br>\n<br>1532779451\n<br>/examplebucket/objectkey |

**表 3-15** 在 URL 中使用临时 AK/SK 和 securitytoken 下载对象请求及 StringToSign

| 请求消息头 | StringToSign |
| --- | --- |
| GET /objectkey?<br>AccessKeyId=MFyfvK41ba2giqM7Uio6PznpdUKGpownRZlmVmHc&Expires=1532779451&Signature=0Akylf43Bm3mD1bh2rM3dmVp1Bo%3D&x-obs-security-token=YwkaRTbdY8g7q....<br>HTTP/1.1<br>Host:<br>examplebucket.obs.*region*.example.com | GET \n<br>\n<br>\n<br>1532779451\n<br>/examplebucket/objectkey?x-obs-security-token=YwkaRTbdY8g7q.... |

根据签名计算规则

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKeyID, UTF-8-Encoding-Of( StringToSign ) ) ) )
```

计算出签名，然后将Host作为URL的前缀，可以生成预定义的URL:

http(s)://examplebucket.obs.*region*.example.com/objectkey?AccessKeyId=AccessKeyID&Expires=1532779451&Signature=0Akylf43Bm3mD1bh2rM3dmVp1Bo%3D

在浏览器中直接输入该地址则可以下载examplebucket桶中的objectkey对象。这个链接的有效期是1532779451(Sat Jul 28 20:04:11 CST 2018)。

在Linux环境上使用curl命令访问注意&字符需要\转义，如下命令将对象objectkey下载到output文件中：

curl http(s)://examplebucket.obs.*region*.example.com/objectkey?AccessKeyId=AccessKeyID\&Expires=1532779451\&Signature=0Akylf43Bm3mD1bh2rM3dmVp1Bo%3D -X GET -o output

**说明**

如果想要在浏览器中使用URL中携带签名生成的预定义URL，则计算签名时不要使用只能携带在头域部分的“Content-MD5”、“Content-Type”、“CanonicalizedHeaders”来计算签名。否则浏览器不能携带这些参数，请求发送到服务端之后，会提示签名错误。

## Java 中签名的计算方法

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.TreeMap;
import java.util.regex.Pattern;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class SignDemo {

    private static final String SIGN_SEP = "\n";

    private static final String OBS_PREFIX = "x-obs-";

    private static final String DEFAULT_ENCODING = "UTF-8";

    private static final List<String> SUB_RESOURCES = Collections.unmodifiableList(Arrays.asList(
            "CDNNotifyConfiguration", "acl", "append", "attname",  "cors", "customdomain", "delete",
            "deletebucket", "inventory", "length", "lifecycle", "location", "logging",
            "metadata", "mirrorBackToSource", "modify", "name", "notification", "obscompresspolicy",
            "partNumber", "policy", "position", "quota","rename", "replication", "response-cache-control",
            "response-content-disposition","response-content-encoding", "response-content-language",
"response-content-type",
            "response-expires","storagePolicy", "storageinfo", "tagging", "torrent", "truncate",
            "uploadId", "uploads", "versionId", "versioning", "versions", "website",
            "x-obs-security-token"));

    private String ak;

    private String sk;

    private boolean isBucketNameValid(String bucketName) {
        if (bucketName == null || bucketName.length() > 63 || bucketName.length() < 3) {
            return false;
        }

        if (!Pattern.matches("^[a-z0-9][a-z0-9.-]+$", bucketName)) {
            return false;
        }

        if (Pattern.matches("(\\d{1,3}\\.){3}\\d{1,3}", bucketName)) {
            return false;
        }

        String[] fragments = bucketName.split("\\.");
        for (int i = 0; i < fragments.length; i++) {
            if (Pattern.matches("^-.*", fragments[i]) || Pattern.matches(".*-$", fragments[i])
                    || Pattern.matches("^$", fragments[i])) {
                return false;
            }
```

```
        }

        return true;
    }

    public String encodeUrlString(String path) throws UnsupportedEncodingException {
        return URLEncoder.encode(path, DEFAULT_ENCODING)
                .replaceAll("\\+", "%20")
                .replaceAll("\\*", "%2A")
                .replaceAll("%7E", "~");
    }

    public String encodeObjectName(String objectName) throws UnsupportedEncodingException {
        StringBuilder result = new StringBuilder();
        String[] tokens = objectName.split("/");
        for (int i = 0; i < tokens.length; i++) {
            result.append(this.encodeUrlString(tokens[i]));
            if (i < tokens.length - 1) {
                result.append("/");
            }
        }
        return result.toString();
    }

    private String join(List<?> items, String delimiter) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < items.size(); i++) {
            String item = items.get(i).toString();
            sb.append(item);
            if (i < items.size() - 1) {
                sb.append(delimiter);
            }
        }
        return sb.toString();
    }

    private boolean isValid(String input) {
        return input != null && !input.equals("");
    }

    public String hmacSha1(String input) throws NoSuchAlgorithmException, InvalidKeyException,
UnsupportedEncodingException {
        SecretKeySpec signingKey = new SecretKeySpec(this.sk.getBytes(DEFAULT_ENCODING), "HmacSHA1");
        Mac mac = Mac.getInstance("HmacSHA1");
        mac.init(signingKey);
        return Base64.getEncoder().encodeToString(mac.doFinal(input.getBytes(DEFAULT_ENCODING)));
    }

    private String stringToSign(String httpMethod, Map<String, String[]> headers, Map<String, String>
queries,
                                String bucketName, String objectName, long expires) throws Exception {
        String contentMd5 = "";
        String contentType = "";
        TreeMap<String, String> canonicalizedHeaders = new TreeMap<String, String>();
        String key;
        List<String> temp = new ArrayList<String>();
        for (Map.Entry<String, String[]> entry : headers.entrySet()) {
            key = entry.getKey();
            if (key == null || entry.getValue() == null || entry.getValue().length == 0) {
                continue;
            }
            key = key.trim().toLowerCase(Locale.ENGLISH);
            if (key.equals("content-md5")) {
                contentMd5 = entry.getValue()[0];
                continue;
            }
            if (key.equals("content-type")) {
                contentType = entry.getValue()[0];
                continue;
```

```
            }
            if (key.startsWith(OBS_PREFIX)) {
                for (String value : entry.getValue()) {
                    if (value != null) {
                        temp.add(value.trim());
                    }
                }
                canonicalizedHeaders.put(key, this.join(temp, ","));
                temp.clear();
            }
        }
        // handle method/content-md5/content-type
        StringBuilder stringToSign = new StringBuilder();
        stringToSign.append(httpMethod).append(SIGN_SEP)
                .append(contentMd5).append(SIGN_SEP)
                .append(contentType).append(SIGN_SEP)
                .append(expires).append(SIGN_SEP);


        // handle canonicalizedHeaders
        for (Map.Entry<String, String> entry : canonicalizedHeaders.entrySet()) {
            stringToSign.append(entry.getKey()).append(":").append(entry.getValue()).append(SIGN_SEP);
        }


        // handle CanonicalizedResource
        stringToSign.append("/");
        if (this.isValid(bucketName)) {
            stringToSign.append(bucketName).append("/");
            if (this.isValid(objectName)) {
                stringToSign.append(this.encodeObjectName(objectName));
            }
        }

        TreeMap<String, String> canonicalizedResource = new TreeMap<String, String>();
        for (Map.Entry<String, String> entry : queries.entrySet()) {
            key = entry.getKey();
            if (key == null) {
                continue;
            }

            if (SUB_RESOURCES.contains(key)) {
                canonicalizedResource.put(key, entry.getValue());
            }
        }

        if (canonicalizedResource.size() > 0) {
            stringToSign.append("?");
            for (Map.Entry<String, String> entry : canonicalizedResource.entrySet()) {
                stringToSign.append(entry.getKey());
                if (this.isValid(entry.getValue())) {
                    stringToSign.append("=").append(entry.getValue());
                }
                stringToSign.append("&");
            }
            stringToSign.deleteCharAt(stringToSign.length() - 1);
        }
        //        System.out.println(String.format("StringToSign:%s%s", SIGN_SEP, stringToSign.toString()));

        return stringToSign.toString();
    }

    public String querySignature(String httpMethod, Map<String, String[]> headers, Map<String, String>
queries,
                                 String bucketName, String objectName, long expires) throws Exception {
        if (!isBucketNameValid(bucketName)) {
            throw new IllegalArgumentException("the bucketName is illegal");
        }
        //1. stringToSign
```

```
        String stringToSign = this.stringToSign(httpMethod, headers, queries, bucketName, objectName,
expires);

        //2. signature
        return this.encodeUrlString(this.hmacSha1(stringToSign));
    }

    public String getURL(String endpoint, Map<String, String> queries,
                         String bucketName, String objectName, String signature, long expires) throws
UnsupportedEncodingException {
        StringBuilder URL = new StringBuilder();
        URL.append("https://").append(bucketName).append(".").append(endpoint).append("/").
                append(this.encodeObjectName(objectName)).append("?");
        String key;
        for (Map.Entry<String, String> entry : queries.entrySet()) {
            key = entry.getKey();
            if (key == null) {
                continue;
            }
            if (SUB_RESOURCES.contains(key)) {
                String value = entry.getValue();
                URL.append(key);
                if (value != null) {
                    URL.append("=").append(value).append("&");
                } else {
                    URL.append("&");
                }
            }
        }
        URL.append("AccessKeyId=").append(this.ak).append("&Expires=").append(expires).
                append("&Signature=").append(signature);
        return URL.toString();
    }

    public static void main(String[] args) throws Exception {
        SignDemo demo = new SignDemo();

        /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文
存放，使用时解密，确保安全；
        本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和
YOUR_SK。*/
    demo.ak = System.getenv("YOUR_AK");
    demo.sk = System.getenv("YOUR_SK");
        String endpoint = "<your-endpoint>";

        String bucketName = "bucket-test";
        String objectName = "hello.jpg";

        // 如果直接使用URL在浏览器地址栏中访问，无法带上头域，此处headers加入头域会导致签名不匹配，使
用headers需要客户端处理
        Map<String, String[]> headers = new HashMap<String, String[]>();
        Map<String, String> queries = new HashMap<String, String>();

        // 请求消息参数Expires，设置24小时后失效
        long expires = (System.currentTimeMillis() + 86400000L) / 1000;
        String signature = demo.querySignature("GET", headers, queries, bucketName, objectName, expires);
        System.out.println(signature);
        String URL = demo.getURL(endpoint, queries, bucketName, objectName, signature, expires);
        System.out.println(URL);
    }
}
```

## 3.2.4 基于浏览器上传的表单中携带签名

OBS服务支持基于浏览器的POST上传对象请求，此类请求的签名信息通过表单的方式上传。POST上传对象：首先，创建一个安全策略，指定请求中需要满足的条件，比如：桶名、对象名前缀；然后，创建一个基于此策略的签名，需要签名的请求表单中必须包含有效的signature和policy；最后，创建一个表单将对象上传到桶中。

签名的计算过程如下：

1. 对policy内容进行UTF-8编码。
2. 对第一步的结果进行Base64编码。
3. 使用SK对第二步的结果进行HMAC-SHA1签名计算。
4. 对第三步的结果进行Base64编码，得到签名。

```
StringToSign = Base64( UTF-8-Encoding-Of( policy ) )
Signature = Base64( HMAC-SHA1( YourSecretAccessKeyID, StringToSign ) )
```

Policy的内容如下：

```
{ "expiration": "2017-12-31T12:00:00.000Z",
  "conditions": [
    {"x-obs-acl": "public-read" },
    {"x-obs-security-token": "YwkaRTbdY8g7q...." },
    {"bucket": "book" },
    ["starts-with", "$key", "user/"]
  ]
}
```

Policy策略中包含有效时间**Expiration**和条件元素**Conditions**。

## Expiration

描述本次签名的有效时间ISO 8601 UTC，如实例中"expiration": "2017-12-31T12:00:00.000Z"表示请求在2017年12月31日12点之后无效。该字段是policy中必选字段。合法格式仅有"yyyy-MM-dd'T'HH:mm:ss'Z'"和"yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"。

## Conditions

Conditions是一个用于验证本次请求合法的一种机制，可以使用这些条件限制请求中必须包含的内容。实例中的条件要求请求的桶名必须是book，对象名必须以user/为前缀，对象的acl必须是公共可读。除了AccessKeyId、signature、file、policy、token、field names以及前缀为x-ignore-外的表单中的所有项，都需要包含在policy中。下表是conditions中应该包含的项：

**表 3-16** policy 中应该包含的条件元素

| 元素名称 | 描述 |
| --- | --- |
| x-obs-acl | 请求中的ACL。<br>支持精确匹配和starts-with条件匹配。 |
| content-length-range | 设置上传对象的最大最小长度，支持range匹配。 |
| Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires | REST请求特定头域。<br>支持精确匹配和starts-with条件匹配。 |
| key | 上传对象的名字。<br>支持精确匹配和starts-with条件匹配。 |

| 元素名称 | 描述 |
| --- | --- |
| bucket | 请求桶名。<br>支持精确匹配。 |
| success_action_redirect | 上传对象成功后重定向的URL地址。具体描述请参见5.4.2-POST上传。<br>支持精确匹配和starts-with条件匹配。 |
| success_action_status | 如果未指定success_action_redirect，则成功上传时返回给客户端的状态码。具体描述请参见5.4.2-POST上传。<br>支持精确匹配。 |
| x-obs-meta-* | 用户自定义元数据。<br>元素中的关键字不允许含有非ASCII码或不可识别字符，如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。<br>支持精确匹配和starts-with条件匹配。 |
| x-obs-* | 其他以x-obs-为前缀的头域。<br>支持精确匹配和starts-with条件匹配。 |
| x-obs-security-token | 请求消息头中字段名。<br>临时AK/SK和securitytoken鉴权必加字段名。 |

Policy条件匹配的方式如下：

**表 3-17** policy 条件匹配方式

| 条件 | 描述 |
| --- | --- |
| Exact Matches | 默认是完全匹配，post表单中该项的值必须和policy的conditions中设置的值完全一样。例如：上传对象的同时设置对象ACL为public-read，表单中x-obs-acl元素的值为public-read，policy中的conditions可以设置为<br>{"x-obs-acl": "public-read" }或者[ "eq", "$x-obs-acl", "public-read"]，这两者是等效的。 |
| Starts With | 如果使用该条件，则post表单中对应元素的值必须是固定字符串开始。例如：上传对象名以user/为前缀，表单中key元素的值可以是user/test1、user/test2，policy的conditions中该条件如下：<br>["starts-with", "$key", "user/"] |
| Matching Any Content | post表单中对应元素的值可以是任意值。例如：请求成功后重定向的地址可以是任意地址，表单中success_action_redirect元素的值可以是任意值，policy的conditions中该条件如下：<br>["starts-with", "$success_action_redirect", ""] |

| 条件 | 描述 |
| --- | --- |
| Specifying Ranges | post表单中file元素文件的内容长度可以是一个指定的范围，只用于限制对象大小。例如上传对象大小为1-10MB，表单中file元素的内容长度可以是1048576-10485760，policy的conditions中该条件如下，注意值没有双引号：<br>["content-length-range", 1048576, 10485760] |

**说明**

policy使用json格式，conditions可以支持 { } 和 [ ] 两种方式，{ }中包含表单元素的key和value两项，以冒号分隔；[ ]中包含条件类型、key、value三项，以逗号分隔，元素key之前使用$字符表示变量。

Policy中必须转义的字符如下：

**表 3-18** policy 中必须转义的字符

| 转义后的字符 | 真实字符 |
| --- | --- |
| \\ | 反斜杠(\) |
| \$ | 美元符号($) |
| \b | 退格 |
| \f | 换页 |
| \n | 换行 |
| \r | 回车 |
| \t | 水平制表 |
| \v | 垂直制表 |
| \u***xxxx*** | 所有Unicode字符 |

## 请求和 Policy 示例

下面的几张表提供了一些请求和Policy的例子。

**示例1**：在examplebucket桶中上传testfile.txt对象，并且设置对象ACL为公共可读。

| 请求 | policy |
| --- | --- |
| POST / HTTP/1.1<br>Host: examplebucket.obs.*region*.example.com<br>Content-Type: multipart/form-data; boundary=7e32233530b26<br>Content-Length: 1250<br>--7e32233530b26<br>Content-Disposition: form-data; name="key"<br>testfile.txt<br>--7e32233530b26<br>Content-Disposition: form-data; name="x-obs-acl"<br>public-read<br>--7e32233530b26<br>Content-Disposition: form-data; name="content-type"<br>text/plain<br>--7e32233530b26<br>Content-Disposition: form-data; name="AccessKeyId"<br>UDSIAMSTUBTEST000002<br>--7e32233530b26<br>Content-Disposition: form-data; name="policy"<br>ewogICJleHBpcmF0aW9uIjogIjIwMTktMDctMDFUMTI6MDA6MDAuMDAwWiIsCiAgImNvbmRpdGlvbnMiOiBbCiAgICB7ImJ1Y2tldCI6ICJleGFtcGxlYnVja2V0Ii B9LAogICAgWyJlcSIsICIka2V5IiwgInRlc3RmaWxlLnR4dCJdLAogICAgeyJ4LW9icy1hY2wiOiAicHVibGljLXJlYWQiIH0sCiAgICBbImVxIiwgIiRDb250ZW50LVR5cGUiLCAidGV4dC9wbGFpbiJdLAogICAgWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDYsIDEwXQogIF0KfQo=<br>--7e32233530b26<br>Content-Disposition: form-data; name="signature"<br>xxl7bZs/5FgtBUggOdQ88DPZUo0= | {<br>"expiration": "2019-07-01T12:00:00.000Z",<br>"conditions": [<br>{"bucket": "examplebucket" },<br>["eq", "$key", "testfile.txt"],<br>{"x-obs-acl": "public-read" },<br>["eq", "$Content-Type", "text/plain"]<br>]<br>} |

| 请求 | policy |
| --- | --- |
| --7e32233530b26<br>Content-Disposition: form-data; name="file"; filename="E:\TEST_FILE\TEST.txt"<br>Content-Type: text/plain<br>123456<br>--7e32233530b26<br>Content-Disposition: form-data; name="submit"<br>Upload<br>--7e32233530b26-- | |

**示例2**：在examplebucket桶中上传file/obj1对象，并且设置对象的四个自定义元数据。

| 请求 | policy |
| --- | --- |
| POST / HTTP/1.1<br>Host: examplebucket.obs.*region*.example.com<br>Content-Type: multipart/form-data; boundary=7e329d630b26<br>Content-Length: 1597<br>--7e3542930b26<br>Content-Disposition: form-data; name="key"<br>file/obj1<br>--7e3542930b26<br>Content-Disposition: form-data; name="AccessKeyId"<br>UDSIAMSTUBTEST000002<br>--7e3542930b26<br>Content-Disposition: form-data; name="policy"<br>ewogICJleHBpcmF0aW9uIjogIjIwMTktMDctMDFUMTI6MDA6MDAuMDAwWiIsCiAgImNvbmRpdGlvbnMiOiBbCiAgICB7ImJ1Y2tldCI6ICJleGFtcGxlYnVja2V0Ii B9LAogICAgWyJzdGFydHMtd2l0aCIsICIka2V5IiwgImZpbGUvIl0sCiAgICB7Ingtb2JzLW1ldGEtdGVzdDEiOiJ2YWx1ZTEifSwKICAgIFsiZXEiLCAiJHgtb2JzLW1ldGEtdGVzdDIiLCAidmFsdWUyIl0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LW9icy1tZXRhLXRlc3QzIiwgImRvYyJdLAogICAgWyJzdGFydHMtd2l0aCIsICIkeC1vYnMtbWV0YS10ZXN0NCIsICIiXQogIF0KfQo=<br>--7e3542930b26<br>Content-Disposition: form-data; name="signature"<br>HTId8hcaisn6FfdWKqSJP9RN4Oo=<br>--7e3542930b26<br>Content-Disposition: form-data; name="x-obs-meta-test1"<br>value1<br>--7e3542930b26<br>Content-Disposition: form-data; name="x-obs-meta-test2" | {<br>"expiration": "2019-07-01T12:00:00.000Z",<br>"conditions": [<br>{"bucket": "examplebucket" },<br>["starts-with", "$key", "file/"],<br>{"x-obs-meta-test1":"value1"},<br>["eq", "$x-obs-meta-test2", "value2"],<br>["starts-with", "$x-obs-meta-test3", "doc"],<br>["starts-with", "$x-obs-meta-test4", ""]<br>]<br>} |

| 请求 | policy |
| --- | --- |
| value2<br>--7e3542930b26<br>Content-Disposition: form-data; name="x-obs-meta-test3"<br>doc123<br>--7e3542930b26<br>Content-Disposition: form-data; name="x-obs-meta-test4"<br>my<br>--7e3542930b26<br>Content-Disposition: form-data; name="file"; filename="E:\TEST_FILE\TEST.txt"<br>Content-Type: text/plain<br>123456<br>--7e3542930b26<br>Content-Disposition: form-data; name="submit"<br>Upload<br>--7e3542930b26-- | |

# 3.3 返回结果

请求发送以后，您会收到响应，包含状态码、响应消息头和消息体。

## 状态码

状态码是一组从2xx（成功）到4xx或5xx（错误）的数字代码，状态码表示了请求响应的状态，完整的状态码列表请参见**状态码**。

## 响应消息头

对应请求消息头，响应同样也有消息头，如“Content-type”。

详细的公共响应消息头字段请参见**表3-19**。

**表 3-19** 公共响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| Content-Length | 响应消息体的字节长度。<br>类型：String<br>默认值：无。 |
| Connection | 指明与服务器的连接是长连接还是短连接。<br>类型：String<br>有效值：keep-alive \| close。<br>默认值：无。 |
| Date | OBS系统响应的时间。<br>类型：String<br>默认值：无。 |
| ETag | 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。实际的ETag是对象的哈希值。ETag只反映变化的内容，而不是其元数据。上传的对象或拷贝操作创建的对象，通过MD5加密后都有唯一的ETag。如果通过多段上传对象，则无论加密方法如何，MD5会拆分ETag，此类情况ETag就不是MD5的摘要。<br>类型：String |
| x-obs-id-2 | 帮助定位问题的特殊符号。<br>类型：String<br>默认值：无。 |
| x-reserved-indicator | 帮助定位问题的特殊符号。<br>类型：String<br>默认值：无。 |
| x-obs-request-id | 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。<br>类型：String<br>默认值：无。 |

## 响应消息体（可选）

响应消息体通常以结构化格式（如JSON或XML）返回，与响应消息头中Content-type对应，传递除响应消息头之外的内容。

# 4 快速入门

## 4.1 创建桶

### 操作场景

桶是OBS中存储对象的容器。您需要先创建一个桶，然后才能在OBS中存储数据。

下面介绍如何调用**创建桶**API在指定的区域创建一个桶，API的调用方法请参见**如何调用API**。

### 前提条件

- 已获取AK和SK，获取方法参见**获取访问密钥（AK/SK）**。
- 您需要规划桶所在的区域信息，并根据区域确定调用API的Endpoint，您可以向企业管理员获取区域和终端节点信息。

区域一旦确定，创建完成后无法修改。

### 在 a1 区域创建一个名为 bucket001 的桶

示例中使用通用的Apache Http Client。

```
package com.obsclient;

import java.io.*;

import org.apache.http.Header;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

public class TestMain {
    /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
    本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和YOUR_SK。*/
    public static String accessKey = System.getenv("YOUR_AK"); //取值为获取的AK
    public static String securityKey = System.getenv("YOUR_SK");  //取值为获取的SK
    public static String region = "a1"; // 取值为规划桶所在的区域
    public static String createBucketTemplate =
            "<CreateBucketConfiguration " +
```

```
            "xmlns=\"http://obs.a1.example.com/doc/2015-06-30/\">\n" +
            "<Location>" + region + "</Location>\n" +
            "</CreateBucketConfiguration>";

    public static void main(String[] str) {

        createBucket();

    }

    private static void createBucket() {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        String requesttime = DateUtils.formatDate(System.currentTimeMillis());
        String contentType = "application/xml";
        HttpPut httpPut = new HttpPut("http://bucket001.obs.a1.example.com");
        httpPut.addHeader("Date", requesttime);
        httpPut.addHeader("Content-Type", contentType);

        /** 根据请求计算签名**/
        String contentMD5 = "";
        String canonicalizedHeaders = "";
        String canonicalizedResource = "/bucket001/";
        // Content-MD5 、Content-Type 没有直接换行， data格式为RFC 1123，和请求中的时间一致
        String canonicalString = "PUT" + "\n" + contentMD5 + "\n" + contentType + "\n" + requesttime + "\n"
+ canonicalizedHeaders + canonicalizedResource;
        System.out.println("StringToSign:[" + canonicalString + "]");
        String signature = null;
        CloseableHttpResponse httpResponse = null;
        try {
            signature = Signature.signWithHmacSha1(securityKey, canonicalString);

            // 增加签名头域 Authorization: OBS AccessKeyID:signature
            httpPut.addHeader("Authorization", "OBS " + accessKey + ":" + signature);

            // 增加body体
            httpPut.setEntity(new StringEntity(createBucketTemplate));

            httpResponse = httpClient.execute(httpPut);

            // 打印发送请求信息和收到的响应消息
            System.out.println("Request Message:");
            System.out.println(httpPut.getRequestLine());
            for (Header header : httpPut.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }

            System.out.println("Response Message:");
            System.out.println(httpResponse.getStatusLine());
            for (Header header : httpResponse.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    httpResponse.getEntity().getContent()));

            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = reader.readLine()) != null) {
                response.append(inputLine);
            }
            reader.close();

            // print result
            System.out.println(response.toString());
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
```

```
            try {
                httpClient.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

}
```

**其中Date头域DateUtils的格式为：**

```
package com.obsclient;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.TimeZone;

public class DateUtils {

    public static String formatDate(long time)
    {
        DateFormat serverDateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z",
Locale.ENGLISH);
        serverDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
        return serverDateFormat.format(time);
    }
}
```

**签名字符串Signature的计算方法为：**

```
package com.obsclient;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.util.Base64;

public class Signature {
    public static String signWithHmacSha1(String sk, String canonicalString) throws
UnsupportedEncodingException {

        try {
            SecretKeySpec signingKey = new SecretKeySpec(sk.getBytes("UTF-8"), "HmacSHA1");
            Mac mac = Mac.getInstance("HmacSHA1");
            mac.init(signingKey);
            return Base64.getEncoder().encodeToString(mac.doFinal(canonicalString.getBytes("UTF-8")));
        } catch (NoSuchAlgorithmException | InvalidKeyException | UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

# 4.2 获取桶列表

## 操作场景

如果用户想要查看自己创建的所有桶信息，可以使用获取桶列表接口查看。

下面介绍如何调用**获取桶列表**API，API的调用方法请参见**如何调用API**。

## 前提条件

- 已获取AK和SK，获取方法参见**获取访问密钥（AK/SK）**。
- 您需要明确需要列举的桶所在的区域信息，并根据区域确定调用API的Endpoint，您可以向企业管理员获取区域和终端节点信息。

## 获取 a1 区域的桶列表

示例中使用通用的Apache Http Client。

```
package com.obsclient;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;


public class TestMain {

    /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
    本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和YOUR_SK。*/
    public static String accessKey = System.getenv("YOUR_AK"); //取值为获取的AK
    public static String securityKey = System.getenv("YOUR_SK");  //取值为获取的SK

    public static void main(String[] str) {

        listAllMyBuckets();

    }


    private static void listAllMyBuckets() {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        String requesttime = DateUtils.formatDate(System.currentTimeMillis());
        HttpGet httpGet = new HttpGet("http://obs.a1.example.com");
        httpGet.addHeader("Date", requesttime);

        /** 根据请求计算签名**/
        String contentMD5 = "";
        String contentType = "";
        String canonicalizedHeaders = "";
        String canonicalizedResource = "/";
        // Content-MD5 、Content-Type 没有直接换行， data格式为RFC 1123，和请求中的时间一致
        String canonicalString = "GET" + "\n" + contentMD5 + "\n" + contentType + "\n" + requesttime + "\n"
+ canonicalizedHeaders + canonicalizedResource;
        System.out.println("StringToSign:[" + canonicalString + "]");
        String signature = null;
        try {
            signature = Signature.signWithHmacSha1(securityKey, canonicalString);

            // 增加签名头域 Authorization: OBS AccessKeyID:signature
```

```
            httpGet.addHeader("Authorization", "OBS " + accessKey + ":" + signature);
            CloseableHttpResponse httpResponse = httpClient.execute(httpGet);

            // 打印发送请求信息和收到的响应消息
            System.out.println("Request Message:");
            System.out.println(httpGet.getRequestLine());
            for (Header header : httpGet.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }

            System.out.println("Response Message:");
            System.out.println(httpResponse.getStatusLine());
            for (Header header : httpResponse.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    httpResponse.getEntity().getContent()));

            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = reader.readLine()) != null) {
                response.append(inputLine);
            }
            reader.close();
            // print result
            System.out.println(response.toString());
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                httpClient.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

}
```

**其中Date头域DateUtils的格式为：**

```
package com.obsclient;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.TimeZone;

public class DateUtils {

    public static String formatDate(long time)
    {
        DateFormat serverDateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z",
Locale.ENGLISH);
        serverDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
        return serverDateFormat.format(time);
    }
}
```

**签名字符串Signature的计算方法为：**

```
package com.obsclient;
```

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.util.Base64;

public class Signature {
    public static String signWithHmacSha1(String sk, String canonicalString) throws
UnsupportedEncodingException {

        try {
            SecretKeySpec signingKey = new SecretKeySpec(sk.getBytes("UTF-8"), "HmacSHA1");
            Mac mac = Mac.getInstance("HmacSHA1");
            mac.init(signingKey);
            return Base64.getEncoder().encodeToString(mac.doFinal(canonicalString.getBytes("UTF-8")));
        } catch (NoSuchAlgorithmException | InvalidKeyException | UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

# 4.3 上传对象

## 操作场景

您可以根据需要，将任何类型的文件上传到OBS桶中进行存储。

下面介绍如何调用**PUT上传**API在指定的桶中上传对象，API的调用方法请参见**如何调用API**。

## 前提条件

- 已获取AK和SK，获取方法参见**获取访问密钥（AK/SK）**。
- 已创建了至少一个可用的桶。
- 已准备好了待上传的文件，并清楚文件所在的本地完整路径。
- 您需要知道待上传桶所在的区域信息，并根据区域确定调用API的Endpoint，您可以向企业管理员获取区域和终端节点信息。

## 向 a1 区域的桶 bucket001 中上传对象，名称为 objecttest1

示例中使用通用的Apache Http Client。

```
package com.obsclient;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
```

```
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;


public class TestMain {

    /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
    本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量YOUR_AK和YOUR_SK。*/
    public static String accessKey = System.getenv("YOUR_AK"); //取值为获取的AK
    public static String securityKey = System.getenv("YOUR_SK");  //取值为获取的SK

    public static void main(String[] str) {

        putObjectToBucket();

    }


    private static void putObjectToBucket() {

        InputStream inputStream = null;
        CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse httpResponse = null;
        String requestTime = DateUtils.formatDate(System.currentTimeMillis());
        HttpPut httpPut = new HttpPut("http://bucket001.obs.a1.example.com/objecttest1");
        httpPut.addHeader("Date", requestTime);

        /** 根据请求计算签名 **/
        String contentMD5 = "";
        String contentType = "";
        String canonicalizedHeaders = "";
        String canonicalizedResource = "/bucket001/objecttest1";
        // Content-MD5 、Content-Type 没有直接换行， data格式为RFC 1123，和请求中的时间一致
        String canonicalString = "PUT" + "\n" + contentMD5 + "\n" + contentType + "\n" + requestTime + "\n"
+ canonicalizedHeaders + canonicalizedResource;
        System.out.println("StringToSign:[" + canonicalString + "]");
        String signature = null;
        try {
            signature = Signature.signWithHmacSha1(securityKey, canonicalString);
            // 上传的文件目录
            inputStream = new FileInputStream("D:\\OBSobject\\text01.txt");
            InputStreamEntity entity = new InputStreamEntity(inputStream);
            httpPut.setEntity(entity);

            // 增加签名头域 Authorization: OBS AccessKeyID:signature
            httpPut.addHeader("Authorization", "OBS " + accessKey + ":" + signature);
            httpResponse = httpClient.execute(httpPut);

            // 打印发送请求信息和收到的响应消息
            System.out.println("Request Message:");
            System.out.println(httpPut.getRequestLine());
            for (Header header : httpPut.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }

            System.out.println("Response Message:");
            System.out.println(httpResponse.getStatusLine());
            for (Header header : httpResponse.getAllHeaders()) {
                System.out.println(header.getName() + ":" + header.getValue());
            }
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    httpResponse.getEntity().getContent()));

            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = reader.readLine()) != null) {
```

```
            response.append(inputLine);
        }
        reader.close();

        // print result
        System.out.println(response.toString());


    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            httpClient.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
  }

}
```

**其中Date头域DateUtils的格式为：**

```
package com.obsclient;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.TimeZone;

public class DateUtils {

   public static String formatDate(long time)
   {
      DateFormat serverDateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z",
Locale.ENGLISH);
      serverDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
      return serverDateFormat.format(time);
   }
}
```

**签名字符串Signature的计算方法为：**

```
package com.obsclient;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.util.Base64;

public class Signature {
   public static String signWithHmacSha1(String sk, String canonicalString) throws
UnsupportedEncodingException {

      try {
         SecretKeySpec signingKey = new SecretKeySpec(sk.getBytes("UTF-8"), "HmacSHA1");
         Mac mac = Mac.getInstance("HmacSHA1");
         mac.init(signingKey);
         return Base64.getEncoder().encodeToString(mac.doFinal(canonicalString.getBytes("UTF-8")));
      } catch (NoSuchAlgorithmException | InvalidKeyException | UnsupportedEncodingException e) {
         e.printStackTrace();
      }
      return null;
```

```
  }
}
```

# 5 API

## 5.1 桶的基础操作

### 5.1.1 获取桶列表

#### 功能介绍

OBS用户可以通过请求查询自己创建的桶列表。

#### 请求消息样式

```
GET / HTTP/1.1
Host: obs.region.example.com
Date: date
Authorization: authorization
```

#### 请求消息参数

该请求消息中不带请求参数。

#### 请求消息头

该操作消息头与普通请求一样，请参见表3-3，但可以带附加消息头，附加请求消息头如下所示。

**表 5-1** 附加请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-bucket-type | 通过此消息头明确获取的列表内容。<br>取值：<br>● OBJECT：获取所有桶列表。<br>● POSIX：获取所有并行文件系统列表。<br>不带此消息头则获取所有桶和并行文件系统列表。<br>示例：x-obs-bucket-type: POSIX | 否 |

## 请求消息元素

该请求消息中不带请求元素。

## 响应消息样式

```
GET HTTP/1.1 status_code
Content-Type: type
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Owner>
      <ID>id</ID>
   </Owner>
   <Buckets>
      <Bucket>
         <Name>bucketName</Name>
         <CreationDate>date</CreationDate>
         <Location>region</Location>
      </Bucket>
      ...
   </Buckets>
</ListAllMyBucketsResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中，会以XML形式将用户拥有的桶列出来，元素的具体含义如**表5-2**所示。

**表 5-2** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ListAllMyBucketsResult | 用户的桶列表。<br>类型：XML |

| 元素名称 | 描述 |
| --- | --- |
| Owner | 桶拥有者信息，包含租户ID。<br>类型：XML |
| ID | 用户的DomainID（账号ID）。<br>类型：String |
| Buckets | 用户所拥有的桶列表。<br>类型：XML |
| Bucket | 具体的桶信息。<br>类型：XML |
| Name | 桶名称。<br>类型：String |
| CreationDate | 桶的创建时间。<br>类型：String |
| Location | 桶的位置信息。<br>类型：String |

## 错误响应消息

该请求无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET / HTTP/1.1
User-Agent: curl/7.29.0
Host: obs.region.example.com
Accept: */*
Date: Mon, 25 Jun 2018 05:37:12 +0000
Authorization: OBS GKDF4C7Q6SI0IPGTXTJN:9HXkVQIiQKw33UEmyBI4rWrzmic=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435722C11379647A8A00A
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSGGDRUM62QZi3hGP8Fz3gOloYCfZ39U
Content-Type: application/xml
Date: Mon, 25 Jun 2018 05:37:12 GMT
Content-Length: 460

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Owner>
    <ID>783fc6652cf246c096ea836694f71855</ID>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>examplebucket01</Name>
      <CreationDate>2018-06-21T09:15:01.032Z</CreationDate>
      <Location>region</Location>
    </Bucket>
    <Bucket>
```

```
    <Name>examplebucket02</Name>
    <CreationDate>2018-06-22T03:56:33.700Z</CreationDate>
    <Location>region</Location>
  </Bucket>
 </Buckets>
</ListAllMyBucketsResult>
```

# 5.1.2 创建桶

## 功能介绍

创建桶是指按照用户指定的桶名创建一个新桶的操作。

**说明**

- 默认情况下，一个用户可以拥有的桶的数量不能超过100个。
- 用户删除桶后，需要等待30分钟才能创建同名桶和并行文件系统。
- OBS支持在创建桶时指定桶的AZ类型，您可以开启或关闭多AZ。关闭多AZ时，桶内数据默认存储在单个AZ内；开启多AZ时，桶内数据冗余存储在多个AZ内，可靠性更高。旧桶AZ类型默认为单AZ。

新创建桶的桶名在OBS中必须是唯一的。如果是同一个用户重复创建同一区域的同名桶时返回成功。除此以外的其他场景重复创建同名桶返回桶已存在。用户可以在请求消息头中加入x-obs-acl等参数，设置要创建桶的权限控制策略。

## 请求消息样式

```
PUT / HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Length: length
Date: date
Authorization: authorization
x-obs-az-redundancy: 3az

<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Location>location</Location>
</CreateBucketConfiguration>
```

## 请求消息参数

该请求消息中不带请求参数。

## 请求消息头

该操作消息头与普通请求一样，请参见表3-3，但可以带附加消息头，附加请求消息头如下所示。

**表 5-3** 附加请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-acl | 创建桶时，可以加上此消息头设置桶的权限控制策略，使用的策略为预定义的常用策略，包括：private、public-read、public-read-write、public-read-delivered、public-read-write-delivered。<br>类型：String | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-grant-read | 授权给指定domain下的所有用户有READ权限。允许列举桶内对象、列举桶中多段任务、列举桶中多版本对象、获取桶元数据。<br>类型：String<br>示例：x-obs-grant-read:id=租户id | 否 |
| x-obs-grant-write | 授权给指定domain下的所有用户有WRITE权限。允许创建、删除、覆盖桶内所有对象，允许初始化段、上传段、拷贝段、合并段、取消多段上传任务。<br>类型：String<br>示例：x-obs-grant-write:id=租户id | 否 |
| x-obs-grant-read-acp | 授权给指定domain下的所有用户有READ_ACP权限。允许读桶的ACL信息。<br>类型：String<br>示例：x-obs-grant-read-acp:id=租户id | 否 |
| x-obs-grant-write-acp | 授权给指定domain下的所有用户有WRITE_ACP权限，允许修改桶的ACL信息。<br>类型： 字符串<br>示例：x-obs-grant-write-acp:id=租户id | 否 |
| x-obs-grant-full-control | 授权给指定domain下的所有用户有FULL_CONTROL权限。<br>类型： 字符串<br>示例：x-obs-grant-full-control:id=租户id | 否 |
| x-obs-grant-read-delivered | 授权给指定domain下的所有用户有READ权限，并且在默认情况下，该READ权限将传递给桶内所有对象。<br>类型：String<br>示例：x-obs-grant-read-delivered:id=租户id | 否 |
| x-obs-grant-full-control-delivered | 授权给指定domain下的所有用户有FULL_CONTROL权限，并且在默认情况下，该FULL_CONTROL权限将传递给桶内所有对象。<br>类型：String<br>示例：x-obs-grant-full-control-delivered:id=租户id | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-az-redundancy | 创建桶时带上此消息头设置桶的存储类型为多AZ。不携带时默认为单AZ。用户携带该头域指定新创的桶的存储类型为多AZ，存在一种情况是当该区域如果不支持多AZ存储，则该桶的存储类型仍为单AZ。<br>类型：String<br>示例：x-obs-az-redundancy：3az | 否 |
| x-obs-fs-file-interface | 创建桶时可以带上此消息头以创建并行文件系统。<br>类型：String<br>示例：x-obs-fs-file-interface:Enabled | 否 |

## 请求消息元素

该操作可以带附加请求消息元素，附加请求消息元素的具体描述如表5-4所示。

**表 5-4** 附加请求消息元素

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Location | 指定Bucket在哪个区域被创建。<br>• 使用默认区域的终端节点创桶时<br>– 不携带Location，桶将默认创建在默认区域<br>– 在Location中指定其它区域，桶将创建在指定区域<br>• 使用非默认区域的终端节点创桶时，必须携带Location，并且Location只能指定为该终端节点对应的区域。<br>类型：String | 否 |

## 响应消息样式

```
HTTP/1.1 status_code
Location: location
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例：创建桶

```
PUT / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:25:05 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:75/Y4Ng1izvzc1nTGxpMXTE6ynw=
Content-Length: 157

<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
</CreateBucketConfiguration>
```

## 响应示例：创建桶

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435CE298386946AE4C482
Location: /examplebucket
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCT9W2tcvLmMJ+plfdopaD62S0npbaRUz
Date: WED, 01 Jul 2015 02:25:06 GMT
Content-Length: 0
```

## 请求示例：创建指定 ACL 的桶

```
PUT / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:25:05 GMT
x-obs-acl:public-read
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:75/Y4Ng1izvzc1nTGxpMXTE6ynw=
Content-Length: 157

<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
</CreateBucketConfiguration>
```

## 响应示例：创建指定 ACL 的桶

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435CE298386946AE4C482
Location: /examplebucket
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCT9W2tcvLmMJ+plfdopaD62S0npbaRUz
Date: WED, 01 Jul 2015 02:25:06 GMT
Content-Length: 0
```

## 请求示例：创建桶时选择多 AZ

```
PUT / HTTP/1.1
Host: examplebucket.obs.region.example.com
Content-Length: length
```

```
Date: date
Authorization: authorization
x-obs-az-redundancy:3az
<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
</CreateBucketConfiguration>
```

### 响应示例：创建桶时选择多 AZ

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435CE298386946AE4C482
Location: /examplebucket
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCT9W2tcvLmMJ+plfdopaD62S0npbaRUz
Date: WED, 01 Jul 2015 02:25:06 GMT
x-obs-az-redundancy:3az
Content-Length: 0
```

### 请求示例：创建并行文件系统

```
PUT / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:25:05 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:75/Y4Ng1izvzc1nTGxpMXTE6ynw=
Content-Length: 157
x-obs-fs-file-interface: Enabled

<CreateBucketConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
</CreateBucketConfiguration>
```

### 响应示例：创建并行文件系统

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435CE298386946AE4C482
Location: /examplebucket
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCT9W2tcvLmMJ+plfdopaD62S0npbaRUz
Date: WED, 01 Jul 2015 02:25:06 GMT
Content-Length: 0
```

## 5.1.3 列举桶内对象

### 功能介绍

对桶拥有读权限的用户可以执行获取桶内对象列表的操作。

如果用户在请求中只指定了桶名，则返回信息中会包含桶内部分或所有对象的描述信息（一次最多返回1000个对象信息）；如果用户还指定了prefix、marker、max-keys、delimiter参数中的一个或多个，则返回的对象列表将按照如**表5-5**所示规定的语义返回指定的对象。

用户也可以请求参数中添加versions参数来执行列举桶内多版本对象的操作。

### 请求消息样式

```
GET / HTTP/1.1

Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息样式（多版本）

```
GET /?versions HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求可以通过带参数，列举出桶内的一部分对象，参数的具体含义如表5-5所示。

**表 5-5** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| prefix | 列举以指定的字符串prefix开头的对象。<br>类型：String | 否 |
| marker | 列举桶内对象列表时，指定一个标识符，从该标识符以后按字典顺序返回对象列表。该字段仅用于非多版本列举。<br>类型：String | 否 |
| max-keys | 指定返回的最大对象数，返回的对象列表将是按照字典顺序的最多前max-keys个对象，范围是[1，1000]，超出范围时，按照默认的1000进行处理。<br>类型：Integer | 否 |
| delimiter | 将对象名进行分组的分隔符。如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条CommonPrefixes；如果没有指定prefix，从对象名的首字符到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条CommonPrefixes。<br>例如，桶中有3个对象，分别为abcd、abcde、bbcde。如果指定delimiter为d，prefix为a，abcd、abcde会被分成一组，形成一条前缀为abcd的CommonPrefixes；如果只指定delimiter为d，abcd、abcde会被分成一组，形成一条前缀为abcd的CommonPrefixes，而bbcde会被单独分成一组，形成一条前缀为bbcd的CommonPrefixes。<br>类型：String | 否 |
| key-marker | 列举对象时的起始位置。该字段仅用于多版本列举。<br>类型：String<br>有效值：上次请求返回体的NextKeyMarker值 | 否 |

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| version-id-marker | 本参数只适用于多版本列举场景<br>与请求中的key-marker配合使用，返回的对象列表将是按照字典顺序排序后在该标识符以后的对象(单次返回最大为1000个)。如果version-id-marker不是key-marker对应的一个版本号，则该参数无效。<br>类型：String<br>有效值：对象的版本号，即上次请求返回体的NextVersionIdMarker值 | 否 |

## 请求消息头

该请求使用公共的请求消息头，具体如表3-3所示。

## 请求消息元素

该请求消息头中不带消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: application/xml
Content-Length: length
<Response Body>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

该请求的响应消息中，会以XML形式将桶中的对象列出来，元素的具体含义如表5-6所示。

**表 5-6** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ListBucketResult | 桶中对象列表。<br>类型：XML |
| Contents | 对象的元数据信息。<br>类型：XML<br>父节点：ListBucketResult |

| 元素名称 | 描述 |
| --- | --- |
| CommonPrefixes | 请求中带delimiter参数时，返回消息带CommonPrefixes分组信息。<br>类型：XML<br>父节点：ListBucketResult |
| Delimiter | 请求中携带的delimiter参数。<br>类型：String<br>父节点：ListBucketResult |
| ETag | 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。实际的ETag是对象的哈希值。ETag只反映变化的内容，而不是其元数据。上传的对象或拷贝操作创建的对象，通过MD5加密后都有唯一的ETag。（当对象是服务端加密的对象时，ETag值不是对象的MD5值，而是通过服务端加密计算出的唯一标识。）<br>类型：String<br>父节点：ListBucketResult.Contents |
| Type | 对象类型，非Normal对象时返回。<br>类型：String<br>父节点：ListBucketResult.Contents |
| ID | 对象拥有者的DomainId。<br>类型：String<br>父节点：ListBucketResult.Contents.Owner |
| IsTruncated | 表明是否本次返回的ListBucketResult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：Boolean<br>父节点：ListBucketResult |
| Key | 对象名。<br>类型：String<br>父节点：ListBucketResult.Contents |
| LastModified | 对象最近一次被修改的时间（UTC时间）。<br>类型：Date<br>父节点：ListBucketResult.Contents |
| Marker | 列举对象时的起始位置标识符。<br>类型：String<br>父节点：ListBucketResult |

| 元素名称 | 描述 |
| --- | --- |
| NextMarker | 如果本次没有返回全部结果，响应请求中将包含此字段，用于标明本次请求列举到的最后一个对象。后续请求可以指定Marker等于该值来列举剩余的对象。<br>类型：String<br>父节点：ListBucketResult |
| MaxKeys | 列举时最多返回的对象个数。<br>类型：String<br>父节点：ListBucketResult |
| Name | 本次请求的桶名。<br>类型：String<br>父节点：ListBucketResult |
| Owner | 用户信息，包含对象拥有者DomainId和对象拥有者名称。<br>类型：XML<br>父节点：ListBucketResult.Contents |
| DisplayName | 对象拥有者名称。<br>类型：String<br>父节点：ListBucketResult.Contents.Owner |
| Prefix | 对象名的前缀，表示本次请求只列举对象名能匹配该前缀的所有对象。<br>类型：String<br>父节点：ListBucketResult |
| Size | 对象的字节数。<br>类型：String<br>父节点：ListBucketResult.Contents |

**表 5-7** 列举多版本对象响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ListVersionsResult | 保存列举桶中对象列表（含多版本）请求结果的容器。<br>类型：Container |
| Name | 桶名。<br>类型：String<br>父节点：ListVersionsResult |

| 元素名称 | 描述 |
| --- | --- |
| Prefix | 对象名的前缀，表示本次请求只列举对象名能匹配该前缀的所有对象。类型：String<br>父节点：ListVersionsResult |
| KeyMarker | 列举对象时对象的起始位置标识符。<br>类型：String<br>父节点：ListVersionsResult |
| VersionIdMarker | 列举对象时对象版本号的起始位置。<br>类型：String<br>父节点：ListVersionsResult |
| NextKeyMarker | 如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的KeyMarker值。<br>类型：String<br>父节点: ListVersionsResult。 |
| NextVersionIdMarker | 如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的VersionIdMarker值。<br>类型：String<br>父节点: ListVersionsResult。 |
| MaxKeys | 列举时最多返回的对象个数。<br>类型：String<br>父节点：ListVersionsResult |
| IsTruncated | 表明是否本次返回的ListVersionsResult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：Boolean<br>父节点：ListVersionsResult |
| Version | 保存版本信息的容器<br>类型：Container<br>父节点：ListVersionsResult |
| DeleteMarker | 保存删除标记的容器<br>类型：Container<br>父节点：ListVersionsResult |

| 元素名称 | 描述 |
| --- | --- |
| Key | 对象名。<br>类型：String<br>父节点：ListVersionsResult.Version \| ListVersionsResult.DeleteMarker |
| VersionId | 对象的版本号。<br>类型：String<br>父节点：ListVersionsResult.Version \| ListVersionsResult.DeleteMarker |
| IsLatest | 标识对象是否是最新的版本，true代表是最新的版本。<br>类型：Boolean<br>父节点：ListVersionsResult.Version \| ListVersionsResult.DeleteMarker |
| LastModified | 对象最近一次被修改的时间（UTC时间）。<br>类型：Date<br>父节点：ListVersionsResult.Version \| ListVersionsResult.DeleteMarker |
| ETag | 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。实际标签是对象的哈希。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或拷贝操作创建的对象，通过MD5加密后都有唯一的ETag。<br>类型：String<br>父节点：ListVersionsResult.Version |
| Type | 对象类型，非Normal对象时返回。<br>类型：String<br>父节点：ListVersionsResult.Version |
| Size | 对象的字节数。<br>类型：String<br>父节点：ListVersionsResult.Version |
| Owner | 用户信息，包含对象拥有者DomainId和对象拥有者名称。<br>类型：Container<br>父节点：ListVersionsResult.Version \| ListVersionsResult.DeleteMarker |

| 元素名称 | 描述 |
| --- | --- |
| ID | 对象拥有者的DomainId。<br>类型：String<br>父节点：ListVersionsResult.Version.Owner \| ListVersionsResult.DeleteMarker.Owner |
| DisplayName | 对象拥有者名称。<br>类型：String<br>父节点：ListVersionsResult.Version.Owner \| ListVersionsResult.DeleteMarker.Owner |
| CommonPrefixes | 请求中带delimiter参数时，返回消息带CommonPrefixes分组信息。<br>类型：Container<br>父节点: ListVersionsResult。 |
| Prefix | CommonPrefixes分组信息中，表明不同的Prefix。<br>类型：String<br>父节点: ListVersionsResult.CommonPrefixes。 |

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例：列举所有对象

```
GET / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:28:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:KiyoYze4pmRNPYfmlXBfRTVxt8c=
```

## 响应示例：列举所有对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435D34E379ABD93320CB9
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSXiN7GPL/yXM6OSBaYCUV1zcY5OelWp
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:23:30 GMT
Content-Length: 586

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Name>examplebucket</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
```

```
 <Contents>
   <Key>object001</Key>
   <LastModified>2015-07-01T00:32:16.482Z</LastModified>
   <ETag>"2fa3bcaaec668adc5da177e67a122d7c"</ETag>
   <Size>12041</Size>
   <Owner>
     <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
     <DisplayName>ObjectOwnerName</DisplayName>
   </Owner>
 </Contents>
</ListBucketResult>
```

## 请求示例：筛选对象

用户有桶名为examplebucket，桶内共有四个名为newfile，obj001，obj002，obs001的对象，如果只需要列出对象名为obj002的对象，请求消息格式为：

```
GET /?marker=obj001&prefix=obj HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:28:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:KiyoYze4pmRNPYfmlXBfRTVxt8c=
```

## 响应示例：筛选对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435D758FBA857E0801874
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCShn/xAyk/xHBX6qgGSB36WXrbco0X80
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:29:48 GMT
Content-Length: 707

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketResult xmlns="http://obs.example.com/doc/2015-06-30/">
<Name>examplebucket</Name>
<Prefix>obj</Prefix>
<Marker>obj001</Marker>
<MaxKeys>1000</MaxKeys>
<IsTruncated>false</IsTruncated>
 <Contents>
   <Key>obj002</Key>
   <LastModified>2015-07-01T02:11:19.775Z</LastModified>
   <ETag>"a72e382246ac83e86bd203389849e71d"</ETag>
   <Size>9</Size>
   <Owner>
     <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
     <DisplayName>ObjectOwnerName</DisplayName>
   </Owner>
 </Contents>
</ListBucketResult>
```

## 请求示例：多版本

```
GET /?versions HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:29:45 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:iZeDESIMxBK2YODk7vIeVpyO8Dl=
```

## 响应示例：多版本

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435D758FBA857E0801874
```

```
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCShn/xAyk/xHBX6qgGSB36WXrbco0X80
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:29:48 GMT
Content-Length: 707

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListVersionsResult xmlns="http://obs.example.com/doc/2015-06-30/">
 <Name>bucket02</Name>
  <Prefix/>
  <KeyMarker/>
  <VersionIdMarker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Version>
    <Key>object001</Key>
    <VersionId>000110000000000013F16000001643A22E476FFFF9046024ECA3655445346485a</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2015-07-01T00:32:16.482Z</LastModified>
    <ETag>"2fa3bcaaec668adc5da177e67a122d7c"</ETag>
    <Size>12041</Size>
    <Owner>
      <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
      <DisplayName>ObjectOwnerName</DisplayName>
    </Owner>
  </Version>
</ListVersionsResult>
```

## 5.1.4 获取桶元数据

### 功能介绍

对桶拥有读权限的用户可以执行查询桶元数据是否存在的操作。

### 请求消息样式

```
HEAD / HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不带消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

如果想要获取CORS配置信息，则需要使用的消息头如下**表1 获取CORS配置的请求消息头**所示。

**表 5-8** 获取 CORS 配置的请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Origin | 预请求指定的跨域请求Origin（通常为域名）。<br>类型：String | 是 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Access-Control-Request-Headers | 实际请求可以带的HTTP头域，可以带多个头域。<br>类型：String | 否 |

## 请求消息元素

该请求消息中不带消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
x-obs-bucket-location: region
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如下**表2 附加响应消息头**中的消息头。

**表 5-9** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-bucket-location | 桶的区域位置信息。<br>类型：String |
| x-obs-version | 桶所在的OBS服务版本号。<br>类型：String |
| x-obs-fs-file-interface | 是否为并行文件系统。取值包含Enabled（并行文件系统）。<br>不携带此头域表示不属于并行文件系统。<br>类型：String |
| x-obs-az-redundancy | 桶的数据冗余存储策略属性，决定数据是单AZ存储还是多AZ存储。<br>取值为3az，表示数据冗余存储在同一区域的多个可用区。<br>不携带此头域表示为单az存储，仅使用1个可用区存储。<br>类型：String |
| x-obs-location-clustergroup-id | 桶所在的集群组ID。<br>类型：String |

| 消息头名称 | 描述 |
| --- | --- |
| Access-Control-Allow-Origin | 当桶设置了CORS配置，如果请求的Origin满足服务端的CORS配置，则在响应中包含这个Origin。<br>类型：String |
| Access-Control-Allow-Headers | 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。<br>类型：String |
| Access-Control-Max-Age | 当桶设置了CORS配置，服务端CORS配置中的MaxAgeSeconds。<br>类型：Integer |
| Access-Control-Allow-Methods | 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE |
| Access-Control-Expose-Headers | 当桶设置了CORS配置，服务端CORS配置中的ExposeHeader。<br>类型：String |

## 响应消息元素

该请求的响应中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例：未携带获取 CORS 配置

```
HEAD / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:30:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:niCQCuGIZpETKIyx1datxHZyYlk=
```

## 响应示例：未携带获取 CORS 配置

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016439C734E0788404623FA8
Content-Type: application/xml
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSxwLpq9Hzf3OnaXr+pI/OPLKdrtiQAF
Date: WED, 01 Jul 2015 02:30:25 GMT
x-obs-bucket-location: region
```

```
x-obs-version: 3.0
Content-Length: 0
```

### 请求示例：桶设置了 CORS 后，获取桶元数据和 CORS 配置

```
HEAD / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:30:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:niCQCuGIZpETKIyx1datxHZyYlk=
Origin:www.example.com
Access-Control-Request-Headers:AllowedHeader_1
```

### 响应示例：桶设置了 CORS 后，获取桶元数据和 CORS 配置

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016439C734E0788404623FA8
Content-Type: application/xml
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSxwLpq9Hzf3OnaXr+pI/OPLKdrtiQAF
Date: WED, 01 Jul 2015 02:30:25 GMT
x-obs-bucket-location: region
Access-Control-Allow-Origin: www.example.com
Access-Control-Allow-Methods: POST,GET,HEAD,PUT
Access-Control-Allow-Headers: AllowedHeader_1
Access-Control-Max-Age: 100
Access-Control-Expose-Headers: ExposeHeader_1
x-obs-version: 3.0
Content-Length: 0
```

## 5.1.5 获取桶区域位置

### 功能介绍

对桶拥有读权限的用户可以执行获取桶区域位置信息的操作。

### 请求消息样式

```
GET /?location HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不带消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求消息中不带消息元素。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: type
```

```
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Location xmlns="http://obs.region.example.com/doc/2015-06-30/">region</Location>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该响应中将桶的区域信息以消息元素的形式返回，元素的具体含义如**表5-10**所示。

**表 5-10** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| Location | 桶的区域位置信息。<br>类型：String |

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
GET /?location HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:30:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:1DrmbCV+lhz3zV7uywlj7lrh0MY=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435D9F27CB2758E9B41A5
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSKWoJmaMyRXqofHgapbETDyI2LM9rUw
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:30:25 GMT
Content-Length: 128

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Location xmlns="http://obs.region.example.com/doc/2015-06-30/">region</Location>
```

# 5.1.6 删除桶

## 功能介绍

删除桶操作用于删除用户指定的桶。只有桶的所有者或者拥有桶的删桶policy权限的用户可以执行删除桶的操作，要删除的桶必须是空桶。如果桶中有对象或者有多段任务则认为桶不为空，可以使用列举桶内对象和列举出多段上传任务接口来确认桶是否为空。

注：

如果删除桶时，服务端返回5XX错误或超时，系统需要时间进行桶信息一致性处理，在此期间桶的信息会不准确，过一段时间再查看桶是否删除成功，查询到桶，需要再次发送删除桶消息。

## 请求消息样式

```
DELETE / HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共的请求消息头，具体请参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，错误已经包含在**表6-2**中。

## 请求示例

```
DELETE / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:31:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: BF260000016435DE6D67C35F9B969C47
x-obs-id-2: 32AAAQAAEAABKAAQAAEAABAAAQAAEAABCTukraCnXLsb7lEw4ZKjzDWWhzXdgme3
Date: WED, 01 Jul 2015 02:31:25 GMT
```

# 5.2 桶的高级配置

## 5.2.1 设置桶策略

### 功能介绍

该接口的实现使用policy子资源创建或者修改一个桶的策略。如果桶已经存在一个策略，那么当前请求中的策略将完全覆盖桶中现存的策略。单个桶的桶策略条数（statement）没有限制，但一个桶中所有桶策略的JSON描述总大小不能超过20KB。

要使用该接口，使用者要求必须是桶的所有者，或者是桶所有者的子用户且具有设置桶策略的权限。

### 请求消息样式

```
PUT /?policy HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: signatureValue
Policy written in JSON
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体请参见**表3-3**。

### 请求消息元素

请求消息体是一个符合JSON格式的字符串，包含了桶策略的信息。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带有响应元素。

### 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例 1

**向OBS租户授予权限**

给租户ID为783fc6652cf246c096ea836694f71855的租户授权。

如何获取租户ID请参考**获取账号ID和用户ID**。

```
PUT /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:32:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=

{
   "Statement": [
      {
         "Sid": "Stmt1375240018061",
         "Action": [
            "GetBucketLogging"
         ],
         "Effect": "Allow",
         "Resource": "logging.bucket",
         "Principal": {
            "ID": [
               "domain/783fc6652cf246c096ea836694f71855:user/*"
            ]
         }
      }
   ]
}
```

## 响应示例 1

```
HTTP/1.1 204 No Content
x-obs-request-id: 7B6DFC9BC71DD58B061285551605709
x-obs-id-2: N0I2REZDOUJDNzFERDU4QjA2MTI4NTU1MTYwNTcwOUFBQUFBQUFBYmJiYmJiYmJD
Date: WED, 01 Jul 2015 02:32:25 GMT
Content-Length: 0
Server: OBS
```

## 请求示例 2

**向OBS用户授予权限**

用户ID为71f3901173514e6988115ea2c26d1999，用户所属租户ID为783fc6652cf246c096ea836694f71855。

如何获取租户ID和用户ID请参考**获取账号ID和用户ID**。

```
PUT /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:33:28 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=

{
   "Statement": [
      {
         "Sid": "Stmt1375240018062",
         "Action": [
            "PutBucketLogging"
         ],
         "Effect": "Allow",
         "Resource": "examplebucket",
         "Principal": {
            "ID": [
               "domain/783fc6652cf246c096ea836694f71855:user/71f3901173514e6988115ea2c26d1999"
            ]
```

```
        }
      }
    ]
}
```

## 响应示例 2

```
HTTP/1.1 204 No Content
x-obs-request-id: 7B6DFC9BC71DD58B061285551605709
x-obs-id-2: N0I2REZDOUJDNzFERDU4QjA2MTI4NTU1MTYwNTcwOUFBQUFBQUFBYmJiYmJiYmJD
Date: WED, 01 Jul 2015 02:33:28 GMT
Content-Length: 0
Server: OBS
```

## 请求示例 3

**拒绝除了某个指定OBS用户的其他用户执行所有操作**

用户ID为71f3901173514e6988115ea2c26d1999，用户所属租户ID为783fc6652cf246c096ea836694f71855。

如何获取租户ID和用户ID请参考**获取账号ID和用户ID**。

```
PUT /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:34:34 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=

{
   "Statement": [
      {
         "Effect": "Deny",
         "Action": ["*"],
         "Resource": [
            "examplebucket/*",
            "examplebucket"
         ],
         "NotPrincipal": {
            "ID": [
               "domain/783fc6652cf246c096ea836694f71855:user/71f3901173514e6988115ea2c26d1999",
               "domain/783fc6652cf246c096ea836694f71855"
            ]
         }
      }
   ]
}
```

## 响应示例 3

```
HTTP/1.1 204 No Content
x-obs-request-id: A603000001604A7DFE4A4AF31E301891
x-obs-id-2: BKOvGmTlt6sda5X4G89PuMO4fabObGYmnpRGkaMba1LqPt0fCACEuCMllAObRK1n
Date: WED, 01 Jul 2015 02:34:34 GMT
Content-Length: 0
Server: OBS
```

## 请求示例 4

**拒绝除了某个指定的域名和不带referer头域的外链请求以实现防盗链白名单**

防盗链白名单：http://storage.example.com

```
PUT /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:34:34 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=
```

```
{
   "Statement": [{
      "Effect": "Deny",
      "Action": [
      "GetObject",
      "GetObjectVersion"
      ],
      "Principal": {
         "ID": ["*"]
      },
      "Resource": ["examplebucket/*"],
      "Condition": {
         "StringNotLike": {
            "Referer": [
            "http://storage.example.com*",
            "${null}"
            ]
         }
      }
   }]
}
```

### 响应示例 4

```
HTTP/1.1 204 No Content
x-obs-request-id: A603000001604A7DFE4A4AF31E301891
x-obs-id-2: BKOvGmTlt6sda5X4G89PuMO4fabObGYmnpRGkaMba1LqPt0fCACEuCMllAObRK1n
Date: WED, 01 Jul 2015 02:34:34 GMT
Content-Length: 0
Server: OBS
```

## 5.2.2 获取桶策略

### 功能介绍

该接口的实现使用policy子资源来将指定桶的策略返回给客户端。

要使用该接口，使用者要求必须是桶的所有者，或者是桶所有者的子用户且具有获取桶策略的权限。

以下两种场景无法使用此接口获取桶策略，系统将返回“404 NoSuchBucketPolicy”的错误：

- 指定桶的策略不存在
- 指定桶的标准桶策略为私有且未设置高级桶策略

### 请求消息样式

```
GET /?policy HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: application/xml
Date: date
Policy Content
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

响应消息体是一个JSON格式的桶策略字符串。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
GET /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:35:46 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=
```

## 响应示例

```
HTTP/1.1 200 OK
x-obs-request-id: A603000001604A7DFE4A4AF31E301891
x-obs-id-2: BKOvGmTlt6sda5X4G89PuMO4fabObGYmnpRGkaMba1LqPt0fCACEuCMllAObRK1n
Date: WED, 01 Jul 2015 02:35:46 GMT
Content-Length: 509
Server: OBS

{
   "Statement":[
      {
         "Sid":"Stmt1375240018061",
         "Effect":"Allow",
         "Principal":{
            "ID":[
               "domain/domainiddomainiddomainiddo006666:user/useriduseriduseriduseridus004001",
               "domain/domainiddomainiddomainiddo006667:user/*"
            ]
         },
         "Action":[
            "*"
         ],
         "Resource":[
            "examplebucket"
         ]
      }
   ]
}
```

# 5.2.3 删除桶策略

## 功能介绍

该接口的实现是通过使用policy子资源来删除一个指定桶上的策略。

要使用该接口，使用者要求必须是桶的所有者，或者是桶所有者的子用户且具有删除桶策略的权限。

无论桶的策略本身是否存在，删除成功后系统都直接返回“204 No Content”的结果。

## 请求消息样式

```
DELETE /?policy HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: text/xml
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
DELETE /?policy HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 02:36:06 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jZiAT8Vx4azWEvPRMWi0X5BpJMA=
```

## 响应示例

```
HTTP/1.1 204 No Content
x-obs-request-id: 9006000001643AAAF70BF6152D71BE8A
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSB4oWmNX3gVGGLr1cRPWjOhffEbq1XV
Date: WED, 01 Jul 2015 02:36:06 GMT
Server: OBS
```

# 5.2.4 设置桶 ACL

## 功能介绍

OBS支持对桶操作进行权限控制。默认情况下，只有桶的创建者才有该桶的读写权限。用户也可以设置其他的访问策略，比如对一个桶可以设置公共访问策略，允许所有人对其都有读权限。

OBS用户在创建桶时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的桶更改或者获取ACL(access control list）。一个桶的ACL最多支持100条Grant授权。PUT接口为幂等的覆盖写语意，新设置的桶ACL将覆盖原有的桶ACL，如果需要修改或者删除某条ACL重新PUT一个新的桶ACL即可。

## 请求消息样式

```
PUT /?acl HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
Content-Type: application/xml
Content-Length: length

<AccessControlPolicy>
   <Owner>
      <ID>ID</ID>
   </Owner>
   <AccessControlList>
      <Grant>
         <Grantee>
            <ID>domainId</ID>
         </Grantee>
         <Permission>permission</Permission>
         <Delivered>false</Delivered>
      </Grant>
   </AccessControlList>
</AccessControlPolicy>
```

## 请求消息参数

该操作请求不带消息参数。

## 请求消息头

使用者可以使用头域设置的方式来更改桶的ACL，每一种头域设置的ACL都有一套自己预先定义好的被授权用户以及相应权限，通过头域设置的方式授予访问权限，使用者必须添加以下的头域并且指定取值。

**表 5-11** 头域方式设置桶 ACL

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| x-obs-acl | 通过canned ACL的方式来设置桶的ACL。<br>取值范围：private \| public-read \| public-read-write \| public-read-delivered \| public-read-write-delivered<br>类型：String | 否 |

## 请求消息元素

更改桶的ACL请求需要在消息元素中带上ACL信息，元素的具体含义如表3-3所示。

**表 5-12** 附加请求消息元素

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Owner | 桶的所有者信息，包含ID。<br>类型：XML | 是 |
| ID | 被授权用户的租户Id。<br>类型：String | 是 |
| Grant | 用于标记用户及用户的权限。单个桶的ACL，Grant元素不能超过100个。<br>类型：XML | 否 |
| Grantee | 记录用户信息。<br>类型：XML | 否 |
| Canned | 向所有人授予权限。<br>取值范围：Everyone<br>类型：String | 否 |
| Delivered | 桶的ACL是否向桶内对象传递。作用于桶内所有对象。<br>类型：Boolean<br>默认：false | 否 |
| Permission | 授予的权限。<br>取值范围：READ \| READ_ACP \| WRITE \| WRITE_ACP \| FULL_CONTROL<br>类型：String | 否 |

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| AccessControlList | 访问控制列表，包含Grant、Grantee、Permission三个元素。<br>类型：XML | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?acl HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:37:22 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:iqSPeUBl66PwXDApxjRKk6hlcN4=
Content-Length: 727

<AccessControlPolicy xmlns="http://obs.example.com/doc/2015-06-30/">
  <Owner>
    <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <ID>783fc6652cf246c096ea836694f71855</ID>
      </Grantee>
      <Permission>READ</Permission>
      <Delivered>false</Delivered>
    </Grant>
    <Grant>
      <Grantee>
        <Canned>Everyone</Canned>
      </Grantee>
      <Permission>READ_ACP</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000164361F2954B4D063164704
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCT78HTIBuhe0FbtSptrb/akwELtwyPKs
Date: WED, 01 Jul 2015 02:37:22 GMT
Content-Length: 0
```

## 5.2.5 获取桶 ACL

### 功能介绍

用户执行获取桶ACL的操作，返回信息包含指定桶的权限控制列表信息。用户必须拥有对指定桶READ_ACP的权限或FULL_CONTROL权限，才能执行获取桶ACL的操作。

### 请求消息样式

```
GET /?acl HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求消息中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Owner>
      <ID>id</ID>
   </Owner>
   <AccessControlList>
      <Grant>
         <Grantee>
            <ID>id</ID>
         </Grantee>
         <Permission>permission</Permission>
         <Delivered>false</Delivered>
      </Grant>
   </AccessControlList>
</AccessControlPolicy>
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应中以消息元素的形式返回桶的ACL信息，元素的具体意义如表5-13所示。

**表 5-13** 响应消息元素

| 元素 | 元素说明 |
| --- | --- |
| Owner | 桶的所有者信息。<br>类型：XML |
| ID | 用户所属租户的租户Id。<br>类型：String |
| AccessControlList | 访问控制列表，记录了对该桶有访问权限的用户列表和这些用户具有的权限。<br>类型：XML |
| Grant | 用于标记用户及用户的权限。<br>类型：XML |
| Grantee | 记录用户信息。<br>类型：XML |
| Canned | 向所有人授予权限。<br>类型：String，其值只能是Everyone。 |
| Delivered | 桶的ACL是否向桶内对象传递。<br>类型：Boolean |
| Permission | 指定的用户对该桶所具有的操作权限。<br>类型：String |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?acl HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:39:28 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:X7HtzGsIEkzJbd8vo1DRu30vVrs=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436B69D82F14E93528658
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSjTh8661+HF5y8uAnTOBIpNO133hji+
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:39:28 GMT
```

```
Content-Length: 784

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://obs.example.com/doc/2015-06-30/">
  <Owner>
    <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <ID>783fc6652cf246c096ea836694f71855</ID>
      </Grantee>
      <Permission>READ</Permission>
      <Delivered>false</Delivered>
    </Grant>
    <Grant>
      <Grantee>
        <Canned>Everyone</Canned>
      </Grantee>
      <Permission>READ_ACP</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

# 5.2.6 设置桶日志管理配置

## 功能介绍

创建桶时，默认是不生成桶的日志的，如果需要生成桶的日志，该桶需要打开日志配置管理的开关。桶日志功能开启后，桶的每次操作将会产生一条日志，并将多条日志打包成一个日志文件。日志文件存放位置需要在开启桶日志功能时指定，可以存放到开启日志功能的桶中，也可以存放到其他你有权限的桶中，但需要和开启日志功能的桶在同一个region中。

由于日志文件是OBS产生，并且由OBS上传到存放日志的桶中，因此OBS需要获得委托授权，用于上传生成的日志文件，所以在配置桶日志管理前，需要先到统一身份认证服务生成一个对OBS服务的委托，并将委托名作为参数配置到桶上，并且在xml文件中<LoggingEnabled>标签下配置相应的日志管理功能。在为委托配置权限时只需设置目标桶的上传对象权限。

## 委托权限示例

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:object:PutObject"
            ],
            "Resource": [
                "OBS:*:*:object:mybucketlogs/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

关闭桶日志功能的方法是上传一个带有空的BucketLoggingStatus标签的logging文件。

## 请求消息样式

```
PUT /?logging HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: signatureValue
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
  <Agency>agency-name</Agency>
  <LoggingEnabled>
    <TargetBucket>mybucketlogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee>
          <ID>domainID</ID>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体请参见**表3-3**。

## 请求消息元素

**表 5-14** 请求消息元素表

| 名字 | 描述 | 是否必选 |
| --- | --- | --- |
| BucketLoggingStatus | 日志状态信息的容器。<br>类型：Container | 是 |
| Agency | 目标桶Owner通过统一身份认证服务创建的对OBS服务的委托的名称。<br>类型：String | 设置logging时必选。关闭logging时勿选。 |
| LoggingEnabled | 该元素起到对日志配置管理的使能作用（呈现此元素则打开日志配置，否则关闭配置）。在此元素下，可加入具体的日志配置信息。<br>类型：Container | 设置logging时必选。关闭logging时勿选。 |

| 名字 | 描述 | 是否必选 |
| --- | --- | --- |
| Grant | 是被授权者及其权限的容器。用于描述谁有什么权限来访问产生的日志文件。<br>类型：Container | 否 |
| Grantee | 作为被授权logging权限用户的容器。<br>类型：Container | 否 |
| ID | 被授权者的租户ID，全局唯一标识。<br>类型：String | 否 |
| Permission | 产生的日志文件对被授权者的具体权限。<br>类型：String<br>权限有效值：FULL_CONTROL \| READ \| WRITE | 否 |
| TargetBucket | 在生成日志时，配置日志桶的所有者可以指定一个桶用于存放产生的日志文件。需要保证配置日志文件的桶owner对存放日志文件的桶有FULL_CONTROL权限。支持多个桶生成的日志放在同一个目标桶中，如果这样做，就需要指定不同的TargetPrefix以达到为来自不同源桶的日志分类的目的。<br>类型：String | 设置logging时必选。关闭logging时勿选。 |
| TargetPrefix | 通过该元素指定一个前缀，所有生成的日志对象的对象名都以此元素的内容为前缀。<br>类型：String | 设置logging时必选。关闭logging时勿选。 |
| TargetGrants | 授权信息的容器。<br>类型：Container | 否 |

## 存储访问日志的 object 命名规则

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

- <TargetPrefix>为用户指定的目标前缀。
- YYYY-mm-DD-HH-MM-SS为日志生成的日期与时间，各字段依次表示年、月、日、时、分、秒。
- <UniqueString>为OBS自动生成的字符串。

一个实际用于存储OBS访问日志的object名称实例如下：

```
bucket-log2015-06-29-12-22-07-N7MXLAF1BDG7MPDV
```

- "bucket-log"为用户指定的目标前缀。
- "2015-06-29-12-22-07"为日志生成的日期与时间。
- "N7MXLAF1BDG7MPDV"为OBS自动生成的字符串。

## 桶访问日志格式

以下所示为在目标桶生成的桶访问日志文件记录：

```
787f2f92b20943998a4fe2ab75eb09b8 bucket [13/Aug/2015:01:43:42 +0000] xx.xx.xx.xx
787f2f92b20943998a4fe2ab75eb09b8 281599BACAD9376ECE141B842B94535B
REST.GET.BUCKET.LOCATION - "GET /bucket?location HTTP/1.1" 200 - 211 - 6 6 "-"  "HttpClient" - -
```

每个桶访问日志都包含如下信息：

**表 5-15** Bucket Logging 格式

| 名称 | 示例 | 含义 |
| --- | --- | --- |
| BucketOwner | 787f2f92b20943998a4fe2ab75eb09b8 | 桶的ownerId |
| Bucket | bucket | 桶名 |
| Time | [13/Aug/2015:14:43:42 +0000] | 请求时间戳。格式为：[dd/MMM/yyyy:HH:mm:ss Z]，即 [日/月/年:小时:分钟:秒 时区] |
| Remote IP | xx.xx.xx.xx | 请求IP |
| Requester | 787f2f92b20943998a4fe2ab75eb09b8 | 请求者ID<br>• 当使用账号或IAM用户发起请求时，此ID为请求者所属账号的账号ID。<br>• 当使用匿名用户发起请求时，取值为Anonymous。 |
| RequestID | 281599BACAD9376ECE141B842B94535B | 请求ID |
| Operation | REST.GET.BUCKET.LOCATION | 操作名称 |
| Key | - | 对象名 |

| 名称 | 示例 | 含义 |
| --- | --- | --- |
| Request-URI | GET /bucket?location HTTP/1.1 | 请求URI |
| HTTPStatus | 200 | 返回码 |
| ErrorCode | - | 错误码 |
| BytesSent | 211 | HTTP响应的字节大小 |
| ObjectSize | - | 对象大小 |
| TotalTime | 6 | 服务端处理时间<br>单位：ms |
| Turn-AroundTime | 6 | 总请求时间<br>单位：ms |
| Referer | - | 请求的referrer头域 |
| User-Agent | HttpClient | 请求的user-agent头域 |
| VersionID | - | 请求中带的versionId |
| STSLogUrn | - | 联邦认证及委托授权信息 |
| StorageClass | STANDARD_IA | 当前的对象存储类型 |
| TargetStorageClass | GLACIER | 通过转换后的对象存储类型 |
| DentryName | 12456%2Ffile.txt | • 对于并行文件系统，是文件/目录的内部标识，由父目录inode编号与文件/目录名称组成，此字段最终呈现为经过URL编码后的格式。<br>• 对于对象桶，该字段为"-"。 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?logging HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:40:06 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:mCOjER/L4ZZUY9qr6AOnkEiwvVk=
Content-Length: 528

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
  <Agency>agencyGrantPutLogging</Agency>
  <LoggingEnabled>
    <TargetBucket>log-bucket</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee>
          <ID>783fc6652cf246c096ea836694f71855</ID>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF26000001643663CE53B6AF31C619FD
x-obs-id-2: 32AAAQAAEAABSAAkpAIAABAAAQAAEAABCT9CjuOx8cETSRbqkm35s1dL/tLhRNdZ
Date: WED, 01 Jul 2015 02:40:06 GMT
Content-Length: 0
```

# 5.2.7 获取桶日志管理配置

## 功能介绍

该接口的目的是查询当前桶的日志管理配置情况。其实现是通过使用http的get方法再加入logging子资源来返回当前桶的日志配置情况。

要使用该接口，使用者必须是桶的所有者或者是被桶策略授权GetBucketLogging权限的用户。

## 请求消息样式

```
GET /?logging HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://obs.region.example.com/doc/2015-06-30/">
<Agency>agency-name</Agency>
<LoggingEnabled>
   <TargetBucket>bucketName</TargetBucket>
   <TargetPrefix>prefix</TargetPrefix>
      <TargetGrants>
         <Grant>
            <Grantee>
               <ID>id</ID>
            </Grantee>
            <Permission>permission</Permission>
         </Grant>
      </TargetGrants>
   </LoggingEnabled>
</BucketLoggingStatus>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应中以消息元素的形式返回桶的日志信息，元素的具体意义如**表5-16**所示。

**表 5-16** 响应消息元素

| 名字 | 描述 |
| --- | --- |
| BucketLoggingStatus | logging状态信息的容器。<br>类型：Container |
| Agency | 产生logging日志桶Owner创建委托OBS上传logging日志的委托名。<br>类型：String |
| LoggingEnabled | 用于logging信息的容器。并且该元素起到对logging配置管理的使能作用（呈现此元素则打开logging配置，否则关闭）。<br>类型：Container |

| 名字 | 描述 |
| --- | --- |
| Grant | 是被授权者及其权限的容器。<br>类型：Container |
| Grantee | 作为被授权logging权限用户的容器。<br>类型：Container |
| ID | 被授权用户的Domain Id，全局唯一标识。<br>类型：String |
| Permission | 对于一个桶的logging权限来说，owner在创桶时将自动获得对源桶的FULL_CONTROL权限。不同的权限决定了对不同日志的访问限制。<br>类型：String<br>权限有效值：FULL_CONTROL \| READ \| WRITE |
| TargetBucket | 在生成日志时，源桶的owner可以指定一个目标桶，将生成的所有日志放到该桶中。在OBS系统中，支持多个源桶生成的日志放在同一个目标桶中，如果这样做，就需要指定不同的TargetPrefix以达到为来自不同源桶的日志分类的目的。<br>类型：String |
| TargetPrefix | 通过该元素可以指定一个前缀给一类日志生成的对象。<br>类型：String |
| TargetGrants | 授权信息的容器。<br>类型：Container |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?logging HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 02:42:46 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:hUk+jTnR07hcKwJh4ousF2E1U3E=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436B8EEE7FBA2AA3335E3
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCShuQJoWFpS77C8bOv1mqURv0UY+0ejx
Content-Type: application/xml
Date: WED, 01 Jul 2015 02:42:46 GMT
Content-Length: 429

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<BucketLoggingStatus xmlns="http://obs.example.com/doc/2015-06-30/">
  <Agency>agency-name</Agency>
  <LoggingEnabled>
    <TargetBucket>log-bucket</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee>
          <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

## 5.2.8 设置桶的生命周期配置

### 功能介绍

OBS系统支持指定规则来实现定时删除桶中对象，这就是生命周期配置。典型的应用场景如：

- 周期性上传的日志文件，可能只需要保留一个星期或一个月，到期后要删除它们。
- 某些文档在一段时间内经常访问，但是超过一定时间后就可能不会再访问了。这种文档您可能会先选择归档，然后在一定时间后删除。

本接口实现为桶创建或更新生命周期配置信息。

**说明**

- 对象生命周期到期以后，对象将会永久删除，无法恢复。

要正确执行此操作，需要确保执行者有PutLifecycleConfiguration权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

生命周期配置实现了定时删除对象的功能，所以如果想要阻止用户删除，以下几项操作的权限都应该被禁止：

- DeleteObject
- DeleteObjectVersion
- PutLifecycleConfiguration

如果想要阻止用户管理桶的生命周期配置，应该禁止PutLifecycleConfiguration权限。

### 请求消息样式

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Length: length
Date: date
Authorization: authorization
Content-MD5: MD5
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
   <Rule>
      <ID>id</ID>
      <Prefix>prefix</Prefix>
      <Status>status</Status>
      <Expiration>
```

```
        <Days>days</Days>
      </Expiration>
      <NoncurrentVersionExpiration>
        <NoncurrentDays>days</NoncurrentDays>
      </NoncurrentVersionExpiration>
   </Rule>
</LifecycleConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用的消息头如下表5-17所示。

**表 5-17** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。<br>类型：String<br>示例：n58IG6hfM7vqI4K0vnWpog== | 是 |

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的生命周期配置信息。配置信息以XML格式上传，具体的配置元素如表5-18描述。

- 如果桶的多版本是Enabled或者Suspended，那么可以设置NoncurrentVersionExpiration来控制对象的历史版本的生命周期。一个历史版本的生命周期，取决于它成为历史版本的时刻（即被新版本覆盖的那个时刻）和NoncurrentDays。例如NoncurrentDays配置为1的话，表示当一个版本成为历史版本之后，再过1天才能删除。对象A的版本V1创建于1号，5号的时候又上传新的版本V2，此时V1成为历史版本，那么再过1天，7号的0点，V1就过期了。（备注：对象过期后被删除的时间可能会有一定的延迟，一般不超过48小时。）
- 如果桶的多版本是Enabled或者Suspended，且最新版本对象满足Expiration规则时的处理：
  - 桶当前的多版本状态为Enabled：

    如果对象的最新版本不是deletemarker，则该对象会产生一个新的deletemarker；

    如果最新版本是deletemarker，且该对象只有这一个版本，则这个版本会被删除；

    如果最新版本是deletemarker，且对象还有其他版本，则该对象的所有版本维持不变，没有新增和删除，也不会被修改（即无任何变化）。
  - 桶当前的多版本状态为Suspended：

    如果对象的最新版本不是deletemarker，且版本不是null版本，则会产生一个新的null版本的deletemarker；

如果对象的最新版本不是deletemarker，且版本是null版本，则这个null版本会被新产生的null版本的deletemarker覆盖；

如果最新版本是deletemarker，且该对象只有这一个版本，则这个版本会被删除；

如果最新版本是deletemarker，且对象还有其他版本，则该对象的所有版本维持不变，没有新增和删除，也不会被修改（即无任何变化）。

**表 5-18** 生命周期配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Date | 指定OBS对该日期之前的对象执行生命周期规则。日期格式必须为ISO8601的格式，并且为UTC的零点。 例如：2018-01-01T00:00:00.000Z，表示将最后修改时间早于2018-01-01T00:00:00.000Z的对象删除，等于或晚于这个时间的对象不会被删除。<br>类型：String<br>父节点：Expiration | 如果没有Days元素，则必选 |
| Days | 指定生命周期规则在对象最后更新过后多少天生效（仅针对对象的最新版本）。<br>类型：Integer<br>父节点：Expiration | 如果没有Date元素，则必选 |
| Expiration | 生命周期配置中表示过期时间的Container（仅针对对象的最新版本）。<br>类型：XML<br>子节点：Date或Days<br>父节点：Rule | 是 |
| ID | 一条Rule的标识，由不超过255个字符的字符串组成。<br>类型：String<br>父节点：Rule | 否 |
| LifecycleConfiguration | 生命周期配置Rule的Container。可以配置多条Rule，但需保证整个配置消息体总大小不超过20KB。<br>类型：XML<br>子节点：Rule<br>父节点：无 | 是 |
| NoncurrentDays | 表示对象在成为历史版本之后第几天时规则生效（仅针对历史版本）。<br>类型：Integer<br>父节点：NoncurrentVersionExpiration | 如果有NoncurrentVersionExpiration元素，则必选 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| NoncurrentVersionExpiration | 生命周期配置中表示历史版本过期时间的Container。您可以将该动作设置在已启用多版本（或暂停）的桶，来让系统删除对象的满足特定生命周期的历史版本（仅针对历史版本）。<br>类型：XML<br>子节点：NoncurrentDays<br>父节点：Rule | 否 |
| Prefix | 对象名前缀，用以标识哪些对象可以匹配到当前这条Rule。<br>类型：String<br>父节点：Rule<br>约束：<br>1. 当按前缀配置时，如果指定的前缀名与某条已配置的生命周期规则指定的前缀名存在包含关系，OBS会将两条规则视为同一条，而禁止您配置本条规则。例如，系统中已存在指定前缀名为“abc”的规则，则不允许再配置指定前缀以“abc”字段开头的规则。<br>2. 如果已存在按前缀配置的生命周期规则，则不允许再新增配置到整个桶的规则。 | 是 |
| Rule | 具体某一条生命周期配置的Container。<br>类型：Container<br>父节点：LifecycleConfiguration | 是 |
| Status | 标识当前这条Rule是否启用。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?lifecycle HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:05:34 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:DpSAlmLX/BTdjxU5HOEwflhM0WI=
Content-MD5: ujCZn5p3fmczNiQQxdsGaQ==
Content-Length: 919

<?xml version="1.0" encoding="utf-8"?>
<LifecycleConfiguration>
  <Rule>
    <ID>delete-2-days</ID>
    <Prefix>test/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>70</Days>
    </Expiration>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>70</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF26000001643670AC06E7B9A7767921
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSvK6z8HV6nrJh49gsB5vqzpgtohkiFm
Date: WED, 01 Jul 2015 03:05:34 GMT
Content-Length: 0
```

# 5.2.9 获取桶的生命周期配置

## 功能介绍

获取该桶设置的生命周期配置信息。

要正确执行此操作，需要确保执行者有GetLifecycleConfiguration执行权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
GET /?lifecycle HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LifecycleConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Rule>
      <ID>id</ID>
      <Prefix>prefix</Prefix>
      <Status>status</Status>
      <Expiration>
         <Date>date</Date>
      </Expiration>
      <NoncurrentVersionExpiration>
         <NoncurrentDays>days</NoncurrentDays>
      </NoncurrentVersionExpiration>
   </Rule>
</LifecycleConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下**表5-19**描述。

**表 5-19** 生命周期配置元素

| 名称 | 描述 |
| --- | --- |
| Date | 指定OBS对该日期之前的对象执行生命周期规则。日期格式必须为ISO8601的格式，并且为UTC的零点。 例如：2018-01-01T00:00:00.000Z，表示将最后修改时间早于2018-01-01T00:00:00.000Z的对象删除，等于或晚于这个时间的对象不会被删除。<br>类型：String<br>父节点：Expiration |
| Days | 指定在对象最后修改时间的多少天后执行生命周期规则（仅针对对象的最新版本）。<br>类型：Integer<br>父节点：Expiration |

| 名称 | 描述 |
| --- | --- |
| Expiration | 生命周期配置中表示过期时间的Container。<br>类型：XML<br>子节点：Date或Days<br>父节点：Rule |
| ID | 一条Rule的标识，由不超过255个字符的字符串组成。<br>类型：String<br>父节点：Rule |
| LifecycleConfiguration | 生命周期配置Rule的Container。可以配置多条Rule，但需保证整个配置消息体总大小不超过20KB。<br>类型：XML<br>子节点：Rule<br>父节点：无 |
| NoncurrentDays | 表示对象在成为历史版本之后第几天时规则生效。<br>类型：Integer<br>父节点：NoncurrentVersionExpiration |
| NoncurrentVersionExpiration | 生命周期配置中表示历史版本过期时间的Container。您可以将该动作设置在已启用多版本（或暂停）的桶，来让系统删除对象的满足特定生命周期的历史版本。<br>类型：XML<br>子节点：NoncurrentDays<br>父节点：Rule |
| Prefix | 对象名前缀，用以标识哪些对象可以匹配到当前这条Rule。<br>类型：String<br>父节点：Rule |
| Rule | 具体某一条生命周期配置的Container。<br>类型：Container<br>父节点：LifecycleConfiguration |
| Status | 标识当前这条Rule是否启用。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled |

## 错误响应消息

此请求可能的特殊错误如下表5-20描述。

**表 5-20** 特殊错误

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| NoSuchLifecycleConfiguration | 桶的生命周期配置不存在 | 404 Not Found |

其余错误已经包含在表6-2中。

## 请求示例

```
GET /?lifecycle HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:06:56 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:/Nof9FCNANfzIXDS0NDp1IfDu8I=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436BA5684FF5A10370EDB
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSEMKZSIeboCA1eAukgYOOAd7oX3ZONn
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:06:56 GMT
Content-Length: 919

<?xml version="1.0" encoding="utf-8"?>
<LifecycleConfiguration>
  <Rule>
    <ID>delete-2-days</ID>
    <Status>Enabled</Status>
    <Expiration>
      <Days>2</Days>
    </Expiration>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>5</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

# 5.2.10 删除桶的生命周期配置

## 功能介绍

删除指定桶的生命周期配置信息。删除后桶中的对象不会过期，OBS不会自动删除桶中对象。

要正确执行此操作，需要确保执行者有PutLifecycleConfiguration权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
DELETE /?lifecycle HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: Authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: text/xml
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
DELETE /?lifecycle HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:12:22 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:5DGAS7SBbMC1YTC4tNXY57Zl2Fo=
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: BF260000016436C2550A1EEA97614A98
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSB7A0KZEBOCutgcfZvaGVthTGOJSuyk
Date: WED, 01 Jul 2015 03:12:22 GMT
```

# 5.2.11 设置桶的多版本状态

## 功能介绍

多版本功能可在用户意外覆盖或删除对象的情况下提供一种恢复手段。用户可以使用多版本功能来保存、检索和还原对象的各个版本，这样用户能够从意外操作或应用程序故障中轻松恢复数据。多版本功能还可用于数据保留和存档。

默认情况下，桶没有设置多版本功能。

本接口设置桶的多版本状态，用来开启或暂停桶的多版本功能。

设置桶的多版本状态为Enabled，开启桶的多版本功能：

- 上传对象时，系统为每一个对象创建一个唯一版本号，上传同名的对象将不再覆盖旧的对象，而是创建新的不同版本号的同名对象
- 可以指定版本号下载对象，不指定版本号默认下载最新对象；
- 删除对象时可以指定版本号删除，不带版本号删除对象仅产生一个带唯一版本号的删除标记，并不删除对象；
- 列出桶内对象列表时默认列出最新对象列表，可以指定列出桶内所有版本对象列表；

设置桶的多版本状态为Suspended，暂停桶的多版本功能：

- 旧的版本数据继续保留 ；
- 上传对象时创建对象的版本号为null，上传同名的对象将覆盖原有同名的版本号为null的对象；
- 可以指定版本号下载对象，不指定版本号默认下载最新对象；
- 删除对象时可以指定版本号删除，不带版本号删除对象将产生一个版本号为null的删除标记，并删除版本号为null的对象；

只有桶的所有者可以设置桶的多版本状态。

## 请求消息样式

```
PUT /?versioning HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
Content-Length: length

<VersioningConfiguration>
   <Status>status</Status>
</VersioningConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的多版本状态，配置信息以XML格式上传。具体的配置元素如**表5-21**描述。

**表 5-21** 桶的多版本状态配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| VersioningConfiguration | 多版本配置的根节点。<br>父节点：无 | 是 |
| Status | 标识桶的多版本状态。<br>类型：String<br>父节点：VersioningConfiguration<br>有效值：Enabled，Suspended | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date

Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?versioning HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:14:18 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:sc2PM13Wlfcoc/YZLK0MwsI2Zpo=
Content-Length: 89

<VersioningConfiguration>
    <Status>Enabled</Status>
</VersioningConfiguration>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF26000001643672B973EEBC5FBBF909
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSH6rPRHjQCa62fcNpCCPs7+1Aq/hKzE
Date: Date: WED, 01 Jul 2015 03:14:18 GMT
Content-Length: 0
```

# 5.2.12 获取桶的多版本状态

## 功能介绍

桶的所有者可以获取指定桶的多版本状态。

如果从未设置桶的多版本状态，则此操作不会返回桶的多版本状态。

## 请求消息样式

```
GET /?versioning HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: type
Content-Length: length

<VersioningConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Status>status</Status>
</VersioningConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应中以消息元素的形式返回桶的多版本状态，元素的具体意义如**表5-22**所示。

**表 5-22** 响应消息元素

| 名字 | 描述 |
| --- | --- |
| VersioningConfiguration | 多版本状态信息的元素。<br>类型：Container |

| 名字 | 描述 |
| --- | --- |
| Status | 标识桶的多版本状态。<br>类型：String<br>有效值：Enabled，Suspended |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?versioning HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:15:20 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:4N5qQIoluLO9xMY0m+8lIn/UWXM=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436BBA4930622B4FC9F17
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSQIrNJ5/Ag6EPN8DAwWlPWgBc/xfBnx
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:15:20 GMT
Content-Length: 180

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VersioningConfiguration xmlns="http://obs.example.com/doc/2015-06-30/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

# 5.2.13 设置桶的消息通知配置

## 功能介绍

OBS消息通知功能能够帮助您对桶的重要的操作及时通知到您，确保您安全、及时知道发生在桶上的关键事件。

默认情况下，您的桶没有配置事件通知。这个时候桶的通知配置将是一个空NotificationConfiguration。对已配置有事件通知的桶，可以通过添加空NotificationConfiguration元素禁用消息通知功能。

```
<NotificationConfiguration>
</NotificationConfiguration>
```

当 OBS 接收到配置消息通知的请求后，会验证指定的消息通知服务（SMN）主题是否存在及主题策略是否授权给了对象存储服务，验证通过后会向该主题订阅者发送一个测试消息通知。

为了能成功执行此配置操作，需要确保执行者拥有PutBucketNotification权限。默认情况下只有桶的所有者拥有该权限，但可以通过设置桶策略授权给其他用户。

## 请求消息样式

```
PUT /?notification HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: authorization string

<NotificationConfiguration>
   <TopicConfiguration>
      <Id>ConfigurationId</Id>
      <Filter>
         <Object>
            <FilterRule>
               <Name>prefix</Name>
               <Value>prefix-value</Value>
            </FilterRule>
            <FilterRule>
               <Name>suffix</Name>
               <Value>suffix-value</Value>
            </FilterRule>
         </Object>
      </Filter>
      <Topic>TopicARN</Topic>
      <Event>event-type</Event>
      <Event>event-type</Event>
      ...
   </TopicConfiguration>
   ...
</NotificationConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的通知，通知的配置信息以XML格式上传。具体的配置元素如**表1**描述。

**表 5-23** 桶的通知服务配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| NotificationConfiguration | 配置桶事件通知的根元素，如果子元素为空，说明消息通知功能处于关闭状态。<br>类型：Container<br>父元素：无<br>子元素：0个或多个TopicConfiguration， | 是 |
| TopicConfiguration | 配置事件通知主题的元素。<br>类型：Container<br>父元素：NotificationConfiguration<br>子元素：Id，Filter，Topic，一个或多个Event | 否 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Topic | 事件通知主题的URN，当OBS检测到桶中发生特定的事件后，将会发布通知消息至该主题，可以在消息通知服务主题部分找到具体值。<br>类型：String<br>父元素：TopicConfiguration<br>模板：<br><Topic>urn:smn:region:project_id:smn_topic</Topic><br>示例：<br><Topic>urn:smn:*exampleRegion*:d745b885f14941369b2d2138e7a65bef:obs_test</Topic> | 如果是父元素TopicConfiguration添加后，本元素是必选项 |
| Id | 每项事件通知配置的唯一标识，如果是用户未指定ID，系统将自动分配一个ID。<br>类型：String<br>父元素：TopicConfiguration， | 否 |
| Filter | Object的元素，用以保存过滤对象名的一组规则。<br>类型：Container<br>父元素：TopicConfiguration，<br>子元素: Object | 否 |
| Object | 定义过滤规则的元素，该规则用以匹配对象名前缀和后缀。<br>类型：Container<br>父元素：Filter<br>子元素：一个或者多个FilterRule | 否 |
| FilterRule | 定义过滤规则键值对的元素。<br>类型：Container<br>父元素：Object<br>子元素：Name，Value | 否 |
| Name | 指定规则按对象名前缀或后缀进行过滤。<br>类型：String<br>父元素：FilterRule<br>合法值：prefix或者suffix | 否 |
| Value | 指定的对象名关键字，根据Name元素定义的前缀或后缀，输入需要过滤的对象的关键字信息，字符越长匹配精度越高，最大可支持1024个字符。<br>类型：String<br>父元素：FilterRule | 否 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Event | 需要发布通知消息的事件类型。<br>**说明**<br>在一个TopicConfiguration，配置项中可以添加多个事件类型。<br>类型：String<br>合法值：<br>上传对象操作可以取以下值：<br>• ObjectCreated:Put<br>• ObjectCreated:Post<br>• ObjectCreated:Copy<br>• ObjectCreated:CompleteMultipartUpload<br>或者使用通配符支持所有上传操作<br>• ObjectCreated:*<br>删除对象操作可以取以下值：<br>• ObjectRemoved:Delete<br>• ObjectRemoved:DeleteMarkerCreated<br>或者使用通配符支持所有删除操作<br>• ObjectRemoved:*<br>父元素：TopicConfiguration， | 如果是父元素TopicConfiguration，添加后，本元素是必选项 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
Content-Type: type
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

当用户执行调用本接口时，系统将会检查NotificationConfiguration元素的有效性，以及配置是否有效。中列出本接口的一些常见错误，以及可能原因。

**表 5-24** 配置桶的通知的错误码列表

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| InvalidArgument | 该错误可能是由于下列原因导致。<br>● 指定了不支持的event。<br>● 指定的URN不存在或者填写错误，请确认URN合法。<br>● 指定的URN中的区域与桶所在的区域不一致，请确保桶所在的区域与URN中的区域一致。<br>● 指定的过滤规则之间存在互相包含或者重叠。 | 400 Bad Request |
| AccessDenied | 执行者不是桶的所有者，或者还未被授权PutBucketNotification权限。 | 403 Forbidden |

## 请求示例

```
PUT /?notification HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:15:45 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:uRTt8YTkAqJCUfWfYkveEcIGAC0=
Content-Length: 538

<NotificationConfiguration>
  <TopicConfiguration>
    <Id>ConfigurationId</Id>
    <Filter>
      <Object>
        <FilterRule>
          <Name>prefix</Name>
          <Value>object</Value>
        </FilterRule>
        <FilterRule>
          <Name>suffix</Name>
          <Value>txt</Value>
        </FilterRule>
      </Object>
    </Filter>
    <Topic>urn:smn:region:4b29a3cb5bd64581bda5714566814bb7:tet555</Topic>
    <Event>ObjectCreated:Put</Event>
  </TopicConfiguration>
</NotificationConfiguration>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 9046000001643C8E80C19FAC4D8068E3
x-obs-id-2: 32AAAQAAEAABSAAkgAIAABAAAQAAEAABCTFAxJPTib3GkcQ7nVVs4C8Z6NNcfVDu
Date: WED, 01 Jul 2015 03:15:46 GMT
Content-Length: 0
```

# 5.2.14 获取桶的消息通知配置

## 功能介绍

获取指定桶的消息通知配置信息。

为了能成功执行此配置操作，需要确保执行者拥有GetBucketNotification权限。默认情况下只有桶的所有者拥有该权限，但可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
GET /?notification HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: type
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8"?>
<NotificationConfiguration xmlns="http://obs.example.com/doc/2015-06-30/">
   <TopicConfiguration>
      <Id>ConfigurationId</Id>
      <Filter>
         <Object>
            <FilterRule>
               <Name>prefix</Name>
               <Value>prefix-value</Value>
            </FilterRule>
            <FilterRule>
               <Name>suffix</Name>
               <Value>suffix-value</Value>
            </FilterRule>
         </Object>
      </Filter>
      <Topic>TopicARN</Topic>
      <Event>event-type</Event>
      <Event>event-type</Event>
      ...
   </TopicConfiguration>
</NotificationConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下表5-25描述。

**表 5-25** Notification 配置元素

| 名称 | 描述 |
| --- | --- |
| NotificationConfiguration | 配置桶事件通知的元素，如果此项内容为空，说明通知功能处于关闭状态。<br>类型：Container<br>父节点：无<br>子节点：一个或多个TopicConfiguration， |
| TopicConfiguration | 配置事件通知主题的元素。<br>类型：Container<br>父节点：NotificationConfiguration<br>子节点：Id，Filter，Topic，一个或多个Event |
| Topic | 事件通知主题的URN，当OBS检测到桶中发生特定的事件后，将会发布通知消息至该主题。<br>类型：String<br>父节点：TopicConfiguration |
| Id | 每项事件通知配置的唯一标识，如果是用户未指定ID，系统将自动分配一个ID。<br>类型：String<br>父节点：TopicConfiguration， |
| Filter | Object的元素，用以保存过滤对象名的一组规则。<br>类型：Container<br>父节点：TopicConfiguration，<br>子节点: Object |
| Object | Object的元素，用以保存过滤对象名的一组规则。<br>类型：Container<br>父节点：TopicConfiguration， |
| FilterRule | 定义过滤规则键值对的元素。<br>类型：Container<br>父节点：Object<br>子节点：Name，Value |

| 名称 | 描述 |
| --- | --- |
| Name | 指定规则按对象名前缀或后缀进行过滤。<br>类型：String<br>父节点：FilterRule<br>合法值：prefix或者suffix |
| Value | 指定的对象名关键字，用以按照前缀或后缀过滤对象。<br>类型：String<br>父节点：FilterRule |
| Event | 需要发布通知消息的事件类型。<br>**说明**<br>在一个TopicConfiguration，配置项中可以添加多个事件类型。<br>类型：String<br>合法值：<br>上传对象操作可以取以下值：<br>• ObjectCreated:Put<br>• ObjectCreated:Post<br>• ObjectCreated:Copy<br>• ObjectCreated:CompleteMultipartUpload<br>或者使用通配符支持所有上传操作<br>• ObjectCreated:*<br>删除对象操作可以取以下值：<br>• ObjectRemoved:Delete<br>• ObjectRemoved:DeleteMarkerCreated<br>或者使用通配符支持所有删除操作<br>• ObjectRemoved:*<br>父节点：TopicConfiguration， |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?notification HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:16:32 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:r5+2zwPTKwupMg6lkeTUUqPcHfQ=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 900B000001643FDDD751B37BA87590D8
```

```
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSJRBSladan5ZCVw6ZIY/DAs0zs6z7Hh
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:16:32 GMT
Content-Length: 490

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NotificationConfiguration xmlns="http://obs.example.com/doc/2015-06-30/">
  <TopicConfiguration>
    <Topic>urn:smn:region:4b29a3cb5bd64581bda5714566814bb7:tet522</Topic>
    <Id>ConfigurationId</Id>
    <Filter>
      <Object>
        <FilterRule>
          <Name>prefix</Name>
          <Value>object</Value>
        </FilterRule>
        <FilterRule>
          <Name>suffix</Name>
          <Value>txt</Value>
        </FilterRule>
      </Object>
    </Filter>
    <Event>ObjectCreated:Put</Event>
  </TopicConfiguration>
</NotificationConfiguration>
```

# 5.2.15 设置桶的跨区域复制配置

## 功能介绍

跨区域复制是指跨不同区域中的桶自动、异步地复制对象。通过激活跨区域复制，OBS可将新创建的对象及修改的对象从一个源桶复制到不同区域中的目标桶。

**说明**

如果已经设置过桶的跨集群或跨区域复制配置，再次设置将会覆盖已有的复制策略。

配置跨区域复制需要选择IAM委托，配置方法请参见《对象存储用户指南》的“创建IAM委托”章节。

设置桶的跨区域复制，需要满足以下两个要求：

1. 要求源桶和目标桶多版本状态保持一致，否则不能设置replication。如何设置桶的多版本，请参见**设置桶的多版本状态**。
2. 源桶的拥有者和代理人（OBS）必须要有目标桶的写权限（目标桶需要配置BucketPolicy），同时代理人（OBS）还要有源桶的读权限。这需要通过“BucketPolicy”来实现这个权限委托。

如何设置桶策略，请参见**设置桶策略**。设置桶策略后，代理人（OBS）就有权限可以读取源桶的对象，也有权限将对象复制到目标桶中。

## 请求消息样式

```
PUT /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
x-obs-date: date
Content-MD5: MD5
Authorization: authorization string
Content-Length: contentlength

<ReplicationConfiguration>
   <Agency>testAcy</Agency>
   <Rule>
      <ID>rule1</ID>
```

```
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <Destination>
       <Bucket>targetbucketname</Bucket>
       <DeleteData>Enabled</DeleteData>
    </Destination>
    <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
  </Rule>
</ReplicationConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用的消息头如下所示。

**表 5-26** 设置桶的复制配置请求消息头

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。 | 是 |

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的复制配置，通知的配置信息以XML格式上传。具体的配置元素如下描述。

**表 5-27** 设置桶的复制配置元素

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| ReplicationConfiguration | 复制规则的容器，最多可以达到100条规则，所有的复制配置大小可达到50KB。<br>类型：Container<br>子节点:Rule<br>父节点:无 | 是 |
| Agency | 用户创建的委托名字，最大长度为64字符。<br>类型：String<br>父节点：ReplicationConfiguration | 是 |

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| Rule | 一条特定复制规则信息的容器。<br>复制配置必须至少配置一条规则，最多能达到100条规则。<br>类型：Container<br>父节点:<br>ReplicationConfiguration | 是 |
| ID | 规则的特殊辨识符，最大长度为255字符。<br>类型：String<br>父节点:Rule | 否 |
| Status | 如果Status为Disabled，这条规则会被忽略。<br>类型：String<br>父节点：Rule<br>有效值：Enabled,Disabled | 是 |
| Prefix | 对象键值名的前缀，适配于一个或者多个对象。如果前缀配置为空，则跨区域复制规则将作用于整个桶。<br>最大前缀长度可达到为1024个字节，不支持重叠的前缀。<br>类型：String<br>父节点:Rule | 是 |
| Destination | 目标桶信息的容器。<br>类型：Container<br>父节点:Rule | 是 |
| Bucket | 存储被规则标识的对象副本的桶名称。<br>如果在复制配置中有多条规则，这些规则必须都要标识同一个桶作为目标桶。<br>类型：String<br>父节点：Destination | 是 |
| DeleteData | 删除同步复制关键字，如果为Enabled，源桶的对象删除操作会复制到目标端。<br>类型：String<br>父节点：Destination<br>有效值：Enabled,Disabled（如果不设置，则默认为Disabled） | 否 |

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| HistoricalObjectReplication | 历史对象复制关键字，如果为Enabled，会复制符合这条规则的历史对象。<br>类型：String<br>父节点：Rule<br>有效值：Enabled, Disabled（如果不设置，则默认为Disabled） | 否 |

## 响应消息样式

```
HTTP/1.1 status_code
Server: OBS
Date:date
Content-Length: contentlength
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

在此请求的响应中不会返回特殊错误。

## 请求示例

```
PUT /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:39:15 +0000
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:CdeqU0Vg9xNdJMZ0PGPgh5EnkO0=
Content-MD5: l/Z8mfSX+VyV8k5EhIQz5Q==
Content-Length: 330

<ReplicationConfiguration>
  <Agency>testAcy</Agency>
  <Rule>
     <ID>Rule-1</ID>
     <Status>Enabled</Status>
     <Prefix></Prefix>
     <Destination>
        <Bucket>dstbucket</Bucket>
        <DeleteData>Enabled</DeleteData>
     </Destination>
     <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
   </Rule>
</ReplicationConfiguration>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: B595000001644178399 32E5A2295674C
```

```
x-obs-id-2: 32AAAQAAEAABKAAQAAEAABAAAQAAEAABCStv51t2NMMx+Ou+ow7IWV4Sxo231fKe
Date: Wed, 27 Jun 2018 13:39:15 GMT
Content-Length: 0
```

# 5.2.16 获取桶的跨区域复制配置

## 功能介绍

获取指定桶的复制配置信息。执行该配置操作前需要确保执行者拥有GetReplicationConfiguration权限。

## 请求消息样式

```
GET /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization:authorization string
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Server:OBS
Content-Length: contentlength

<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://obs.example.com/doc/2006-03-01/">
  <Agency>testAcy</Agency>
  <Rule>
     <ID>rule1</ID>
     <Status>Enabled</Status>
     <Prefix></Prefix>
     <Destination>
        <Bucket>exampletargetbucket</Bucket>

        <DeleteData>Enabled</DeleteData>
     </Destination>
     <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
  </Rule>
</ReplicationConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下描述。

**表 5-28** 桶的复制配置元素

| 名称 | 描述 |
| --- | --- |
| ReplicationConfiguration | 复制规则的容器，最多可以达到100条规则，所有的复制配置大小可达到50KB 。<br>类型：Container<br>子节点：Rule<br>父节点：无 |
| Agency | 用户创建的委托名字，最大长度为64字符。<br>类型：String<br>父节点：ReplicationConfiguration |
| Rule | 一条特定复制规则信息的容器。<br>复制配置必须至少配置一条规则，最多能达到100条规则。<br>类型：Container<br>父节点：ReplicationConfiguration |
| ID | 规则的特殊辨识符，最大长度为255字符。<br>类型：String<br>父节点：Rule |
| Status | 如果Status为Disabled，这条规则会被忽略。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled |
| Prefix | 对象键值名的前缀，适配于一个或者多个对象。如果前缀配置为空，则跨区域复制规则将作用于整个桶。<br>最大前缀长度可达到为1024个字节，不支持重叠的前缀。<br>类型：String<br>父节点：Rule |
| Destination | 目标桶信息的容器。<br>类型：Container<br>父节点：Rule |
| Bucket | 存储被规则标识的对象副本的桶名称。<br>如果在复制配置中有多条规则，这些规则必须都要标识同一个桶作为目标桶。<br>类型：String<br>父节点：Destination |

| 名称 | 描述 |
| --- | --- |
| DeleteData | 删除同步复制关键字，如果为Enabled，源桶的对象删除操作会复制到目标端。<br>类型：String<br>父节点：Destination<br>有效值：Enabled,Disabled（如果不设置，则默认为Disabled） |
| HistoricalObjectReplication | 历史对象复制关键字，如果为Enabled，会复制符合这条规则的历史对象。<br>类型：String<br>父节点：Rule<br>有效值：Enabled,Disabled（如果不设置，则默认为Disabled） |

## 错误响应消息

在此请求的响应中错误响应消息如下描述。

**表 5-29** 桶的错误响应元素

| 错误码 | 描述 | HTTP响应码 | SOAP错误码前缀 |
| --- | --- | --- | --- |
| NoSuchReplicationConfiguration | 跨region复制配置不存在 | 404 not found | Client |

## 请求示例

```
GET /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:42:40 +0000
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jGHviInfRyOkT/EpySpua1hlBuY=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: B59500000164417B57D02F7EF8823152
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSHu6lz4vgk5G3E32OFcIPEZZgdOEYE/
Content-Type: application/xml
Date: Wed, 27 Jun 2018 13:42:39 GMT
Content-Length: 337

<?xml version="1.0" encoding="utf-8"?>
<ReplicationConfiguration xmlns="http://obs.example.com/doc/2006-03-01/">
  <Rule>
    <ID>Rule-1</ID>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>dstbucket</Bucket>
```

```
    <DeleteData>Enabled</DeleteData>
  </Destination>
  <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
 </Rule>
 <Agency>testAcy</Agency>
</ReplicationConfiguration>
```

# 5.2.17 删除桶的跨区域复制配置

## 功能介绍

删除桶的复制配置。执行该配置操作前需要确保执行者拥有DeleteReplicationConfiguration权限。

## 请求消息样式

```
DELETE /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization string
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 204 No Content
Server: OBS
Date: date
Connection: keep-alive
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

在此请求的响应中不会返回特殊错误。

## 请求示例

```
DELETE /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
```

```
Accept: */*
Date: Wed, 27 Jun 2018 13:45:50 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:3ycNYD0CfMf0gOmmXzdGJ58KjHU=
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 900B000001643FE6BBCC9C9F54FA7A7E
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCS8Exs52zCf9duxPLnBircmGa/JOCjec
Date: Wed, 27 Jun 2018 13:45:50 GMT
```

## 5.2.18 设置桶标签

### 功能介绍

OBS使用PUT操作为一个已经存在的桶添加标签。

为桶添加标签后，该桶上所有请求产生的话单里都会带上这些标签，从而可以针对话单报表做分类筛选，进行更详细的成本分析。例如：某个应用程序在运行过程会往桶里上传数据，我们可以用应用名称作为标签，设置到被使用的桶上。在分析话单时，就可以通过应用名的标签来分析此应用的成本。

要正确执行此操作，需要确保执行者有PutBucketTagging权限。缺省情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

### 接口约束

- 每个桶最多能设置10个标签。

### 请求消息样式

```
PUT /?tagging HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization string
Content-MD5: md5
<Tagging>
   <TagSet>
      <Tag>
         <Key>Tag Name</Key>
         <Value>Tag Value</Value>
      </Tag>
   </TagSet>
</Tagging>
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用的消息头如下**表5-30**所示。

**表 5-30** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。<br>类型：String<br>示例：n58IG6hfM7vqI4K0vnWpog== | 是 |

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的标签。标签的配置信息以XML格式上传。具体的配置元素如**表5-31**。

**表 5-31** 桶的标签配置元素

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Tagging | TagSet和Tag的根元素<br>类型：Container<br>父元素：无 | 是 |
| TagSet | Tag的集合元素<br>类型：Container<br>父元素：Tagging | 是 |
| Tag | Tag的信息元素<br>类型：Container<br>父元素：TagSet | 是 |
| Key | **参数解释：**<br>标签的名字。类型：String。父元素：Tag<br>**约束限制：**<br>• 标签的键名（Key）的最大长度为36个字符。<br>• 标签的键名（Key）和键值（Value）不能包含字符“,”、“*”、“\|”、“/”、“<”、“>”、“=”、“\”以及ASCII码0x00--0x1F的控制字符，在发送到服务器之前，必须将键名（Key）和键值（Value）进行UrlEncode编码。<br>**取值范围：**<br>长度大于0小于36的字符串<br>**默认取值：**<br>无 | 是 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Value | **参数解释：**<br>标签的值。类型：String。父元素：Tag<br>**约束限制：**<br>• 标签的键值（Value）的最大长度为43个字符。<br>• 标签的键名（Key）和键值（Value）不能包含字符“,”、“*”、“\|”、“/”、“<”、“>”、“=”、“\”以及ASCII码0x00--0x1F的控制字符，在发送到服务器之前，必须将键名（Key）和键值（Value）进行UrlEncode编码。<br>**取值范围：**<br>长度大于等于0小于43的字符串。<br>**默认取值：**<br>无 | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
x-obs-request-id: request id
x-obs-id-2: id
Content-Length: length
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

除了公共的错误码外，此接口还会返回一些其他的错误码。下表中列出本接口的一些常见错误，以及可能原因。如**表5-32**。

**表 5-32** 配置桶标签错误码列表

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| InvalidTagError | 配置桶标签时，提供了无效的Tag。 | 400 Bad Request |
| MalformedXMLError | 配置桶标签时，提供的xml格式错误 | 400 Bad Request |

## 请求示例

例如要为桶名为examplebucket的桶打上键名（Key）为TagKey(Name1)，键值（Value）为TagValue(Value1)的标签，则发送的请求为：

```
PUT /?tagging HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:22:50 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:Pf1ZyGvVYg2BzOjokZ/BAeR1mEQ=
Content-MD5: MnAEvkfQIGnBpchOE2U6Og==
Content-Length: 182

<Tagging xmlns="http://obs.example.com/doc/2015-06-30/">
  <TagSet>
    <Tag>
      <Key>TagKey%28Name1%29</Key>
      <Value>TagValue%28Value1%29</Value>
    </Tag>
  </TagSet>
</Tagging>
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: BF26000001643FEBA09B1ED46932CD07
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSEZp87iEirC6DggPB5cN49pSvHBWClg
Date: Wed, 27 Jun 2018 13:22:50 GMT
```

# 5.2.19 获取桶标签

## 功能介绍

OBS使用GET操作来获取指定桶的标签。

要正确执行此操作，需要确保执行者有GetBucketTagging权限。缺省情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
GET /?tagging HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization string
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
x-obs-request-id: request id
x-obs-id-2: id
Content-Type: application/xml
Content-Length: length
Date: date
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Tagging xmlns="http://obs.example.com/doc/2015-06-30/">
   <TagSet>
      <Tag>
         <Key>key</Key>
         <Value>value</Value>
      </Tag>
   </TagSet>
</Tagging>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下**表5-33**。

**表 5-33** 桶标签配置元素

| 名称 | 描述 |
| --- | --- |
| Tagging | TagSet和Tag的元素<br>类型：Container<br>父元素：无 |
| TagSet | Tag的集合元素<br>类型：Container<br>父元素：Tagging |
| Tag | Tag信息的元素<br>类型：Container<br>父元素：TagSet |
| Key | Tag的名字<br>类型：String<br>父元素：Tag |
| Value | Tag的值<br>类型：String<br>父元素：Tag |

## 错误响应消息

除了公共的错误码外，此接口还会返回一些其他的错误码。下表中列出本接口的一些常见错误，以及可能原因。如**表5-34**。

**表 5-34** 配置桶标签的错误码列表

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| NoSuchTagSet | 指定的桶没有设置标签 | 404 Not Found |

## 请求示例

```
GET /?tagging HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:25:44 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:H1INcyc5i0XlHqYTfuzkPxLZUPM=
```

## 响应示例

```
HTTP/1.1 200 OK
x-obs-request-id: 0002B7532E0000015BEB35330C5884X1
x-obs-id-2: s12w20LYNQqSb7moq4ibgJwmQRSmVQV+rFBqplOGYkXUpXeS/nOmbkyD+E35K79j
Content-Type: application/xml
Date: Wed, 27 Jun 2018 13:25:44 GMT
Content-Length: 441

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Tagging xmlns="http://obs.example.com/doc/2015-06-30/">
  <TagSet>
    <Tag>
      <Key>TagName1</Key>
      <Value>TageSetVaule1</Value>
    </Tag>
  </TagSet>
</Tagging>
```

# 5.2.20 删除桶标签

## 功能介绍

OBS使用DELETE操作来删除指定桶的标签。

要正确执行此操作，需要确保执行者有DeleteBucketTagging权限。缺省情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
DELETE /?tagging HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization string
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
x-obs-request-id: request id
x-obs-id-2: id
Content-Length: length
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**

## 请求示例

```
DELETE /?tagging HTTP/1.1
User-Agent: curl/7.19.7
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:46:58 GMT
Authorization: authorization string
```

## 响应示例

```
HTTP/1.1 204 No Content
x-obs-request-id: 0002B7532E0000015BEB2C212E53A17L
x-obs-id-2: CqT+86nnOkB+Cv9KZoVgZ28pSgMF+uGQBUC68flvkQeq6CxoCz65wWFMNBpXvea4
Content-Length: 0
Date: Wed, 27 Jun 2018 13:46:58 GMT
```

# 5.2.21 设置桶配额

## 功能介绍

桶空间配额值必须为非负整数，单位为Byte（字节），能设的最大值为$2^{63}$-1。桶的默认配额为0，表示没有限制桶配额。

**说明**

1. 桶配额设置后，如果想取消配额限制，可以把配额设置为0。
2. 由于桶配额的校验依赖于桶存量，而桶存量是后台计算，因此桶配额可能不会及时生效，存在滞后性。可能会出现桶存量超出配额或者删除数据后存量未能及时回落的情况。
3. 桶存量查询接口请参见**获取桶存量信息**。
4. 桶存量超出配额后再上传对象，会返回HTTP状态码403 Forbidden，错误码InsufficientStorageSpace。请扩大配额，或取消配额限制（设置为0），或删除不需要的对象。

## 请求消息样式

```
PUT /?quota HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Content-Length: length
Authorization: authorization

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <StorageQuota>value</StorageQuota>
</Quota>
```

## 请求消息参数

该请求在请求消息中没有带有参数。

## 请求消息头

该请求没有特殊的请求消息头，公共部分参见**表3-3**。

## 请求消息元素

该操作需要附加请求消息元素来指定桶的空间配额，具体见**表5-35**。

**表 5-35** 附加请求消息元素

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| StorageQuota | 指定桶空间配额值单位为字节。<br>类型：Integer | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?quota HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
```

```
Date: WED, 01 Jul 2015 03:24:37 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:k/rbwnYaqYf0Ae6F0M3OJQ0dmI8=
Content-Length: 106

<Quota xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <StorageQuota>10240000</StorageQuota>
</Quota>
```

## 响应示例

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435E09A2BCA388688AA08
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSHbmBecv7ohDSvqaRObpxzgzJ9+l8xT
Date: WED, 01 Jul 2015 03:24:37 GMT
Content-Length: 0
```

# 5.2.22 获取桶配额

## 功能介绍

桶的拥有者可以执行获取桶配额信息的操作。桶的拥有者状态为欠费冻结时不可以查询桶配额信息。桶空间配额值的单位为Byte（字节），0代表不设上限。

## 请求消息样式

```
GET /?quota HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请示消息中不带消息参数。

## 请求消息头

该请求使用公共的请求消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: application/xml
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota xmlns="http://obs.region.example.com/doc/2015-06-30/">
    <StorageQuota>quota</StorageQuota>
</Quota>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该响应以消息元素的形式返回桶的配额信息，元素的具体意义如表5-36所示。

**表 5-36** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| Quota | 桶的配额，包含配额量元素。<br>类型：XML |
| StorageQuota | 桶的配额量。单位字节。<br>类型：String |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?quota HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:27:45 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:8m4bW1gFCNeXQlfu45uO2gpo7l8=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436B55D8DED9AE26C4D18
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSs2Q5vz5AfpAJ/CMNgCfo2hmDowp7M9
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:27:45 GMT
Content-Length: 150

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Quota xmlns="http://obs.example.com/doc/2015-06-30/">
  <StorageQuota>0</StorageQuota>
</Quota>
```

# 5.2.23 获取桶存量信息

## 功能介绍

查询桶对象个数及对象占用空间，对象占用空间大小值为非负整数，单位为Byte（字节）。

**说明**

由于OBS桶存量是后台统计，因此存量会有一定的时延，不能实时更新，因此不建议对存量做实时校验。

## 请求消息样式

```
GET /?storageinfo HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: authorization
```

## 请求消息参数

该请求不使用请求消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用请求消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: type
Content-Length: length
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetBucketStorageInfoResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
<Size>size</Size>
<ObjectNumber>number</ObjectNumber>
</GetBucketStorageInfoResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该响应中将桶存量信息以消息元素的形式返回，元素的具体含义如**表5-37**所示。

**表 5-37** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| GetBucketStorageInfoResult | 保存桶存量请求结果，包含存量大小和对象个数。<br>类型：XML |
| Size | 返回存量大小。<br>类型：Long |
| ObjectNumber | 返回对象个数。<br>类型：Integer |

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
GET /?storageinfo HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:31:18 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:bLcdeJGYWw/eEEjMhPZx2MK5R9U=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016435DD2958BFDCDB86B55E
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSitZctaPYVnat49fVMd1O+OWIP1yrg3
Content-Type: application/xml
WED, 01 Jul 2015 03:31:18 GMT
Content-Length: 206

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetBucketStorageInfoResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Size>25490</Size>
  <ObjectNumber>24</ObjectNumber>
</GetBucketStorageInfoResult>
```

# 5.2.24 设置桶清单

## 功能介绍

OBS使用PUT操作为一个桶配置清单规则，每个桶最多可以配置10条清单规则。

要使用此操作，需确保执行者有PutBucketInventoryConfiguration操作的权限 。桶拥有者默认具有此权限，并且可以将此权限授予其他人。

## 请求消息样式

```
PUT /?inventory&id=configuration-id  HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
Content-Length: length
Expect: 100-continue

<InventoryConfiguration>
   <Id>configuration-id</Id>
   <IsEnabled>true</IsEnabled>
   <Filter>
        <Prefix>inventoryTestPrefix</Prefix>
   </Filter>
   <Destination>
        <Format>CSV</Format>
        <Bucket>destbucket</Bucket>
        <Prefix>dest-prefix</Prefix>
   </Destination>
   <Schedule>
         <Frequency>Daily</Frequency>
   </Schedule>
   <IncludedObjectVersions>All</IncludedObjectVersions>
   <OptionalFields>
         <Field>Size</Field>
         <Field>LastModifiedDate</Field>
         <Field>ETag</Field>
         <Field>StorageClass</Field>
         <Field>IsMultipartUploaded</Field>
```

```
        <Field>ReplicationStatus</Field>
        <Field>EncryptionStatus</Field>
    </OptionalFields>
</InventoryConfiguration>
```

## 请求消息参数

**表 5-38** 请求消息参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| id | 清单配置的id，必须和消息体中的清单配置id一致。<br>类型：String<br>规格：最长64字节<br>默认值：无<br>有效字符："a-z"、"A-Z"、"0-9"、"-"、"_"和"." | 是 |

## 请求消息头

该请求使用公共消息头，具体参见表3-3。

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的清单。清单的配置信息以XML格式上传。具体的配置元素如表5-39。

**表 5-39** 桶的清单配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| InventoryConfiguration | 清单配置。<br>类型：Container<br>父节点：无<br>子节点：Id、IsEnabled、Filter、Destination、Schedule、IncludedObjectVersions以及OptionalFields | 是 |
| Id | 清单配置的id，必须和请求参数中的清单配置id一致。<br>类型：String<br>规格：最长64字节<br>默认值：无<br>有效字符："a-z"、"A-Z"、"0-9"、"-"、"_"和"."<br>父节点：InventoryConfiguration | 是 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| IsEnabled | 规则是否启用，如果设置为true，则生成清单，反之不生成。<br>类型：Boolean<br>有效值：true、false<br>父节点：InventoryConfiguration | 是 |
| Filter | 清单过滤器配置，清单只包含符合过滤器规则的对象（只支持按对象名前缀进行过滤），如果没有配置过滤器，则包含所有对象。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Prefix | 否 |
| Prefix | 前缀过滤条件，清单文件中只生成以此前缀开头的对象列表。<br>类型：String<br>父节点：Filter | 否 |
| Schedule | 清单文件的生成周期。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Frequency | 是 |
| Frequency | 清单文件的生成周期，只支持按天和按周生成清单，第一次配置完桶清单，任务会在一个小时内启动，之后每隔一个周期启动一次。<br>类型：String<br>父节点：Schedule<br>有效值：Daily、Weekly | 是 |
| Destination | 清单的目标配置。<br>类型：Container<br>父节点：InventoryConfiguration | 是 |
| Format | 生成的清单文件的格式，现只支持CSV格式。<br>类型：String<br>父节点：Destination<br>有效值：CSV | 是 |
| Bucket | 存放清单文件的目标桶的桶名。<br>类型：String<br>父节点：Destination | 是 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Prefix | 生成的清单文件对象名会以此前缀开头，如果不配置前缀，则生成的清单文件对象名默认以BucketInventory开头。<br>类型：String<br>父节点：Destination | 否 |
| IncludedObjectVersions | 清单文件中包含对象的多版本配置。<br>• 如果设置为All，清单会包含对象所有的版本，清单中会增加版本相关的字段：VersionId、IsLatest、和DeleteMarker。<br>• 如果设置为Current，则清单文件中只会列出当前版本信息，不会出现版本相关字段。<br>类型：String<br>父节点：InventoryConfiguration<br>有效值：All、Current | 是 |
| OptionalFields | 在此选项中可以添加一些额外的对象元数据字段，生成的清单文件中会包含OptionalFields中配置的字段。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Field | 否 |
| Field | 可选字段类型，OptionalFields可以包含多个Field元素。<br>类型：String<br>父节点：OptionalFields<br>有效值：Size、LastModifiedDate、StorageClass、ETag、IsMultipartUploaded、ReplicationStatus、EncryptionStatus。 | 否 |

## 响应消息样式

```
HTTP/1.1 status_code
x-obs-request-id: request id
x-obs-id-2: id
Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

除了公共的错误码外，此接口还会返回一些其他的错误码。下面列出本接口的一些常见错误，以及可能原因，如表5-40。

**表 5-40** 设置桶清单错误码列表

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| MalformedXML | 清单的XML配置格式错误。 | 400 Bad Request |
| InvalidArgument | 无效参数。 | 400 Bad Request |
| InventoryCountOverLimit | 配置清单数量超过最大限制。 | 400 Bad Request |
| PrefixExistInclusionRelationship | 清单配置中的前缀存在包含关系。 | 400 Bad Request |

## 请求示例

```
PUT /?inventory&id=test_id HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Tue, 08 Jan 2019 08:17:10 +0000
Authorization: OBS UDSIAMSTUBTEST000001:/e2fqSfzLDb+0M36D4Op/s5KKr0=
Content-Length: 600
Expect: 100-continue

<InventoryConfiguration>
  <Id>test_id</Id>
  <IsEnabled>true</IsEnabled>
  <Filter>
       <Prefix>inventoryTestPrefix</Prefix>
  </Filter>
  <Destination>
       <Format>CSV</Format>
       <Bucket>destbucket</Bucket>
       <Prefix>dest-prefix</Prefix>
  </Destination>
  <Schedule>
        <Frequency>Daily</Frequency>
  </Schedule>
  <IncludedObjectVersions>All</IncludedObjectVersions>
  <OptionalFields>
        <Field>Size</Field>
        <Field>LastModifiedDate</Field>
        <Field>ETag</Field>
        <Field>StorageClass</Field>
        <Field>IsMultipartUploaded</Field>
        <Field>ReplicationStatus</Field>
        <Field>EncryptionStatus</Field>
  </OptionalFields>
</InventoryConfiguration>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
```

```
x-obs-request-id: 000001682C8545B0680893425D60AB83
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSIGTuRtBfo7lpHSt0ZknhdDHmllwd/p
Date: Tue, 08 Jan 2019 08:12:38 GMT
Content-Length: 0
```

# 5.2.25 获取桶清单

## 功能介绍

OBS使用GET操作来获取指定桶的某个清单配置。

要正确执行此操作，需要确保执行者有GetBucketInventoryConfiguration权限。桶拥有者默认具有此权限，并且可以将此权限授予其他人。

## 请求消息样式

```
GET /?inventory&id=configuration-id HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
```

## 请求消息参数

**表 5-41** 请求消息参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| id | 需要获取的清单配置的id。<br>类型：String<br>规格：最长64字节<br>默认值：无<br>有效字符："a-z"、"A-Z"、"0-9"、"-"、"_"和"." | 是 |

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
Content-Type: application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InventoryConfiguration  xmlns="http://obs.region.example.com/doc/2015-06-30/">
```

```
<Id>configuration-id</Id>
<IsEnabled>true</IsEnabled>
<Destination>
  <Format>CSV</Format>
  <Bucket>destbucket</Bucket>
  <Prefix>prefix</Prefix>
</Destination>
<Schedule>
  <Frequency>Daily</Frequency>
</Schedule>
<IncludedObjectVersions>Current</IncludedObjectVersions>
<OptionalFields>
  <Field>Size</Field>
  <Field>LastModifiedDate</Field>
  <Field>ETag</Field>
  <Field>StorageClass</Field>
  <Field>IsMultipartUploaded</Field>
  <Field>ReplicationStatus</Field>
  <Field>EncryptionStatus</Field>
</OptionalFields>
</InventoryConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如**表5-42**。

**表 5-42** 桶清单响应消息元素

| 名称 | 描述 |
| --- | --- |
| InventoryConfiguration | 清单配置。<br>类型：Container<br>父节点：无<br>子节点：Id、IsEnabled、Filter、Destination、Schedule、IncludedObjectVersions以及OptionalFields |
| Id | 清单配置的id，必须和请求参数中的清单配置id一致。<br>类型：String<br>规格：最长64字节<br>默认值：无<br>有效字符："a-z"、"A-Z"、"0-9"、"-"、"_"和"."<br>父节点：InventoryConfiguration |
| IsEnabled | 规则是否启用，如果设置为true，则生成清单，反之不生成。<br>类型：Boolean<br>有效值：true、false<br>父节点：InventoryConfiguration |

| 名称 | 描述 |
| --- | --- |
| Filter | 清单过滤器配置，清单只包含符合过滤器规则的对象（只支持按对象名前缀进行过滤），如果没有配置过滤器，则包含所有对象。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Prefix |
| Prefix | 前缀过滤条件，清单文件中只生成以此前缀开头的对象列表。<br>类型：String<br>父节点：Filter |
| Schedule | 清单文件的生成周期。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Frequency |
| Frequency | 清单文件的生成周期，只支持按天和按周生成清单，第一次配置完桶清单，任务会在一个小时内启动，之后每隔一个周期启动一次。<br>类型：String<br>父节点：Schedule<br>有效值：Daily、Weekly |
| Destination | 清单的目标配置。<br>类型：Container<br>父节点：InventoryConfiguration |
| Format | 生成的清单文件的格式，现只支持CSV格式。<br>类型：String<br>父节点：Destination<br>有效值：CSV |
| Bucket | 存放清单文件的目标桶的桶名。<br>类型：String<br>父节点：Destination |
| Prefix | 生成的清单文件对象名会以此前缀开头，如果不配置前缀，则生成的清单文件对象名默认以BucketInventory开头。<br>类型：String<br>父节点：Destination |

| 名称 | 描述 |
| --- | --- |
| IncludedObjectVersions | 清单文件中包含对象的多版本配置。<br>• 如果设置为All，清单会包含对象所有的版本，清单中会增加版本相关的字段：VersionId、IsLatest、和DeleteMarker。<br>• 如果设置为Current，则清单文件中只会列出当前版本信息，不会出现版本相关字段。<br>类型：String<br>父节点：InventoryConfiguration<br>有效值：All、Current |
| OptionalFields | 在此选项中可以添加一些额外的对象元数据字段，生成的清单文件中会包含OptionalFields中配置的字段。<br>类型：Container<br>父节点：InventoryConfiguration<br>子节点：Field |
| Field | 可选字段类型，OptionalFields可以包含多个Field元素。<br>类型：String<br>父节点：OptionalFields<br>有效值：Size、LastModifiedDate、StorageClass、ETag、IsMultipartUploaded、ReplicationStatus、EncryptionStatus。 |

## 错误响应消息

除了公共的错误码外，此接口还会返回一些其他的错误码。下表中列出本接口的一些常见错误，以及可能原因。如**表5-43**。

**表 5-43** 获取桶清单的错误码列表

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| NoSuchInventoryConfiguration | 没有指定Id对应的清单配置。 | 404 Not Found |

## 请求示例

```
GET /?inventory&id=id1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Tue, 08 Jan 2019 09:32:24 +0000
Authorization: OBS UDSIAMSTUBTEST000001:ySWncC9M08jNsyXdJLSMJkpi7XM=
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 000001682CB4C2EE6808A0D8DF9F3D00
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSBjn5O7Jv9CqvUMO0BenehRdil1n8rR
Content-Type: application/xml
Date: Tue, 08 Jan 2019 09:04:30 GMT
Content-Length: 626

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InventoryConfiguration  xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <Id>id1</Id>
  <IsEnabled>true</IsEnabled>
  <Destination>
    <Format>CSV</Format>
    <Bucket>bucket</Bucket>
    <Prefix>prefix</Prefix>
  </Destination>
  <Schedule>
    <Frequency>Daily</Frequency>
  </Schedule>
  <IncludedObjectVersions>Current</IncludedObjectVersions>
  <OptionalFields>
    <Field>Size</Field>
    <Field>LastModifiedDate</Field>
    <Field>ETag</Field>
    <Field>StorageClass</Field>
    <Field>IsMultipartUploaded</Field>
    <Field>ReplicationStatus</Field>
    <Field>EncryptionStatus</Field>
  </OptionalFields>
</InventoryConfiguration>
```

## 5.2.26 列举桶清单

### 功能介绍

OBS使用不带清单id的GET操作来获取指定桶的所有清单配置，获取到的清单配置一次性返回，不分页。

要正确执行此操作，需要确保执行者有GetBucketInventoryConfiguration权限。缺省情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

### 请求消息样式

```
GET /?inventory HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
```

### 请求消息参数

该请求消息中不使用请求消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
Content-Type: application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListInventoryConfiguration  xmlns="http://obs.region.example.com/doc/2015-06-30/">
 <InventoryConfiguration>
  <Id>id</Id>
  <IsEnabled>true</IsEnabled>
  <Destination>
    <Format>CSV</Format>
    <Bucket>bucket</Bucket>
    <Prefix>prefix</Prefix>
  </Destination>
  <Schedule>
    <Frequency>Daily</Frequency>
  </Schedule>
  <IncludedObjectVersions>Current</IncludedObjectVersions>
  <OptionalFields>
    <Field>Size</Field>
    <Field>LastModifiedDate</Field>
    <Field>ETag</Field>
    <Field>StorageClass</Field>
    <Field>IsMultipartUploaded</Field>
    <Field>ReplicationStatus</Field>
    <Field>EncryptionStatus</Field>
  </OptionalFields>
 </InventoryConfiguration>
</ListInventoryConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如**表5-44**。

**表 5-44** 桶的清单配置元素

| 名称 | 描述 |
| --- | --- |
| ListInventoryConfiguration | 桶清单配置列表。<br>类型：Container |
| InventoryConfiguration | 桶清单配置，配置元素见**表5-42**。<br>类型：Container<br>父节点：ListInventoryConfiguration |

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
GET /?inventory HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Tue, 08 Jan 2019 09:32:24 +0000
Authorization: OBS UDSIAMSTUBTEST000001:ySWncC9M08jNsyXdJLSMJkpi7XM=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 000001682CB4C2EE6808A0D8DF9F3D00
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSBjn5O7Jv9CqvUMO0BenehRdil1n8rR
Content-Type: application/xml
Date: Tue, 08 Jan 2019 09:04:30 GMT
Content-Length: 626

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListInventoryConfiguration  xmlns="http://obs.region.example.com/doc/2015-06-30/">
 <InventoryConfiguration>
  <Id>id1</Id>
  <IsEnabled>true</IsEnabled>
  <Destination>
   <Format>CSV</Format>
   <Bucket>bucket</Bucket>
   <Prefix>prefix</Prefix>
  </Destination>
  <Schedule>
   <Frequency>Daily</Frequency>
  </Schedule>
  <IncludedObjectVersions>Current</IncludedObjectVersions>
  <OptionalFields>
   <Field>Size</Field>
   <Field>LastModifiedDate</Field>
   <Field>ETag</Field>
   <Field>StorageClass</Field>
   <Field>IsMultipartUploaded</Field>
   <Field>ReplicationStatus</Field>
   <Field>EncryptionStatus</Field>
  </OptionalFields>
 </InventoryConfiguration>
</ListInventoryConfiguration>
```

# 5.2.27 删除桶清单

## 功能介绍

OBS使用DELETE操作来删除指定桶的清单配置（通过清单id来指定清单配置）。

要正确执行此操作，需要确保执行者有DeleteBucketInventoryConfiguration权限。缺省情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
DELETE /?inventory&id=configuration-id HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
```

```
Accept: */*
Date: date
Authorization: authorization string
```

## 请求消息参数

**表 5-45** 请求消息参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| id | 需要删除的清单配置的id。<br>类型：String<br>规格：最长64字节<br>默认值：无<br>有效字符："a-z"、"A-Z"、"0-9"、"-"、"_"和"." | 是 |

## 请求消息头

该请求使用公共消息头，具体参见表3-3。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
DELETE /test?inventory&id=id1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Tue, 08 Jan 2019 13:18:35 +0000
Authorization: OBS UDSIAMSTUBTEST000001:UT9F2YUgaFu9uFGMmxFj2CBgQHs=
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 000001682D993B666808E265A3F6361D
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSyB46jGSQsu06m1nyIeKxTuJ+H27ooC
Date: Tue, 08 Jan 2019 13:14:03 GMT
```

## 5.2.28 设置桶的自定义域名

### 功能介绍

OBS使用PUT操作为桶设置自定义域名，设置成功之后，用户访问桶的自定义域名就能访问到桶。

必须保证此自定义域名通过DNS能够正确解析到OBS服务。

### 请求消息样式

```
PUT /?customdomain=domainname HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
Content-Length: 0
```

### 请求参数

**表 5-46** 请求消息参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| customdomain | 桶的自定义域名。<br>类型：String，必须满足域名规则。<br>规格：最长256字节。<br>默认值：无。<br>约束：一个桶最多可以设置30个自定义域名，一个自定义域名只能被一个桶使用。 | 是 |

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

此请求消息中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
```

```
Date: date
Content-Length: 0
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带有响应元素。

### 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

### 请求示例

```
PUT /?customdomain=obs.ccc.com HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Mon, 14 Jan 2019 08:31:36 +0000
Authorization: OBS UDSIAMSTUBTEST000094:u2kJF4kENs6KlIDcAZpAKSKPtnc=
Content-Length: 0
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 000001697692CC5380E9D272E6D8F830
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSsfu2GXj9gScHhFnrrTPY2cFOEZuvta
Date: Wed, 13 Mar 2019 10:22:05 GMT
Content-Length: 0
```

## 5.2.29 获取桶的自定义域名

### 功能介绍

OBS使用GET操作来获取桶的自定义域名。

### 请求消息样式

```
GET /?customdomain HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
```

### 请求参数

该请求消息中不使用请求消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

此请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
Content-Type: application/xml
Date: date
Content-Length: 272

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketCustomDomainsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Domains>
    <DomainName>domainname</DomainName>
    <CreateTime>createtime</CreateTime>
  </Domains>
</ListBucketCustomDomainsResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该响应以消息元素的形式返回桶的自定义域名，元素的具体意义如**表1响应消息元素**所示。

**表 5-47** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ListBucketCustomDomainsResult | 自定义域名返回结果容器。<br>类型：Container<br>子节点：Domains<br>父节点：无 |
| Domains | 自定义域名元素。<br>类型：Container<br>子节点：DomainName、CreateTime<br>父节点：ListBucketCustomDomainsResult |
| DomainName | 自定义域名。<br>类型：String<br>子节点：无<br>父节点:：Domains |
| CreateTime | 自定义域名创建时间。<br>类型：String，UTC时间<br>子节点：无<br>父节点:：Domains |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /?customdomain HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Mon, 14 Jan 2019 08:31:45 +0000
Authorization: OBS UDSIAMSTUBTEST000094:veTm8B18MPLFqNyGh2wmQqovZ2U=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 000001697693130C80E9D2D29FA84FC2
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSM80AI9weqGUsIFJScVxSKlG4DmypX9
Content-Type: application/xml
Date: Wed, 13 Mar 2019 10:22:24 GMT
Content-Length: 272

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketCustomDomainsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Domains>
    <DomainName>obs.ccc.com</DomainName>
    <CreateTime>2019-03-13T10:22:05.912Z</CreateTime>
  </Domains>
</ListBucketCustomDomainsResult>
```

# 5.2.30 删除桶的自定义域名

## 功能介绍

OBS使用DELETE操作来删除桶的自定义域名。

## 请求消息样式

```
DELETE /?customdomain=domainname HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: date
Authorization: authorization string
```

## 请求参数

**表 5-48** 请求消息参数

| 参数 | 描述 | 是否必选 |
| --- | --- | --- |
| customdomain | 需要删除的自定义域名。<br>类型：String，必须满足域名规则。<br>规格：最长256字节。<br>默认值：无。 | 是 |

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

此请求消息中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: request id
x-obs-id-2: id
Date: date
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带有响应元素。

### 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

### 请求示例

```
DELETE /?customdomain=obs.ccc.com HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Mon, 14 Jan 2019 08:27:50 +0000
Authorization: OBS UDSIAMSTUBTEST000094:ACgHHA1z+dqZhqS7D2SbU8ugluw=
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 000001697694073F80E9D3D43BB10B8F
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSyjWyXNRPSnFymJW0AI59GKpW0Qm9UJ
Date: Wed, 13 Mar 2019 10:23:26 GMT
```

## 5.2.31 设置桶的跨集群复制配置

### 功能介绍

跨集群复制是指跨不同集群中的桶自动、异步地复制对象。通过激活跨集群复制，OBS可将新创建的对象及修改的对象从一个源桶复制到相同区域、不同集群中的目标桶。

**说明**

如果已经设置过桶的跨集群或跨区域复制配置，再次设置将会覆盖已有的复制策略。

设置桶的跨集群复制，需要满足以下两个要求：

1. 要求源桶和目标桶多版本状态保持一致，否则不能设置replication。如何设置桶的多版本，请参见**设置桶的多版本状态**。
2. 源桶的拥有者和代理人（OBS）必须要有目标桶的写权限（目标桶需要配置BucketPolicy），同时代理人（OBS）还要有源桶的读权限。这需要通过“BucketPolicy”来实现这个权限委托。

   如何设置桶策略，请参见**设置桶策略**。设置桶策略后，代理人（OBS）就有权限可以读取源桶的对象，也有权限将对象复制到目标桶中。

## 请求消息样式

```
PUT /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
x-obs-date: date
Content-MD5: l/Z8mfSX+VyV8k5EhIQz5Q==
Authorization: authorization string
Content-Length: contentlength
<ReplicationConfiguration>
  <Agency>testAcy</Agency>
   <Rule>
      <ID>rule1</ID>
      <Prefix>key-prefix</Prefix>
      <Status>rule-status</Status>
      <Destination>
         <Bucket>targetbucketname</Bucket>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用的消息头如下所示。

**表 5-49** 设置桶的复制配置请求消息头

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。 | 是 |

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的复制配置，通知的配置信息以XML格式上传。具体的配置元素如下描述。

**表 5-50** 设置桶的复制配置元素

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| ReplicationConfiguration | 复制规则的容器，最多可以达到100条规则，所有的复制配置大小可达到50KB。<br>类型：Container<br>子节点：Rule<br>父节点：无 | 是 |
| Agency | 用户创建的委托名字，最大长度为64个字符。<br>类型：String<br>父节点：ReplicationConfiguration | 是 |
| Rule | 一条特定复制规则信息的容器。<br>复制配置必须至少配置一条规则，最多能达到100条规则。<br>类型：Container<br>父节点：ReplicationConfiguration | 是 |
| ID | 规则的特殊辨识符，最大长度为255个字符。<br>类型：String<br>父节点：Rule | 否 |
| Status | 如果Status为Disabled，这条规则会被忽略。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled | 是 |
| Prefix | 对象键值名的前缀，适配于一个或者多个对象。<br>最大前缀长度可达到为1024个字节，不支持重叠的前缀。<br>类型：String<br>父节点：Rule | 是 |
| Destination | 目标桶信息的容器。<br>类型：Container<br>父节点：Rule | 是 |
| Bucket | 存储被规则标识的对象副本的桶名称。<br>如果在复制配置中有多条规则，这些规则必须都要标识同一个桶作为目标桶。<br>类型：String<br>父节点：Destination | 是 |

| 名称 | 描述 | 是否必须 |
| --- | --- | --- |
| DeleteData | 删除同步复制关键字，如果为Enabled，源桶的对象删除操作会复制到目标端。<br>类型：String<br>父节点：Destination<br>有效值：Enabled，Disabled | 否 |
| HistoricalObjectReplication | 历史对象复制关键字，如果为Enabled，会复制符合这条规则的历史对象。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled | 否 |

## 响应消息样式

```
HTTP/1.1 status_code
Server: OBS
Date:date
Content-Length: contentlength
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带有响应元素。

## 错误响应消息

在此请求的响应中不会返回特殊错误。

## 请求示例

```
PUT /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:39:15 +0000
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:CdeqU0Vg9xNdJMZ0PGPgh5EnkO0=
Content-MD5: l/Z8mfSX+VyV8k5EhIQz5Q==
Content-Length: 330

<ReplicationConfiguration>
  <Agency>testAcy</Agency>
  <Rule>
    <ID>Rule-1</ID>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>dstbucket</Bucket>
      <DeleteData>Enabled</DeleteData>
    </Destination>
    <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
```

```
    </Rule>
</ReplicationConfiguration>
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: B595000001644178399 32E5A2295674C
x-obs-id-2: 32AAAQAAEAABKAAQAAEAABAAAQAAEAABCStv51t2NMMx+Ou+ow7IWV4Sxo231fKe
Date: Wed, 27 Jun 2018 13:39:15 GMT
Content-Length: 0
```

## 5.2.32 获取桶的跨集群复制配置

### 功能介绍

获取指定桶的复制配置信息。执行该配置操作前需要确保执行者拥有GetReplicationConfiguration权限。

### 请求消息样式

```
GET /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization:authorization string
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Server:OBS
Content-Length: contentlength
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://obs.example.com/doc/2006-03-01/">
  <Agency>testAcy</Agency>
  <Rule>
     <ID>rule1</ID>
     <Status>Enabled</Status>
     <Prefix></Prefix>
     <Destination>
        <Bucket>exampletargetbucket</Bucket>
        <DeleteData>Enabled</DeleteData>
     </Destination>
     <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
  </Rule>
</ReplicationConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下描述。

**表 5-51** 桶的复制配置元素

| 名称 | 描述 |
| --- | --- |
| ReplicationConfiguration | 复制规则的容器，最多可以达到100条规则，所有的复制配置大小可达到50KB 。<br>类型：Container<br>子节点：Rule<br>父节点：无 |
| Rule | 一条特定复制规则信息的容器。<br>复制配置必须至少配置一条规则，最多能达到100条规则。<br>类型：Container<br>父节点：ReplicationConfiguration |
| ID | 规则的特殊辨识符，最大长度为255个字符。<br>类型：String<br>父节点：Rule |
| Status | 如果Status为Disabled，这条规则会被忽略。<br>类型：String<br>父节点：Rule<br>有效值：Enabled，Disabled |
| Prefix | 对象键值名的前缀，适配于一个或者多个对象。<br>最大前缀长度可达到为1024个字节，不支持重叠的前缀。<br>类型：String<br>父节点：Rule |
| Destination | 目标桶信息的容器。<br>类型：Container<br>父节点：Rule |
| Bucket | 存储被规则标识的对象副本的桶名称。<br>如果在复制配置中有多条规则，这些规则必须都要标识同一个桶作为目标桶。<br>类型：String<br>父节点：Destination |

| 名称 | 描述 |
| --- | --- |
| DeleteData | 删除同步复制关键字，如果为Enabled，源桶的对象删除操作会复制到目标端。<br>类型：String<br>父节点：Destination<br>有效值：Enabled,Disabled |
| HistoricalObjectReplication | 历史对象复制关键字，如果为Enabled，会复制符合这条规则的历史对象。<br>类型：String<br>父节点：Rule<br>有效值：Enabled, Disabled |

## 错误响应消息

在此请求的响应中错误响应消息如下描述。

**表 5-52** 桶的错误响应元素

| 错误码 | 描述 | HTTP响应码 | SOAP错误码前缀 |
| --- | --- | --- | --- |
| NoSuchReplicationConfiguration | 复制配置不存在 | 404 not found | Client |

## 请求示例

```
GET /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:42:40 +0000
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:jGHviInfRyOkT/EpySpua1hlBuY=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: B59500000164417B57D02F7EF8823152
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSHu6lz4vgk5G3E32OFcIPEZZgdOEYE/
Content-Type: application/xml
Date: Wed, 27 Jun 2018 13:42:39 GMT
Content-Length: 337

<?xml version="1.0" encoding="utf-8"?>
<ReplicationConfiguration xmlns="http://obs.example.com/doc/2006-03-01/">
  <Rule>
    <ID>Rule-1</ID>
    <Status>Enabled</Status>
    <Prefix></Prefix>
    <Destination>
      <Bucket>dstbucket</Bucket>
      <DeleteData>Enabled</DeleteData>
    </Destination>
```

```
  <HistoricalObjectReplication>Enabled</HistoricalObjectReplication>
 </Rule>
 <Agency>testAcy</Agency>
</ReplicationConfiguration>
```

## 5.2.33 删除桶的跨集群复制配置

### 功能介绍

删除桶的复制配置。执行该配置操作前需要确保执行者拥有DeleteReplicationConfiguration权限。

### 请求消息样式

```
DELETE /?replication HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization string
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 204 No Content
Server: OBS
Date: date
Connection: keep-alive
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带有响应元素。

### 错误响应消息

在此请求的响应中不会返回特殊错误。

### 请求示例

```
DELETE /?replication HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Wed, 27 Jun 2018 13:45:50 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:3ycNYD0CfMf0gOmmXzdGJ58KjHU=
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 900B000001643FE6BBCC9C9F54FA7A7E
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCS8Exs52zCf9duxPLnBircmGa/JOCjec
Date: Wed, 27 Jun 2018 13:45:50 GMT
```

# 5.3 静态网站托管

## 5.3.1 设置桶的网站配置

### 功能介绍

OBS允许在桶内保存静态的网页资源，如.html网页文件、flash文件、音视频文件等，当客户端通过桶的Website接入点访问这些对象资源时，浏览器可以直接解析出这些支持的网页资源，呈现给最终用户。典型的应用场景有：

- 重定向所有的请求到另外一个站点。
- 设定特定的重定向规则来重定向特定的请求。

本接口实现为桶创建或更新网站配置信息。

要正确执行此操作，需要确保执行者有PutBucketWebsite权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

**说明**

1.尽量避免目标桶名中带有“.”，否则通过HTTPS访问时可能出现客户端校验证书出错。

2.设置桶的网络配置请求消息体的上限是10KB。

### 请求消息样式

```
PUT /?website HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Length: length
Date: date
Authorization: authorization
<WebsiteConfiguration>
   <RedirectAllRequestsTo>
      <HostName>hostName</HostName>
   </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见表3-3。

### 请求消息元素

在此请求中，需要在请求的消息体中配置桶的网站配置信息，配置信息以XML格式上传。

- 如果重定向所有请求，网站配置元素如表5-53描述。

**表 5-53** 重定向所有请求 Website 配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| WebsiteConfiguration | 网站配置的根节点。<br>类型：Container<br>父节点：无 | 是 |
| RedirectAllRequestsTo | 描述所有请求的重定向行为，如果这个节点出现，所有其他的兄弟节点都不能出现。<br>类型：Container<br>父节点：WebsiteConfiguration | 是 |
| HostName | 描述重定向的站点名。<br>类型：String<br>父节点：RedirectAllRequestsTo | 是 |
| Protocol | 描述重定向请求时使用的协议（http，https），默认使用http协议。<br>类型：String<br>父节点：RedirectAllRequestsTo | 否 |

- 如果想要设定重定向规则，网站配置元素如表5-54描述。

**表 5-54** 设定重定向规则 Website 配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| WebsiteConfiguration | Website配置的根节点。<br>类型：Container<br>父节点：无 | 是 |
| IndexDocument | *Suff元素*。<br>类型：Container<br>父节点：WebsiteConfiguration | 是 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Suffix | *Suffix*元素被追加在对文件夹的请求的末尾（例如：*Suffix*配置的是“index.html”，请求的是“samplebucket/images/”，返回的数据将是“samplebucket”桶内名为“images/index.html”的对象的内容）。*Suffix*元素不能为空或者包含“/”字符。<br>类型：String<br>父节点：IndexDocument | 是 |
| ErrorDocument | *Key*元素。<br>类型：Container<br>父节点：WebsiteConfiguration | 否 |
| Key | 当4XX错误出现时使用的对象的名称。这个元素指定了当错误出现时返回的页面。<br>类型：String<br>父节点：ErrorDocument<br>条件：父节点ErrorDocument存在时 | 否 |
| RoutingRules | *Routing元素*。<br>类型：Container<br>父节点：WebsiteConfiguration | 否 |
| RoutingRule | 重定向规则的元素。一条重定向规则包含一个*Condition*和一个*Redirect*，当*Condition*匹配时，*Redirect生效*。<br>类型：Container<br>父节点：RoutingRules<br>元素中至少要有一个*RoutingRule*元素 | 是 |
| Condition | 描述重定向规则匹配的条件的元素。<br>类型：Container<br>父节点：RoutingRule | 否 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| KeyPrefixEquals | 描述当重定向生效时对象名的前缀。<br>例如：<br>● 重定向ExamplePage.html对象的请求，*KeyPrefixEquals*设为ExamplePage.html。<br>类型：String<br>父节点：Condition<br>条件：父节点Condition存在，并且兄弟节点HttpErrorCodeReturnedEquals不存在。如果设定了两个条件，只有都匹配时，*Redirect*才生效。 | 否 |
| HttpErrorCodeReturnedEquals | 描述*Redirect*生效时的HTTP错误码。当发生错误时，如果错误码等于这个值，那么*Redirect*生效。<br>例如：<br>● 当返回的http错误码为404时重定向到NotFound.html，可以将*Condition*中的HttpErrorCodeReturnedEquals设置为404，*Redirect*中的ReplaceKeyWith设置为NotFound.html。<br>类型：String<br>父节点：Condition<br>条件：父节点Condition存在，并且兄弟节点KeyPrefixEquals不存在。如果设定了多个条件，需要同时匹配所有的条件，*Redirect*才可生效。 | 否 |
| Redirect | 重定向信息的元素。可以重定向到另一个站点、另一个页面或使用另一个协议。当事件或错误发生时，可以指定不同的返回码。<br>类型：Container<br>父节点：RoutingRule | 是 |
| Protocol | 描述重定向请求时使用的协议。<br>类型：String<br>父节点：Redirect<br>可选值：http、https<br>条件：有其他兄弟节点存在时非必选 | 否 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| HostName | 描述重定向请求时使用的站点名。<br>类型：String<br>父节点：Redirect<br>条件：有其他兄弟节点存在时非必选 | 否 |
| ReplaceKeyPrefixWith | 描述重定向请求时使用的对象名前缀，请求中的对象名会将KeyPrefixEquals的内容替换为ReplaceKeyPrefixWith的内容。<br>例如：<br>想把所有对docs（目录下的对象）的请求重定向到documents（目录下的对象），可以将Condition中的KeyPrefixEquals设置为docs，Redirect中的ReplaceKeyPrefixWith设置为documents。那么对于对象名称为"docs/a.html"，重定向的结果为"documents/a.html"。<br>类型：String<br>父节点：Redirect<br>条件：有其他兄弟节点存在时非必选，不可与*ReplaceKeyWith*同时存在 | 否 |
| ReplaceKeyWith | 描述重定向请求时使用的对象名，请求中的整个对象名会被替换为ReplaceKeyWith的内容。<br>例如：<br>想把所有对"docs"目录下的所有对象的请求重定向到"documents/error.html"，可以将Condition中的KeyPrefixEquals设置为docs，Redirect中的ReplaceKeyWith设置为"documents/error.html"。那么对于对象名称为"docs/a.html"和"docs/b.html"，重定向的结果都为"documents/error.html"。<br>类型：String<br>父节点：Redirect<br>条件：有其他兄弟节点存在时非必选，不可与*ReplaceKeyPrefixWith*同时存在 | 否 |
| HttpRedirectCode | 描述响应中的HTTP状态码。<br>类型：String<br>父节点：Redirect<br>条件：有其他兄弟节点存在时非必选 | 否 |

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息不带消息元素。

### 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

### 请求示例：将该桶的所有请求重定向至其他桶或 URL

```
PUT /?website HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:40:29 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:pUK7Yp0yebnq4P6gqzVjoS7whoM=
Content-Length: 194

<WebsiteConfiguration xmlns="http://obs.example.com/doc/2015-06-30/">
  <RedirectAllRequestsTo>
    <HostName>www.example.com</HostName>
  </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

### 响应示例：将该桶的所有请求重定向至其他桶或 URL

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000164360D144670B9D02AABC6
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSItqMZ/AoFUX97l1xx8s67V3cCQtXWk
Date: WED, 01 Jul 2015 03:40:29 GMT
Content-Length: 0
```

## 5.3.2 获取桶的网站配置

### 功能介绍

获取该桶设置的网站配置信息。

要正确执行此操作，需要确保执行者有GetBucketWebsite执行权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

### 请求消息样式

```
GET /?website HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: type
Content-Length: length
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WebsiteConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <RedirectAllRequestsTo>
      <HostName>hostName</HostName>
   </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素与设置桶的网站配置请求的请求消息元素一致，见**请求消息元素**。

## 错误响应消息

此请求可能的特殊错误如下**表5-55**描述。

**表 5-55** 特殊错误

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| NoSuchWebsiteConfiguration | 桶的Website配置不存在 | 404 Not Found |

其余错误已经包含在**表6-2**中。

## 请求示例

```
GET /?website HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:41:54 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:Yxt1Ru+feHE0S94R7dcBp+hfLnI=
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000164363442EC03A8CA3DD7F5
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSFbGOmlN0BVp1kbwN3har8jbVvtKEKN
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:41:54 GMT
Content-Length: 250

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WebsiteConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <RedirectAllRequestsTo>
    <HostName>www.example.com</HostName>
  </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

## 5.3.3 删除桶的网站配置

### 功能介绍

删除指定桶的网站配置信息。

要正确执行此操作，需要确保执行者有DeleteBucketWebsite权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

### 请求消息样式

```
DELETE /?website HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求消息中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: type
Content-Length: length
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
DELETE /?website HTTP/1.1
User-Agent: curl/7.29.0
Host: bucketname.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:44:37 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:AZ1b0N5eLknxNOe/c0BISV1bEqc=
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: BF260000016436378623OE2001DC0807
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSFUG4fEyDRgzUiEY2i71bJndBCy+wUZ
Date: WED, 01 Jul 2015 03:44:37 GMT
```

# 5.3.4 设置桶的 CORS 配置

## 功能介绍

CORS（Cross Origin Resource Sharing），即跨域资源共享，是W3C标准化组织提出的一种规范机制，允许客户端的跨域请求的配置。在通常的网页请求中，由于安全策略SOP（Same Origin Policy）的存在，一个网站的脚本和内容是不能与另一个网站的脚本和内容发生交互的。

OBS允许在桶内保存静态的网页资源，在正确的使用下，OBS的桶可以成为网站资源（请参见**设置桶的网站配置**）。只有进行了适当的CORS配置，OBS中的网站才能响应另一个网站的跨域请求。

典型的应用场景如下：

- 你可以使用CORS支持，使用JavaScript和HTML 5来构建Web应用，直接访问OBS中的资源，而不再需要代理服务器做中转。
- 可以使用HTML 5中的拖拽功能，直接向OBS上传文件，展示上传进度，或是直接从Web应用中更新内容。
- 托管在不同域中的外部网页、样式表和HTML 5应用，现在可以引用存储在OBS中的Web字体或图片，让这些资源能被多个网站共享。

要正确执行此操作，需要确保执行者有PutBucketCORS权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
PUT /?cors HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Length: length
Date: date
Authorization: authorization
Content-MD5: MD5
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
   <CORSRule>
      <ID>id</ID>
      <AllowedMethod>method</AllowedMethod>
      <AllowedOrigin>origin</AllowedOrigin>
```

```
    <AllowedHeader>header</AllowedHeader>
    <MaxAgeSeconds>seconds</MaxAgeSeconds>
    <ExposeHeader>header</ExposeHeader>
  </CORSRule>
</CORSConfiguration>
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头外加CORS请求消息头，具体参见**表3-3**和**表5-56**。

**表 5-56** CORS 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。<br>类型：String<br>示例：n58IG6hfM7vqI4K0vnWpog== | 是 |

## 请求消息元素

在此请求中，需要在请求的消息体中配置桶的CORS配置信息。配置信息以XML格式上传，具体的配置元素如**表5-57**描述。

**表 5-57** CORS 配置元素

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| CORSConfiguration | CORSRules的根节点，最大不超过64 KB。<br>类型：Container<br>父节点：无。 | 是 |
| CORSRule | CORS规则，CORSConfiguration下可最多包含100个规则。<br>类型：Container<br>父节点：CORSConfiguration。 | 是 |
| ID | 一条Rule的标识，由不超过255个字符的字符串组成。<br>类型：String<br>父节点：CORSRule。 | 否 |

| 名称 | 描述 | 是否必选 |
| --- | --- | --- |
| AllowedMethod | CORS规则允许的Method。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE<br>父节点：CORSRule。 | 是 |
| AllowedOrigin | CORS规则允许的Origin（表示域名的字符串，仅支持英文域名），通过正则表达式进行匹配，可以带一个匹配符“*”。每一个AllowedOrigin可以带最多一个“*”通配符。<br>类型：String<br>父节点：CORSRule。 | 是 |
| AllowedHeader | 配置CORS请求中允许携带的“Access-Control-Request-Headers”头域。如果一个请求带了“Access-Control-Request-Headers”头域，则只有匹配上AllowedHeader中的配置才认为是一个合法的CORS请求（通过正则表达式进行匹配）。每一个AllowedHeader可以带最多一个“*”通配符，不可出现空格。<br>类型：String<br>父节点：CORSRule。 | 否 |
| MaxAgeSeconds | 客户端可以缓存的CORS响应时间，以秒为单位。<br>每个CORSRule可以包含至多一个MaxAgeSeconds，可以设置为负值。<br>类型：Integer<br>父节点：CORSRule。 | 否 |
| ExposeHeader | CORS响应中带的附加头域，给客户端提供额外的信息，不可出现空格。<br>类型：String<br>父节点：CORSRule。 | 否 |

## 响应消息样式

```
HTTP/1.1 status_code

Date: date
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /?cors HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:51:52 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:lq7BGoqE9yyhdEwE6KojJ7ysVxU=
Content-MD5: NGLzvw81f/A2C9PiGO0aZQ==
Content-Length: 617

<?xml version="1.0" encoding="utf-8"?>
<CORSConfiguration>
  <CORSRule>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>
    <AllowedOrigin>www.example.com</AllowedOrigin>
    <AllowedHeader>AllowedHeader_1</AllowedHeader>
    <AllowedHeader>AllowedHeader_2</AllowedHeader>
    <MaxAgeSeconds>100</MaxAgeSeconds>
    <ExposeHeader>ExposeHeader_1</ExposeHeader>
    <ExposeHeader>ExposeHeader_2</ExposeHeader>
  </CORSRule>
</CORSConfiguration>
```

## 响应示例

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF26000001643627112BD03512FC94A4
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSYi6wLC4bkrvuS9sqnlRjxK2a5Fe3ry
Date: WED, 01 Jul 2015 03:51:52 GMT
Content-Length: 0
```

# 5.3.5 获取桶的 CORS 配置

## 功能介绍

获取指定桶的CORS配置信息。

要正确执行此操作，需要确保执行者有GetBucketCORS权限。默认情况下只有桶的所有者可以执行此操作，也可以通过设置桶策略或用户策略授权给其他用户。

## 请求消息样式

```
GET /?cors HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体参见表3-3。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type:  application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CORSConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <CORSRule>
      ...
   </CORSRule>
</CORSConfiguration>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

在此请求返回的响应消息体中包含的配置元素如下表5-58描述。

**表 5-58** CORS 配置元素

| 名称 | 描述 |
| --- | --- |
| CORSConfiguration | CORSRules的根节点，最大不超过64 KB。<br>类型：Container<br>父节点：无。 |
| CORSRule | CORS规则，CORSConfiguration下可最多包含100个规则。<br>类型：Container<br>父节点：CORSConfiguration。 |
| ID | 一条Rule的标识，由不超过255个字符的字符串组成。<br>类型：String<br>父节点：CORSRule。 |
| AllowedMethod | CORS规则允许的Method。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE<br>父节点：CORSRule。 |

| 名称 | 描述 |
| --- | --- |
| AllowedOrigin | CORS规则允许的Origin（表示域名的字符串），可以带一个匹配符”*”。每一个AllowedOrigin可以带最多一个“*”通配符。<br>类型：String<br>父节点：CORSRule。 |
| AllowedHeader | 配置CORS请求中允许携带的“Access-Control-Request-Headers”头域。如果一个请求带了“Access-Control-Request-Headers”头域，则只有匹配上AllowedHeader中的配置才认为是一个合法的CORS请求。每一个AllowedHeader可以带最多一个“*”通配符，不可出现空格。<br>类型：String<br>父节点：CORSRule。 |
| MaxAgeSeconds | 客户端可以缓存的CORS响应时间，以秒为单位。<br>每个CORSRule可以包含至多一个MaxAgeSeconds，可以设置为负值。<br>类型：Integer<br>父节点：CORSRule。 |
| ExposeHeader | CORS响应中带的附加头域，给客户端提供额外的信息，不可出现空格。<br>类型：String<br>父节点：CORSRule。 |

## 错误响应消息

此请求可能的特殊错误如下表5-59描述。

**表 5-59** 特殊错误

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| NoSuchCORSConfiguration | 桶的CORS配置不存在 | 404 Not Found |

其余错误已经包含在表6-2中。

## 请求示例

```
GET /?cors HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:54:36 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:WJGghTrPQQXRuCx5go1fHyE+Wwg=
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000164363593F10738B80CACBE
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSpngvwC5TskcLGh7Fz5KRmCFIayuY8p
Content-Type: application/xml
Date: WED, 01 Jul 2015 03:54:36 GMT
Content-Length: 825

<?xml version="1.0" encoding="utf-8"?>
<CORSConfiguration xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <CORSRule>
    <ID>783fc6652cf246c096ea836694f71855</ID>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedOrigin>obs.example.com</AllowedOrigin>
    <AllowedOrigin>www.example.com</AllowedOrigin>
    <AllowedHeader>AllowedHeader_1</AllowedHeader>
    <AllowedHeader>AllowedHeader_2</AllowedHeader>
    <MaxAgeSeconds>100</MaxAgeSeconds>
    <ExposeHeader>ExposeHeader_1</ExposeHeader>
    <ExposeHeader>ExposeHeader_2</ExposeHeader>
  </CORSRule>
</CORSConfiguration>
```

## 5.3.6 删除桶的 CORS 配置

### 功能介绍

删除指定桶的CORS配置信息。删除后桶以及桶中的对象将不能再被其他网址发送的请求访问。

要正确执行此操作，需要确保执行者有PutBucketCORS权限(在桶策略中配置删除桶CORS权限时和设置桶CORS使用同一个Action)。

### 请求消息样式

```
DELETE /?cors HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求消息中不使用消息参数。

### 请求消息头

该请求使用公共消息头，具体参见**表3-3**。

### 请求消息元素

该请求消息中不使用消息元素。

### 响应消息样式

```
HTTP/1.1 status_code
Date: date
```

```
Content-Type:  application/xml
Content-Length: length
```

### 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带消息元素。

### 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

### 请求示例

```
DELETE /?cors HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 03:56:41 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:mKUs/uIPb8BP0ZhvMd4wEy+EbiI=
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: BF26000001643639F290185BB27F793A
x-obs-id-2: 32AAAQAAEAABSAAgAAEAABAAAQAAEAABCSLWMRFJfckapW+ktT/+1AnAz7XlNU0b
Date: WED, 01 Jul 2015 03:56:41 GMT
```

## 5.3.7 OPTIONS 桶

### 功能介绍

OPTIONS，称为预请求，是客户端发送给服务端的一种请求，通常被用于检测客户端是否具有对服务端进行操作的权限。只有当预请求成功返回，客户端才开始执行后续的请求。

OBS允许在桶内保存静态的网页资源，在正确的使用下，OBS的桶可以成为网站资源。在这种使用场景下，OBS中的桶作为服务端，需要处理客户端发送的OPTIONS预请求。

要处理OPTIONS，OBS的桶必须已经配置CORS，关于CORS的使用说明，请参见章节**设置桶的CORS配置**。

### 与 OPTIONS 对象的区别

OPTIONS对象需在URL中指定对象名；OPTIONS桶提交的URL为桶域名，无需指定对象名。两者的请求行分别为：

```
OPTIONS /object HTTP/1.1
OPTIONS / HTTP/1.1
```

### 请求消息样式

```
OPTIONS / HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: authorization
Origin: origin
Access-Control-Request-Method: method
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用的消息头如下表5-60所示。

**表 5-60** OPTIONS 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Origin | 预请求指定的跨域请求Origin（通常为域名）。<br>类型：String | 是 |
| Access-Control-Request-Method | 实际请求可以带的HTTP方法，可以带多个方法头域。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE | 是 |
| Access-Control-Request-Headers | 实际请求可以带的HTTP头域，可以带多个头域。<br>类型：String | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: application/xml
Access-Control-Allow-Origin: origin
Access-Control-Allow-Methods: method
Access-Control-Allow-Header: header
Access-Control-Max-Age: time
Access-Control-Expose-Headers: header
Date: date
Content-Length: length
```

## 响应消息头

该响应使用的消息头如下表5-61所示。

**表 5-61** CORS 响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| Access-Control-Allow-Origin | 如果请求的Origin满足服务端的CORS配置，则在响应中包含这个Origin。<br>类型：String |
| Access-Control-Allow-Headers | 如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。<br>类型：String |
| Access-Control-Max-Age | 服务端CORS配置中的MaxAgeSeconds。<br>类型：Integer |
| Access-Control-Allow-Methods | 如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE |
| Access-Control-Expose-Headers | 服务端CORS配置中的ExposeHeader。<br>类型：String |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

此请求可能的特殊错误如下表5-62描述。

**表 5-62** 特殊错误

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| Bad Request | Invalid Access-Control-Request-Method: null<br>桶配置了CORS，OPTIONS桶时，没有加入method头域。 | 400 BadRequest |
| Bad Request | Insufficient information. Origin request header needed.<br>桶配置了CORS，OPTIONS桶时，没有加入origin头域。 | 400 BadRequest |

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| AccessForbidden | CORSResponse: This CORS request is not allowed. This is usually because the evalution of Origin, request method / Access-Control-Request-Method or Access-Control-Requet-Headers are not whitelisted by the resource's CORS spec.<br>桶配置了CORS，OPTIONS桶时，Origin、method、Headers与任一rule匹配不上。 | 403 Forbidden |

其余错误已经包含在表6-2中。

## 请求示例

```
OPTIONS / HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:02:15 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:7RqP1vjemo6U+Adv9/Y6eGzWrzA=
Origin: www.example.com
Access-Control-Request-Method: PUT
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF260000016436314E8FF936946DBC9C
Access-Control-Allow-Origin: www.example.com
Access-Control-Allow-Methods: POST,GET,HEAD,PUT,DELETE
Access-Control-Max-Age: 100
Access-Control-Expose-Headers: ExposeHeader_1,ExposeHeader_2
Access-Control-Allow-Credentials: true
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCTlYimJvOyJncCLNm5y/iz6MAGLNxTuS
Date: WED, 01 Jul 2015 04:02:15 GMT
Content-Length: 0
```

# 5.3.8 OPTIONS 对象

## 功能介绍

请参见章节 **OPTIONS桶**。

## 与 OPTIONS 桶的区别

OPTIONS对象需在URL中指定对象名；OPTIONS桶提交的URL为桶域名，无需指定对象名。两者的请求行分别为：

```
OPTIONS /object HTTP/1.1
OPTIONS / HTTP/1.1
```

## 请求消息样式

```
OPTIONS /object HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: authorization
Origin: origin
Access-Control-Request-Method: method
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该请求使用的消息头如下表5-63所示。

**表 5-63** OPTIONS 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Origin | 预请求指定的跨域请求Origin（通常为域名）。<br>类型：String | 是 |
| Access-Control-Request-Method | 实际请求可以带的HTTP方法，可以带多个方法头域。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE | 是 |
| Access-Control-Request-Headers | 实际请求可以带的HTTP头域，可以带多个头域。<br>类型：String | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: type
Access-Control-Allow-Origin: origin
Access-Control-Allow-Methods: method
Access-Control-Allow-Header: header
Access-Control-Max-Age: time
Access-Control-Expose-Headers: header
Date: date
Content-Length: length
```

## 响应消息头

该请求使用的消息头如下表5-64所示。

**表 5-64** CORS 请求消息头

| 消息头名称 | 描述 |
| --- | --- |
| Access-Control-Allow-Origin | 如果请求的Origin满足服务端的CORS配置，则在响应中包含这个Origin。<br>类型：String |
| Access-Control-Allow-Headers | 如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。<br>类型：String |
| Access-Control-Max-Age | 服务端CORS配置中的MaxAgeSeconds。<br>类型：Integer |
| Access-Control-Allow-Methods | 如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE |
| Access-Control-Expose-Headers | 服务端CORS配置中的ExposeHeader。<br>类型：String |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

此请求可能的特殊错误如下表5-65描述。

**表 5-65** 特殊错误

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| Bad Request | Invalid Access-Control-Request-Method: null<br>桶配置了CORS，OPTIONS桶时，没有加入method头域。 | 400 BadRequest |
| Bad Request | Insufficient information. Origin request header needed.<br>桶配置了CORS，OPTIONS桶时，没有加入origin头域。 | 400 BadRequest |

| 错误码 | 描述 | HTTP状态码 |
| --- | --- | --- |
| AccessForbidden | CORSResponse: This CORS request is not allowed. This is usually because the evalution of Origin, request method / Access-Control-Request-Method or Access-Control-Requet-Headers are not whitelisted by the resource's CORS spec.<br>桶配置了CORS，OPTIONS桶时，Origin、method、Headers与任一rule匹配不上。 | 403 Forbidden |

其余错误已经包含在表6-2中。

## 请求示例

```
OPTIONS /object_1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:02:19 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:bQZG9c2aokAJsHOOkuVBK6cHZZQ=
Origin: www.example.com
Access-Control-Request-Method: PUT
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF26000001643632D12EFCE1C1294555
Access-Control-Allow-Origin: www.example.com
Access-Control-Allow-Methods: POST,GET,HEAD,PUT,DELETE
Access-Control-Max-Age: 100
Access-Control-Expose-Headers: ExposeHeader_1,ExposeHeader_2
Access-Control-Allow-Credentials: true
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCS+DXV4zZetbTqFehhEcuXywTa/mi3T3
Date: WED, 01 Jul 2015 04:02:19 GMT
Content-Length: 0
```

# 5.4 对象操作

## 5.4.1 PUT 上传

### 功能介绍

用户在OBS系统中创建了桶之后，可以采用PUT操作的方式将对象上传到桶中。上传对象操作是指在指定的桶内增加一个对象，执行该操作需要用户拥有桶的写权限。

**说明**

同一个桶中存储的对象名是唯一的。

在桶未开启多版本的情况下，如果在指定的桶内已经有相同的对象键值的对象，用户上传的新对象会覆盖原来的对象；为了确保数据在传输过程中没有遭到破坏，用户可

以在请求消息头中加入Content-MD5参数。在这种情况下，OBS收到上传的对象后，会对对象进行MD5校验，如果不一致则返回出错信息。

用户还可以在上传对象时指定x-obs-acl参数，设置对象的权限控制策略。如果匿名用户在上传对象时未指定x-obs-acl参数，则该对象默认可以被所有OBS用户访问。

单次上传对象大小范围是[0, 5GB]，如果需要上传超过5GB的大文件，需要通过**多段操作**来分段上传。

OBS没有文件夹的概念。为了使用户更方便进行管理数据，OBS提供了一种方式模拟文件夹：通过在对象的名称中增加“/”，例如“test/123.jpg”。此时，“test”就被模拟成了一个文件夹，“123.jpg”则模拟成“test”文件夹下的文件名了，而实际上，对象名称（Key）仍然是“test/123.jpg”。此类命名方式的对象，在控制台上会以文件夹的形式展示。当您上传此类方式命名的对象时，如果其大小不为0，控制台上会展示为空文件夹，但是存储总用量为对象大小。

## 与 POST 上传的区别

PUT上传中参数通过请求头域传递；POST上传则作为消息体中的表单域传递。

PUT上传需在URL中指定对象名；POST上传提交的URL为桶域名，无需指定对象名。两者的请求行分别为：

```
PUT /ObjectName HTTP/1.1
POST / HTTP/1.1
```

**说明**

使用PUT上传请求时，消息体中如果使用POST格式，则上传到OBS中的对象会以表单形式呈现。

关于POST上传的更多详细信息，请参考**POST上传**。

## 多版本

如果桶的多版本状态是开启的，系统会自动为对象生成一个唯一的版本号，并且会在响应报头x-obs-version-id返回该版本号。如果桶的多版本状态是暂停的，则对象的版本号为**null**。关于桶的多版本状态，参见**设置桶的多版本状态**。

## 请求消息样式

```
PUT /ObjectName HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Type: application/xml
Content-Length: length
Authorization: authorization
Date: date
<Optional Additional Header>
<object Content>
```

## 请求消息参数

该请求消息中不使用参数。

## 请求消息头

该请求使用公共的消息头，具体请参见**表3-3**。该请求可以使用附加的消息头，具体如**表5-66**所示。

**说明**

OBS支持在上传对象时在请求里携带HTTP协议规定的6个请求头：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type、Content-Language。如果上传Object时设置了这些请求头，OBS会直接将这些头域的值保存下来。这6个值也可以通过OBS提供的修改对象元数据API接口进行修改。在该Object被下载或者HEAD的时候，这些保存的值将会被设置到对应的HTTP头域中返回客户端。

**表 5-66** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Content-MD5 | 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过base64编码后得到的字符串。<br>类型：String<br>示例：n58IG6hfM7vqI4K0vnWpog==。 | 否 |
| x-obs-acl | 创建对象时，可以加上此消息头设置对象的权限控制策略，使用的策略为预定义的常用策略，包括：private；public-read；public-read-write<br>类型：String<br>说明：字符串形式的预定义策略。<br>示例：x-obs-acl: public-read。 | 否 |
| x-obs-grant-read | 创建对象时，使用此头域授权租户下所有用户有读对象和获取对象元数据的权限。<br>类型：String<br>示例：x-obs-grant-read: id=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-grant-read-acp | 创建对象时，使用此头域授权租户下所有用户有获取对象ACL的权限。<br>类型：String<br>示例：x-obs-grant-read-acp: id=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-grant-write-acp | 创建对象时，使用此头域授权domain下所有用户有写对象ACL的权限。<br>类型：String<br>示例：x-obs-grant-write-acp: id=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-grant-full-control | 创建对象时，使用此头域授权domain下所有用户有读对象、获取对象元数据、获取对象ACL、写对象ACL的权限。<br>类型：String<br>示例：x-obs-grant-full-control: id=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-meta-* | 创建对象时，可以在HTTP请求中加入以“x-obs-meta-”开头的消息头，用来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回消息的头中出现。<br>类型：String<br>示例：x-obs-meta-test: test metadata<br>约束：自定义元数据key-value对都必须符合US-ASCII。 | 否 |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL，OBS将这个值从头域中取出，保存在对象的元数据中。<br>例如，重定向请求到桶内另一对象：<br>x-obs-website-redirect-location:/anotherPage.html<br>或重定向请求到一个外部URL：<br>x-obs-website-redirect-location:http://www.example.com/<br>类型：String<br>默认值：无<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2KB。 | 否 |
| success-action-redirect | 此参数的值是一个URL，用于指定当此次请求操作成功响应后的重定向的地址。<br>● 如果此参数值有效且操作成功，响应码为303，Location头域由此参数以及桶名、对象名、对象的ETag组成。<br>● 如果此参数值无效，则OBS忽略此参数的作用，Location头域为对象地址，响应码根据操作成功或失败正常返回。<br>类型：String | 否 |
| x-obs-expires | 表示对象的过期时间，单位是天。过期之后对象会被自动删除。（从对象创建时间开始计算）<br>此字段对于每个对象支持上传时配置，也支持后期通过修改元数据接口修改。<br>类型：Integer<br>示例：x-obs-expires:3 | 否 |

## 请求消息元素

该请求消息中不使用消息元素，在消息体中带的是对象的数据。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Length: length
Content-Type: type
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如**表5-67**中的消息头。

**表 5-67** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 对象的版本号。如果桶的多版本状态为开启，则会返回对象的版本号。<br>类型：String |

## 响应消息元素

该请求的响应消息不带消息元素。

## 错误响应消息

该请求的返回无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例：上传对象

```
PUT /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:11:15 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:gYqplLq30dEX7GMi2qFWyjdFsyw=
Content-Length: 10240
Expect: 100-continue

[1024 Byte data content]
```

## 响应示例：上传对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000164364C10805D385E1E3C67
ETag: "d41d8cd98f00b204e9800998ecf8427e"
x-obs-id-2: 32AAAWJAMAABAAAQAAEAABAAAQAAEAABCTzu4Jp2lquWuXsjnLyPPiT3cfGhqPoY
Date: WED, 01 Jul 2015 04:11:15 GMT
Content-Length: 0
```

## 请求示例：上传对象的同时设置 ACL

```
PUT /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:13:55 GMT
x-obs-grant-read:id=52f24s3593as5730ea4f722483579ai7,id=a93fcas852f24s3596ea8366794f7224
```

```
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:gYqplLq30dEX7GMi2qFWyjdFsyw=
Content-Length: 10240
Expect: 100-continue

[1024 Byte data content]
```

## 响应示例：上传对象的同时设置 ACL

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BB7800000164845759E4F3B39ABEE55E
ETag: "d41d8cd98f00b204e9800998ecf8427e"
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSReVRNuas0knI+Y96iXrZA7BLUgj06Z
Date: WED, 01 Jul 2015 04:13:55 GMT
Content-Length: 0
```

## 请求示例：桶开启多版本时上传对象

```
PUT /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:17:12 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:uFVJhp/dJqj/CJIVLrSZ0gpw3ng=
Content-Length: 10240
Expect: 100-continue

[1024 Byte data content]
```

## 响应示例：桶开启多版本时上传对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: DCD2FC9CAB78000001439A51DB2B2577
ETag: "d41d8cd98f00b204e9800998ecf8427e"
X-OBS-ID-2: GcVgfeOJHx8JZHTHrRqkPsbKdB583fYbr3RBbHT6mMrBstReVILBZbMAdLiBYy1l
Date: WED, 01 Jul 2015 04:17:12 GMT
x-obs-version-id: AAABQ4q2M9_c0vycq3gAAAAAVURTRkha
Content-Length: 0
```

## 请求示例：上传对象时携带 MD5

```
PUT /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:17:50 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:uFVJhp/dJqj/CJIVLrSZ0gpw3ng=
Content-Length: 10
Content-MD5: 6Afx/PgtEy+bsBjKZzihnw==
Expect: 100-continue

1234567890
```

## 响应示例：上传对象时携带 MD5

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BB7800000164B165971F91D82217D105
X-OBS-ID-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCSEKhBpS4BB3dSMNqMtuNxQDD9XvOw5h
ETag: "1072e1b96b47d7ec859710068aa70d57"
Date: WED, 01 Jul 2015 04:17:50 GMT
Content-Length: 0
```

## 请求示例：上传时配置 website 实现下载对象重定向

**当桶设置了Website配置，您可以在上传对象时进行以下设置，设置后用户在下载对象时会重定向**

```
PUT /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:17:12 GMT
x-obs-website-redirect-location: http://www.example.com/
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:uFVJhp/dJqj/CJIVLrSZ0gpw3ng=
Content-Length: 10240
Expect: 100-continue

[1024 Byte data content]
```

## 响应示例：上传时配置 website 实现下载对象重定向

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: DCD2FC9CAB78000001439A51DB2B2577
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCTmxB5ufMj/7/GzP8TFwTbp33u0xhn2Z
ETag: "1072e1b96b47d7ec859710068aa70d57"
Date: WED, 01 Jul 2015 04:17:12 GMT
x-obs-version-id: AAABQ4q2M9_c0vycq3gAAAAAVURTRkha
Content-Length: 0
```

## 请求示例：在 URL 中携带签名并上传对象

```
PUT /object02?
AccessKeyId=H4IPJX0TQTHTHEBQQCEC&Expires=1532688887&Signature=EQmDuOhaLUrzrzRNZxwS72CXeXM%3D HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Content-Length: 1024

[1024 Byte data content]
```

## 响应示例：在 URL 中携带签名并上传对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: DCD2FC9CAB78000001439A51DB2B2577
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCTmxB5ufMj/7/GzP8TFwTbp33u0xhn2Z
ETag: "1072e1b96b47d7ec859710068aa70d57"
Date: Fri, 27 Jul 2018 10:52:31 GMT
x-obs-version-id: AAABQ4q2M9_c0vycq3gAAAAAVURTRkha
Content-Length: 0
```

# 5.4.2 POST 上传

## 功能介绍

上传对象操作是指在指定的桶内增加一个对象，执行该操作需要用户拥有桶的写权限。

**说明**

同一个桶中存储的对象名是唯一的。

在桶未开启多版本的情况下，如果在指定的桶内已经有相同的对象键值的对象，用户上传的新对象会覆盖原来的对象；为了确保数据在传输过程中没有遭到破坏，用户可

以在表单域中加入Content-MD5参数。在这种情况下，OBS收到上传的对象后，会对对象进行MD5校验，如果不一致则返回出错信息。用户还可以在上传对象时指定x-obs-acl参数，设置对象的权限控制策略。

用户除了可以用PUT直接上传对象外，还可以使用POST上传对象。

单次上传对象大小范围是[0, 5GB]，如果需要上传超过5GB的大文件，需要通过**多段操作**来分段上传。

## 与 PUT 上传的区别

PUT上传中参数通过请求头域传递；POST上传则作为消息体中的表单域传递。

PUT上传需在URL中指定对象名；POST上传提交的URL为桶域名，无需指定对象名。两者的请求行分别为：

```
PUT /ObjectName HTTP/1.1
POST / HTTP/1.1
```

关于PUT上传的更多详细信息，请参考**PUT上传**。

## 多版本

如果桶的多版本状态是开启的，系统会自动为对象生成一个唯一的版本号；如果桶的多版本状态是暂停的，则系统生成的对象版本号为**null**，并由响应报头x-obs-version-id返回该版本号。关于桶的多版本状态，参见**设置桶的多版本状态**。

## 请求消息样式

```
POST / HTTP/1.1
Host: bucketname.obs.region.example.com
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length


--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-obs-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-obs-meta-tag"

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

```
access-key-id
--9431149156168
Content-Disposition: form-data; name="policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to OBS
--9431149156168--
```

## 请求消息参数

该请求消息中不使用参数。

## 请求消息头

该请求使用公共的消息头，具体请参见**表3-3**。

如果想要获取CORS配置信息，则需要使用的消息头如下**表5-68**所示。

**表 5-68** 获取 CORS 配置的请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Origin | 预请求指定的跨域请求Origin（通常为域名）。<br>类型：String | 是 |
| Access-Control-Request-Headers | 实际请求可以带的HTTP头域，可以带多个头域。<br>类型：String | 否 |

## 请求消息元素

该请求消息的消息元素以表单形式组织，表单字段的具体含义如**表5-69**所示。

**表 5-69** 请求消息表单元素

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| file | 上传的对象内容。文件名与文件路径均会被忽略，不会作为对象名称。对象名称是另一参数key的值。<br>类型：二进制或文本类型。<br>约束条件：此参数必须为最后一个参数，否则此参数之后的参数会被丢弃；一个请求中只能含有一个file参数。 | 是 |
| key | 通过此请求创建的对象名称。<br>类型：String | 是 |
| AccessKeyId | 用来指明请求发起者的Access Key。<br>类型：String<br>约束条件：如果该请求包括安全策略参数policy或signature时，则必须包括此参数。 | 是，有条件 |
| policy | 该请求的安全策略描述。policy格式请参考**基于浏览器上传的表单中携带签名**章节中policy格式<br>类型：String<br>限制：当Bucket提供了AccessKeyId(或signature)表单域时，则必须包括此参数。 | 是，有条件 |
| signature | 根据StringToSign计算出的签名字符串。<br>类型：String<br>限制：当Bucket提供了AccessKeyId(或policy)表单域时，则必须包括此参数。 | 是，有条件 |
| token | 用来统一指明请求发起者的Access Key，请求签名和请求的安全策略。token的优先级高于单独指定的Access Key，请求签名和请求的安全策略。<br>类型：String<br>示例：<br>HTML中：<input type="text" name="token" value="ak:signature:policy" /> | 否 |
| x-obs-acl | 创建对象时，可以加上此消息头设置对象的权限控制策略，使用的策略为预定义的常用策略，包括：private；public-read；public-read-write；public-read-delivered ; public-read-write-delivered。<br>类型：String<br>示例：<br>POLICY中：{"acl": "public-read" },<br>HTML中：<input type="text" name="acl" value="public-read" /> | 否 |

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-grant-read | 创建对象时，使用此头域授权domain下所有用户有读对象和获取对象元数据的权限。<br>类型：String<br>示例：<br>POLICY中：{"grant-read": "id=domainId1" },<br>HTML中：<input type="text" name="grant-read" value="id=domainId1" /> | 否 |
| x-obs-grant-read-acp | 创建对象时，使用此头域授权domain下所有用户有获取对象ACL的权限。<br>类型：String<br>示例：<br>POLICY中：{"grant-read-acp": "id=domainId1" },<br>HTML中：<input type="text" name="grant-read-acp" value="id=domainId1" /> | 否 |
| x-obs-grant-write-acp | 创建对象时，使用此头域授权domain下所有用户有写对象ACL的权限。<br>类型：String<br>示例：<br>POLICY中：{"grant-write-acp": "id=domainId1" },<br>HTML中：<input type="text" name="grant-write-acp" value="id=domainId1" /> | 否 |
| x-obs-grant-full-control | 创建对象时，使用此头域授权domain下所有用户有读对象、获取对象元数据、获取对象ACL、写对象ACL的权限。<br>类型：String<br>示例：<br>POLICY中：{"grant-full-control": "id=domainId1" },<br>HTML中：<input type="text" name="grant-full-control" value="id=domainId1" /> | 否 |
| Cache-Control,<br>Content-Type,<br>Content-Disposition,<br>Content-Encoding<br>Expires | 这5个参数是HTTP标准消息头，OBS将这些参数记录下来，当用户下载此对象或Head Object时，在响应消息头中携带这些参数。<br>类型：String<br>示例：<br>POLICY中：["starts-with", "$Content-Type", "text/"],<br>HTML中：<input type="text" name="content-type" value="text/plain" /> | 否 |

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| success_action_redirect | 此参数的值是一个URL，用于指定当此次请求操作成功响应后的重定向的地址。<br>● 如果此参数值有效且操作成功，响应码为303，Location头域由此参数以及桶名、对象名、对象的ETag组成。<br>● 如果此参数值无效，则OBS忽略此参数的作用，Location头域为对象地址，响应码根据操作成功或失败正常返回。<br>类型：String<br>示例：<br>POLICY中：{"success_action_redirect": "http://123458.com"},<br>HTML中：<input type="text" name="success_action_redirect" value="http://123458.com" /> | 否 |
| x-obs-meta-* | 创建对象时，可以在HTTP请求中加入“x-obs-meta-”消息头或以“x-obs-meta-”开头的消息头，用来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回消息的消息头中出现。<br>类型：String<br>示例：<br>POLICY中：{" x-obs-meta-test ": " test metadata " },<br>HTML中：<input type="text" name=" x-obs-meta-test " value=" test metadata " /> | 否 |
| success_action_status | 这个参数指定成功响应的状态码，允许设定的值为200，201，204。<br>● 如果此参数值被设定为200或204，OBS响应消息中body为空。<br>● 如果此参数值被设定为201，则OBS响应消息中包含一个XML文档描述此次请求的响应。<br>● 当请求不携带此参数或参数无效时，OBS响应码为204。<br>类型：String<br>示例：<br>POLICY中：["starts-with", "$success_action_status", ""],<br>HTML中：<input type="text" name="success_action_status" value="200" /> | 否 |

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL，OBS将这个值从头域中取出，保存在对象的元数据中。<br>默认值：无<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2K。 | 否 |
| x-obs-expires | 表示对象的过期时间，单位是天。过期之后对象会被自动删除。（从对象最后修改时间开始计算）<br>类型：Integer<br>示例：x-obs-expires:3 | 否 |

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: application/xml
Location: location
Date: date
ETag: etag
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如**表5-70**中的消息头。

**表 5-70** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 对象的版本号。如果桶的多版本状态为开启，则会返回对象的版本号。如果桶的多版本状态为暂停，则会返回**null**。<br>类型：String |
| Access-Control-Allow-Origin | 当桶设置了CORS配置，如果请求的Origin满足服务端的CORS配置，则在响应中包含这个Origin。<br>类型：String |
| Access-Control-Allow-Headers | 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。<br>类型：String |
| Access-Control-Max-Age | 当桶设置了CORS配置，服务端CORS配置中的MaxAgeSeconds。<br>类型：Integer |

| 消息头名称 | 描述 |
| --- | --- |
| Access-Control-Allow-Methods | 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE |
| Access-Control-Expose-Headers | 当桶设置了CORS配置，服务端CORS配置中的ExposeHeader。<br>类型：String |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例：普通 POST 上传

```
POST / HTTP/1.1
Date: WED, 01 Jul 2015 04:15:23 GMT
Host: examplebucket.obs.region.example.com
Content-Type: multipart/form-data; boundary=7db143f50da2
Content-Length: 2424
Origin: www.example.com
Access-Control-Request-Headers:acc_header_1

--7db143f50da2
Content-Disposition: form-data; name="key"

object01
--7db143f50da2
Content-Disposition: form-data; name="acl"

public-read
--7db143f50da2
Content-Disposition: form-data; name="content-type"

text/plain
--7db143f50da2
Content-Disposition: form-data; name="expires"

WED, 01 Jul 2015 04:16:15 GMT
--7db143f50da2
Content-Disposition: form-data; name="AccessKeyId"

14RZT432N80TGDF2Y2G2
--7db143f50da2
Content-Disposition: form-data; name="policy"

ew0KICAiZXhaaXJhdGlvbiI6ICIyMDE1LTA3LTAxVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ1Y2tldCI6ICJleG1hcGxlYnVja2V0IiB9LA0KICAgIHsiYWNsIjogInB1YmxpYy1yZWFkIiB9LA0KICAgIHsiRXhhaXJlcyI6ICIxMDAwIiB9LA0KICAgIFsiZXEiLCAiJGtleSIsICJvYmplY3QwMSJdLA0KICAgIFsic3RhcnRzLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0LyJdLA0KICBdDQp9DQo=
--7db143f50da2
Content-Disposition: form-data; name="signature"
```

```
Vk6rwO0Nq09BLhvNSIYwSJTRQ+k=
--7db143f50da2
Content-Disposition: form-data; name="x-obs-persistent-headers"

test:dmFsdWUx
--7db143f50da2
Content-Disposition: form-data; name="x-obs-grant-read"

id=52f24s3593as5730ea4f722483579xxx
--7db143f50da2
Content-Disposition: form-data; name="x-obs-server-side-encryption"

http://www.example.com/
--7db143f50da2
Content-Disposition: form-data; name="file"; filename="C:\Testtools\UpLoadFiles\object\1024Bytes.txt"
Content-Type: text/plain

01234567890
--7db143f50da2
Content-Disposition: form-data; name="submit"

Upload
--7db143f50da2--
```

## 响应示例：普通 POST 上传

桶配置cors后，响应会包含Access-Control-*的信息。

```
HTTP/1.1 204 No Content
x-obs-request-id: 90E2BA00C26C00000133B442A90063FD
x-obs-id-2: OTBFMkJBMDBDMjZDMDAwMDAxMzNCNDQyQTkwMDYzRkRBQUFBQUFBQWJiYmJiYmJi
Access-Control-Allow-Origin: www.example.com
Access-Control-Allow-Methods: POST,GET,HEAD,PUT
Access-Control-Allow-Headers: acc_header_01
Access-Control-Max-Age: 100
Access-Control-Expose-Headers: exp_header_01
Content-Type: text/xml
Location: http://examplebucket.obs.region.example.com/object01
Date: WED, 01 Jul 2015 04:15:23 GMT
ETag: "ab7abb0da4bca5323ab6119bb5dcd296"
```

## 请求示例：使用 token 进行鉴权

```
POST / HTTP/1.1
Content-Type:multipart/form-data; boundary=9431149156168
Content-Length: 634
Host: examplebucket.obs.region.example.com

--9431149156168
Content-Disposition: form-data; name="key"
obj01

--9431149156168
Content-Disposition: form-data; name="token"
UDSIAMSTUBTEST002538:XsVcTzR2/
A284oE4VH9qPndGcuE=:eyJjb25kaXRpb25zIjogW3siYnVja2V0IjogInRlc3QzMDAzMDU4NzE2NjI2ODkzNjcuMTIifSwgeyJDb250ZW50LVR5cGUiOiAiYXBwbGljYXRpb24veG1sIn0sIFsiZXEiLCAiJGtleSIsICJvYmoudHh0Il1dLCAiZXhwaXJhdGlvbiI6ICIyMDIyLTA5LTA5VDEyOjA5OjI3WiJ9

--9431149156168
Content-Disposition: form-data; name="file"; filename="myfile"
Content-Type: text/plain
01234567890

--9431149156168--
Content-Disposition: form-data; name="submit"
Upload to OBS
```

## 响应示例：使用 token 进行鉴权

```
HTTP/1.1 204 No Content
Server: OBS
Location: http://examplebucket.obs.region.example.com/my-obs-object-key-demo
ETag: "7eda50a430fed940023acb9c4c6a2fff"
x-obs-request-id: 000001832010443D80F30B649B969C47
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCTj0yO9KJd5In+i9pzTgCDVG9vMnk7O/
Date: Fri,09Sep 2022 02: 24:40 GMT
```

## 请求示例：设置对象过期时间

```
POST / HTTP/1.1
Date: WED, 01 Jul 2015 04:15:23 GMT
Host: examplebucket.obs.region.example.com
Content-Type: multipart/form-data; boundary=14882896926023390562087O
Content-Length: 1639
Origin: www.example.com
Access-Control-Request-Headers:acc_header_1

--14882896926023390562087O
Content-Disposition: form-data; name="key"

object01
--14882896926023390562087O
Content-Disposition: form-data; name="ObsAccessKeyId"

55445349414d535455425445535430303033033
--14882896926023390562087O
Content-Disposition: form-data; name="signature"

396246666f6f42793872792f7a3958524f6c44334e4e69763950553d--7db143f50da2
--14882896926023390562087O
Content-Disposition: form-data; name="policy"

65794a6c65484270636d463061573975496a6f694d6a41794d7930774e6930784e565178...
--14882896926023390562087O
Content-Disposition: form-data; name="x-obs-expires"

4
--14882896926023390562087O
Content-Disposition: form-data; name="file"; filename="test.txt"
Content-Type: text/plain

01234567890
--14882896926023390562087O
Content-Disposition: form-data; name="submit"

Upload
--14882896926023390562087O--
```

## 响应示例：设置对象过期时间

```
HTTP/1.1 204 No Content
Server: OBS
Date: Thu, 15 Jun 2023 12:39:03 GMT
Connection: keep-alive
Location: http://examplebucket.obs.region.example.com/my-obs-object-key-demo
x-obs-expiration: expiry-date="Tue, 20 Jun 2023 00:00:00 GMT"
ETag: "d41d8cd98f00b204e9800998ecf8427e"
x-obs-request-id: 00000188BF11049553064911000FC30D
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCSwj2PcBE0YcoLHUDO7GSj+rVByzjflA
x-forward-status: 0x40020000000001
x-dae-api-type: REST.POST.OBJECT
```

## 请求示例：指定状态码

### 指定成功响应的状态码为200

```
POST /srcbucket HTTP/1.1
User-Agent: PostmanRuntime/7.26.8
Accept: */*
Postman-Token: 667dcc44-1c48-41ba-9e41-9f87d8975089
Host: obs.region.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Type: multipart/form-data; boundary=--------------------------285613759795901770404350
Content-Length: 1134

----------------------------285613759795901770404350
Content-Disposition: form-data; name="key"

obj
----------------------------285613759795901770404350
Content-Disposition: form-data; name="ObsAccessKeyId"

XXXXXXXXXXXXXXX000003
----------------------------285613759795901770404350
Content-Disposition: form-data; name="signature"

9rc4bVhDPQ7eHtw17hWtYxLnBWU=
----------------------------285613759795901770404350
Content-Disposition: form-data; name="policy"

eyJleHBpcmF0aW9uIjoiMjAyMy0wNi0xNVQxNDoxMTozNFoiLCAiY29uZGl0aW9ucyI6W3siYnVja2V0Ijoic3JjYnVja2V0MiJ9LHsic3VjY2Vzc19hY3Rpb25fc3RhdHVzIjoiMjAwIn0seyJjb250ZW50LXR5cGUiOiJ0ZXh0L3BsYWluIn0seyJrZXkiOiIzMzMifSxdfQ==
----------------------------285613759795901770404350
Content-Disposition: form-data; name="success_action_status"

200
----------------------------285613759795901770404350
Content-Disposition: form-data; name="file"; filename="test.txt"
Content-Type: text/plain


----------------------------285613759795901770404350
Content-Disposition: form-data; name="submit"

Upload to OBS
----------------------------285613759795901770404350--
```

## 响应示例：指定状态码

### 指定成功响应的状态码为200，响应消息

```
HTTP/1.1 200 OK
Server: OBS
Date: Thu, 15 Jun 2023 13:12:51 GMT
Content-Length: 0
Connection: keep-alive
Location: http://obs.region.example.com/srcbucket/obj
ETag: "d41d8cd98f00b204e9800998ecf8427e"
x-obs-request-id: 00000188BF2FF55F5306426E000FE366
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCScDjcXgZ7oMYSVnZnk4+HrClVwLVPTi
x-forward-status: 0x40020000000001
x-dae-api-type: REST.POST.OBJECT
```

# 5.4.3 复制对象

## 功能介绍

复制对象（Copy Object）特性用来为OBS上已经存在的对象创建一个副本。

当进行复制对象操作时，目标对象默认复制源对象的元数据；用户也可以将目标对象的元数据替换为本次请求中所带的元数据。新建的目标对象不会复制源对象的ACL信息，默认的新建对象的ACL是private，用户可以使用设置ACL的操作接口来重新设定新对象的ACL。

复制对象操作的请求需要通过头域携带拷贝的原桶和对象信息，不能携带消息实体。

目标对象大小范围是[0, 5GB]，如果源对象大小超过5GB，只能使用**拷贝段**功能拷贝部分对象。

**说明**

复制对象的结果不能仅根据HTTP返回头域中的status_code来判断请求是否成功，头域中status_code返回200时表示服务端已经收到请求，且开始处理复制对象请求。复制是否成功会在响应消息的body中，只有body体中有ETag标签才表示成功，否则表示复制失败。

## 多版本

默认情况下，x-obs-copy-source标识复制源对象的最新版本。如果源对象的最新版本是删除标记，则认为该对象已删除。要复制指定版本的对象，可以在x-obs-copy-source请求消息头中携带versionId参数。

如果目标对象的桶的多版本状态是开启的，系统为目标对象生成唯一的版本号（此版本号与源对象的版本号不同），并且会在响应报头x-obs-version-id返回该版本号。如果目标对象的桶的多版本状态是暂停的，则目标对象的版本号为**null**。

**须知**

在桶没有开启多版本的情况下，将源对象object_A复制为目标对象object_B，如果在复制操作之前对象object_B已经存在，复制操作执行之后老对象object_B则会被新复制对象object_B覆盖，复制成功后，只能下载到新的对象object_B，老对象object_B将会被删除。因此在使用copy接口时请确保目标对象不存在或者已无价值，避免因copy导致数据误删除。复制过程中源对象object_A无任何变化。

## 请求消息样式

```
PUT /destinationObjectName HTTP/1.1
Host: bucketname.obs.region.example.com
x-obs-copy-source: /sourceBucket/sourceObject
x-obs-metadata-directive: metadata_directive
x-obs-copy-source-if-match: etag
x-obs-copy-source-if-none-match: etag
x-obs-copy-source-if-unmodified-since: time_stamp
x-obs-copy-source-if-modified-since: time_stamp
Authorization: signature
Date: date
```

## 请求消息参数

该请求消息中不使用消息参数。

## 请求消息头

该消息可以带附加的消息头指定复制的信息，具体如表3-3所示。

**表 5-71** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-acl | 复制对象时，可以加上此消息头设置对象的权限控制策略，使用的策略为预定义的常用策略，包括：private；public-read；public-read-write。<br>类型：String<br>示例：x-obs-acl: acl | 否 |
| x-obs-grant-read | 创建对象时，使用此头域授权domain下所有用户有读对象和获取对象元数据的权限<br>类型：String | 否 |
| x-obs-grant-read-acp | 创建对象时，使用此头域授权domain下所有用户有获取对象ACL的权限。<br>类型：String | 否 |
| x-obs-grant-write-acp | 创建对象时，使用此头域授权domain下所有用户有写对象ACL的权限。<br>类型：String | 否 |
| x-obs-grant-full-control | 创建对象时，使用此头域授权domain下所有用户有读对象、获取对象元数据、获取对象ACL、写对象ACL的权限。<br>类型：String | 否 |
| x-obs-copy-source | 用来指定复制对象操作的源桶名以及源对象名。当源对象存在多个版本时，通过versionId参数指定版本源对象。<br>类型：String<br>约束：中文字符与%，需要进行URLEncode<br>示例：x-obs-copy-source: /source_bucket/sourceObject | 是 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-metadata-directive | 此参数用来指定新对象的元数据是从源对象中复制，还是用请求中的元数据替换。<br>类型：String<br>有效取值：COPY或REPLACE。<br>默认值：COPY。<br>示例：x-obs-metadata-directive: metadata_directive<br>约束条件：如果此参数的值不是COPY或REPLACE，则OBS立即返回400错误；如果用户进行修改元数据操作（源对象与目标对象相同），则此参数只能为REPLACE，否则此请求作为无效请求，服务端响应400。 | 否 |
| x-obs-copy-source-if-match | 只有当源对象的Etag与此参数指定的值相等时才进行复制对象操作，否则返回412（前置条件不满足）。<br>类型：String<br>示例：x-obs-copy-source-if-match: etag<br>约束条件：此参数可与x-obs-copy-source-if-unmodified-since一起使用，但不能与其它条件复制参数一起使用。 | 否 |
| x-obs-copy-source-if-none-match | 只有当源对象的Etag与此参数指定的值不相等时才进行复制对象操作，否则返回412（前置条件不满足）。<br>类型：String<br>示例：x-obs-copy-source-if-none-match: etag<br>约束条件：此参数可与x-obs-copy-source-if-modified-since一起使用，但不能与其它条件复制参数一起使用。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-copy-source-if-unmodified-since | 只有当源对象在此参数指定的时间之后没有修改过才进行复制对象操作，否则返回412（前置条件不满足），此参数可与x-obs-copy-source-if-match一起使用，但不能与其它条件复制参数一起使用。<br>类型：String<br>格式：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。<br>1. EEE, dd MMM yyyy HH:mm:ss z<br>2. EEEE, dd-MMM-yy HH:mm:ss z<br>3. EEE MMM dd HH:mm:ss yyyy<br>对应示例：<br>1. x-obs-copy-source-if-unmodified-since: Sun, 06 Nov 1994 08:49:37 GMT<br>2. x-obs-copy-source-if-unmodified-since: Sunday, 06-Nov-94 08:49:37 GMT<br>3. x-obs-copy-source-if-unmodified-since: Sun Nov 6 08:49:37 1994<br>约束条件：此参数指定的时间不能晚于当前的服务器时间（GMT时间），否则参数不生效。 | 否 |
| x-obs-copy-source-if-modified-since | 只有当源对象在此参数指定的时间之后修改过才进行复制对象操作，否则返回412（前置条件不满足），此参数可与x-obs-copy-source-if-none-match一起使用，但不能与其它条件复制参数一起使用。<br>类型：String<br>格式：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。<br>1. EEE, dd MMM yyyy HH:mm:ss z<br>2. EEEE, dd-MMM-yy HH:mm:ss z<br>3. EEE MMM dd HH:mm:ss yyyy<br>对应示例：<br>1. x-obs-copy-source-if-unmodified-since: Sun, 06 Nov 1994 08:49:37 GMT<br>2. x-obs-copy-source-if-unmodified-since: Sunday, 06-Nov-94 08:49:37 GMT<br>3. x-obs-copy-source-if-unmodified-since: Sun Nov 6 08:49:37 1994<br>约束条件：此参数指定的时间不能晚于当前的服务器时间（GMT时间），否则参数不生效。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL，OBS将这个值从头域中取出，保存在对象的元数据中。<br>类型：String<br>默认值：无<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2K。 | 否 |
| success_action_redirect | 此参数的值是一个URL，用于指定当此次请求操作成功响应后的重定向的地址。<br>• 如果此参数值有效且操作成功，响应码为303，Location头域由此参数以及桶名、对象名、对象的ETag组成。<br>• 如果此参数值无效，则OBS忽略此参数的作用，Location头域为对象地址，响应码根据操作成功或失败正常返回。<br>类型：String | 否 |

其他消息头请参见**表3-3**章节。

## 请求消息元素

该请求在消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: application/xml
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyObjectResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <LastModified>modifiedDate</LastModified>
   <ETag>etagValue</ETag>
</CopyObjectResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如下**表5-72**中的消息头。

**表 5-72** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-copy-source-version-id | 源对象的版本号。<br>类型：String |
| x-obs-version-id | 目标对象的版本号。<br>类型：String |

## 响应消息元素

该请求的响应消息通过消息元素来返回复制结果，元素具体含义如表5-73所示。

**表 5-73** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| CopyObjectResult | 复制对象结果的Container。<br>类型：XML |
| LastModified | 对象上次修改的时间。<br>类型：String |
| ETag | 新对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。<br>类型：String |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例：普通复制

**普通对象拷贝，将桶bucket中的对象srcobject拷贝到桶examplebucket中destobject对象**

```
PUT /destobject HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:19:21 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:2rZR+iaH8xUewvUKuicLhLHpNoU=
x-obs-copy-source: /bucket/srcobject
```

## 响应示例：普通复制

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 001B21A61C6C00000134031BE8005293
x-obs-id-2: MDAxQjIxQTYxQzZDMDAwMDAxMzQwMzFCRTgwMDUyOTNBQUFBQUFBQWJiYmJiYmJi
```

```
Date: WED, 01 Jul 2015 04:19:21 GMT
Content-Length: 249

<?xml version="1.0" encoding="utf-8"?>
<CopyObjectResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <LastModified>2015-07-01T00:48:07.706Z</LastModified>
  <ETag>"507e3fff69b69bf57d303e807448560b"</ETag>
</CopyObjectResult>
```

### 请求示例：复制多版本对象

**拷贝一个多版本对象，将桶bucket中的版本号为AAABQ4uBLdLc0vycq3gAAAAEVURTRkha的对象srcobject拷贝到桶examplebucket中destobject对象**

```
PUT /destobject HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:20:29 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:4BLYv+1UxfRSHBMvrhVLDszxvcY=
x-obs-copy-source: /bucket/srcobject?versionId=AAABQ4uBLdLc0vycq3gAAAAEVURTRkha
```

### 响应示例：复制多版本对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: DCD2FC9CAB78000001438B8A9C898B79
x-obs-id-2: DB/qBZmbN6AIoX9mrrSNYdLxwvbO0tLR/l6/XKTT4NmZspzharwp5Z74ybAYVOgr
Content-Type: application/xml
x-obs-version-id: AAABQ4uKnOrc0vycq3gAAAAFVURTRkha
x-obs-copy-source-version-id: AAABQ4uBLdLc0vycq3gAAAAEVURTRkha
Date: WED, 01 Jul 2015 04:20:29 GMT
Transfer-Encoding: chunked

<?xml version="1.0" encoding="utf-8"?>
<CopyObjectResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <LastModified>2015-07-01T01:48:07.706Z</LastModified>
  <ETag>"507e3fff69b69bf57d303e807448560b"</ETag>
</CopyObjectResult>
```

## 5.4.4 下载对象

### 功能介绍

GET操作从对象存储下载对象。使用GET接口前，请确认必须拥有对象的READ权限。如果对象Owner向匿名用户授予READ访问权限，则可以在不使用鉴权头域的情况下访问该对象。

### 多版本

默认情况下，获取的是最新版本的对象。如果最新版本的对象是删除标记，则返回对象不存在。如果要获取指定版本的对象，请求可携带versionId消息参数。

### 请求消息样式

```
GET /ObjectName HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
Range:bytes=byte_range
<Optional Additional Header>
```

**说明**

其中Range字段可选，如果没有的话得到全部内容。

## 请求消息参数

GET操作获取对象内容时，允许用户通过请求参数的方式对一些消息头值进行重写，可以重写的消息头有：Content-Type、Content-Language、Expires、Cache-Control、Content-Disposition以及Content-Encoding共6个。另外所需恢复的对象拥有多个版本时，可以通过versionId参数，指定需要下载的版本。具体的说明如表5-74所示。

**说明**

OBS不会处理请求中携带的Accept-Encoding，也不会对上传的数据做任何压缩、解压的操作，压缩解压的操作由客户端决定。某些HTTPClient在默认情况下可能会根据服务端返回的Content-Encoding对数据做相应的解压处理，客户端程序需要根据自己的需求决定是否做解压处理以及如何解压（修改OBS端保存的对象元数据Content-Encoding或者在下载对象时对Content-Encoding进行重写）。如果在下载对象的请求中指明了重写消息头，OBS返回的HTTP标准消息头中将以请求中指定的重写内容为准。

**表 5-74** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| response-content-type | 重写响应中的Content-Type头。<br>类型：String | 否 |
| response-content-language | 重写响应中的Content-Language头。<br>类型：String | 否 |
| response-expires | 重写响应中的Expires头。<br>类型：String | 否 |
| response-cache-control | 重写响应中的Cache-Control头。<br>类型：String | 否 |
| response-content-disposition | 重写响应中的Content-Disposition头。<br>类型：String<br>示例：<br>response-content-disposition=attachment; filename*=utf-8''name1<br>下载对象重命名为“name1”，如果name1中存在中文，需要将中文进行URL编码。 | 否 |
| response-content-encoding | 重写响应中的Content-Encoding头。<br>类型：String | 否 |

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 指定获取对象的版本号。<br>类型：String | 否 |
| attname | 重写响应中的Content-Disposition头。<br>类型：String<br>示例：<br>attname=name1<br>下载对象重命名为“name1”。 | 否 |

## 请求消息头

该请求除使用公共消息头外，还可以使用附加的消息头来完成获取对象的功能，消息头的意义如表5-75所示。

**表 5-75** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Range | 获取对象时，获取在Range范围内的对象内容。如果Range不合法则忽略此字段获取整个对象。<br>Range是一个范围，它的起始值最小为0，最大为对象长度减1。Range范围的起始值为必填项，如果Range只包含起始值，表示获取起始值到对象长度减1这个区间的对象内容。<br>携带Range头域后，响应消息的ETag仍是对象的ETag，而不是Range范围内对象的ETag。<br>类型：String<br>bytes=byte_range<br>示例1：bytes=0-4<br>示例2：bytes=1024<br>示例3：bytes=10-20,30-40（表示多个区间） | 否 |
| If-Modified-Since | 如果对象在请求中指定的时间之后有修改，则返回对象内容；否则的话返回304（not modified）。<br>类型：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| If-Unmodified-Since | 如果对象在请求中指定的时间之后没有修改，则返回对象内容；否则的话返回412（precondition failed）。<br>类型：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。 | 否 |
| If-Match | 如果对象的ETag和请求中指定的ETag相同，则返回对象内容，否则的话返回412（precondition failed）。<br>类型：String<br>（ETag值，例：0f64741bf7cb1089e988e4585d0d3434。） | 否 |
| If-None-Match | 如果对象的ETag和请求中指定的ETag不相同，则返回对象内容，否则的话返回304（not modified）。<br>类型：String<br>（ETag值，例：0f64741bf7cb1089e988e4585d0d3434。） | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: type
Date: date
Content-Length: length
Etag： etag
Last-Modified： time

<Object Content>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如下**表5-76**中的消息头。

**表 5-76** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-expiration | 当对象单独设置了对象lifecycle，过期时间以对象lifecycle为准，该消息头用expiry-date描述对象的详细过期信息；如果对象没有设置对象lifecycle，设置了桶级别lifecycle，过期时间以桶级别lifecycle为准，该消息头用expiry-date和rule-id两个键值对描述对象的详细过期信息；否则不显示该头域。<br>类型：String |
| x-obs-website-redirect-location | 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将请求重定向到该属性指定的桶内的另一个对象或外部的URL。<br>类型：String |
| x-obs-delete-marker | 标识对象是否是删除标记。如果不是，则响应中不会出现该消息头。<br>类型：Boolean<br>有效值：true\|false<br>默认值：false |
| x-obs-version-id | 对象的版本号。如果该对象无版本号，则响应中不会出现该消息头。<br>有效值：字符串<br>默认值：无 |
| x-obs-object-type | 对象为非Normal对象时，会返回此头域，可取值为：Appendable。<br>类型：String |
| x-obs-next-append-position | 对象为Appendable对象时，会返回此头域。<br>类型：Integer |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例：下载整个对象

```
GET /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
```

```
Date: WED, 01 Jul 2015 04:24:33 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:NxtSMS0jaVxlLnxlO9awaMTn47s=
```

## 响应示例：下载整个对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3F2A89604C49ABEE55E
Accept-Ranges: bytes
ETag: "3b46eaf02d3b6b1206078bb86a7b7013"
Last-Modified: WED, 01 Jul 2015 01:20:29 GMT
Content-Type: binary/octet-stream
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSQwxJ2I1VvxD/Xgwuw2G2RQax30gdXU
Date: WED, 01 Jul 2015 04:24:33 GMT
Content-Length: 4572

[4572 Bytes object content]
```

## 请求示例：指定 Range 下载对象

### 指定Range下载对象（下载对象单个区间内容）

```
GET /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Mon, 14 Sep 2020 09:59:04 GMT
Range:bytes=20-30
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:mNPLWQMDWg30PTkAWiqJaLl3ALg=
```

### 指定Range下载对象（下载对象多个区间内容）

```
GET /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Mon, 14 Sep 2020 10:02:43 GMT
Range:bytes=20-30,40-50
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:ZwM7Vk2d7sD9o8zRsRKehgKQDkk=
```

## 响应示例：指定 Range 下载对象

### 指定Range下载对象（下载对象单个区间内容）

```
HTTP/1.1 206 Partial Content
Server: OBS
x-obs-request-id: 000001748C0DBC35802E360C9E869F31
Accept-Ranges: bytes
ETag: "2200446c2082f27ed2a569601ca4e360"
Last-Modified: Mon, 14 Sep 2020 01:16:20 GMT
Content-Range: bytes 20-30/4583
Content-Type: binary/octet-stream
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSn2JHu4okx9NBRNZAvBGawa3lt3g31g
Date: Mon, 14 Sep 2020 09:59:04 GMT
Content-Length: 11

[ 11 Bytes object content]
```

### 指定Range下载对象（下载对象多个区间内容）

```
HTTP/1.1 206 Partial Content
Server: OBS
x-obs-request-id: 8DF400000163D3F2A89604C49ABEE55E
Accept-Ranges: bytes
ETag: "2200446c2082f27ed2a569601ca4e360"
Last-Modified: Mon, 14 Sep 2020 01:16:20 GMT
Content-Type: multipart/byteranges;boundary=35bcf444-e65f-4c76-9430-7e4a68dd3d26
```

```
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSIBWFOVW8eeWujkqSnoIANC2mNR1cdF
Date: Mon, 14 Sep 2020 10:02:43 GMT
Content-Length: 288

--35bcf444-e65f-4c76-9430-7e4a68dd3d26
Content-type: binary/octet-stream
Content-range: bytes 20-30/4583
[ 11 Bytes object content]
--35bcf444-e65f-4c76-9430-7e4a68dd3d26
Content-type: binary/octet-stream
Content-range: bytes 40-50/4583
[ 11 Bytes object content]
--35bcf444-e65f-4c76-9430-7e4a68dd3d26
```

## 请求示例：判断对象 Etag 值

**如果对象Etag值匹配则下载该对象**

```
GET /object01 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:24:33 GMT
If-Match: 682e760adb130c60c120da3e333a8b09
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:NxtSMS0jaVxlLnxlO9awaMTn47s=
```

## 响应示例：判断对象 Etag 值，Etag 不匹配

如果存储的对象的Etag值不是682e760adb130c60c120da3e333a8b09，则提示下载失败

```
HTTP/1.1 412 Precondition Failed
Server: OBS
x-obs-request-id: 8DF400000163D3F2A89604C49ABEE55E
Content-Type: application/xml
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSQwxJ2I1VvxD/Xgwuw2G2RQax30gdXU
Date: WED, 01 Jul 2015 04:20:51 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Error>
  <Code>PreconditionFailed</Code>
  <Message>At least one of the pre-conditions you specified did not hold</Message>
  <RequestId>8DF400000163D3F2A89604C49ABEE55E</RequestId>
  <HostId>ha0ZGaSKVm+uLOrCXXtx4Qn1aLzvoeblctVXRAqA7pty10mzUUW/yOzFue04lBqu</HostId>
  <Condition>If-Match</Condition>
</Error>
```

## 响应示例：判断对象 Etag 值匹配，下载成功

如果存储的对象的Etag值是682e760adb130c60c120da3e333a8b09，则下载成功

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 5DEB00000164A21E1FC826C58F6BA001
Accept-Ranges: bytes
ETag: "682e760adb130c60c120da3e333a8b09"
Last-Modified: Mon, 16 Jul 2015 08:03:34 GMT
Content-Type: application/octet-stream
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSbkdml1sLSvKnoHaRcOwRl+6+ustDwk
Date: Mon, 16 Jul 2015 08:04:00 GMT
Content-Length: 8

[ 8 Bytes object content]
```

## 请求示例：在 URL 中携带签名下载对象

```
GET /object02?
AccessKeyId=H4IPJX0TQTHTHEBQQCEC&Expires=1532688887&Signature=EQmDuOhaLUrzrzRNZxwS72CXeXM%3D HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: Fri, 27 Jul 2018 10:52:31 GMT
```

## 响应示例：在 URL 中携带签名下载对象

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 804F00000164DB5E5B7FB908D3BA8E00
ETag: "682e760adb130c60c120da3e333a8b09"
Last-Modified: Mon, 16 Jul 2015 08:03:34 GMT
Content-Type: application/octet-stream
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCTlpxILjhVK/heKOWIP8Wn2IWmQoerfw
Date: Fri, 27 Jul 2018 10:52:31 GMT
Content-Length: 8

[ 8 Bytes object content]
```

## 请求示例：下载对象并重命名，使用 response-content-disposition 参数

### 下载对象并重命名，使用response-content-disposition参数实现

```
GET /object01?response-content-disposition=attachment; filename*=utf-8''name1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:24:33 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:NxtSMS0jaVxlLnxlO9awaMTn47s=
```

## 响应示例：下载对象并重命名，使用 response-content-disposition 参数

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 804F00000164DB5E5B7FB908D3BA8E00
ETag: "682e760adb130c60c120da3e333a8b09"
Last-Modified: Mon, 16 Jul 2015 08:03:34 GMT
Content-Type: application/octet-stream
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCTlpxILjhVK/heKOWIP8Wn2IWmQoerfw
Date: Fri, 27 Jul 2018 10:52:31 GMT
Content-Length: 8
Content-Disposition: attachment; filename*=utf-8''name1

[ 8 Bytes object content]
```

## 请求示例：下载对象并重命名，使用 attname 参数

### 下载对象并重命名，使用attname参数实现

```
GET /object01?attname=name1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:24:33 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:NxtSMS0jaVxlLnxlO9awaMTn47s=
```

## 响应示例：下载对象并重命名，使用 attname 参数

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 804F00000164DB5E5B7FB908D3BA8E00
```

```
ETag: "682e760adb130c60c120da3e333a8b09"
Last-Modified: Mon, 16 Jul 2015 08:03:34 GMT
Content-Type: application/octet-stream
x-obs-id-2: 32AAAUJAIAABAAAQAAEAABAAAQAAEAABCTlpxILjhVK/heKOWIP8Wn2IWmQoerfw
Date: Fri, 27 Jul 2018 10:52:31 GMT
Content-Length: 8
Content-Disposition: attachment; filename*=utf-8''name1

[ 8 Bytes object content]
```

# 5.4.5 获取对象元数据

## 功能介绍

拥有对象读权限的用户可以执行HEAD操作命令获取对象元数据，返回信息包含对象的元数据信息。

## 多版本

默认情况下，获取的是最新版本的对象元数据。如果最新版本的对象是删除标记，则返回404。如果要获取指定版本的对象元数据，请求可携带versionId消息参数。

## 请求消息样式

```
HEAD /ObjectName HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

请求参数说明如表5-77所示。

**表 5-77** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 对象的版本号。<br>类型：String | 否 |

## 请求消息头

该请求使用公共消息头，具体请参考表3-3。

另外该请求可以使用附加的消息头，具体如表5-78所示。

**表 5-78** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Origin | 预请求指定的跨域请求Origin（通常为域名）。<br>类型：String | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Access-Control-Request-Headers | 实际请求可以带的HTTP头域，可以带多个头域。<br>类型：String | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Content-Type: type
Date: date
Content-Length: length
Etag: etag
Last-Modified: time
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如下**表5-79**中的消息头。

**表 5-79** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-expiration | 当对象单独设置了对象lifecycle，过期时间以对象lifecycle为准，该消息头用expiry-date描述对象的详细过期信息；如果对象没有设置对象lifecycle，设置了桶级别lifecycle，过期时间以桶级别lifecycle为准，该消息头用expiry-date和rule-id两个键值对描述对象的详细过期信息；否则不显示该头域。<br>类型：String |
| x-obs-website-redirect-location | 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将请求重定向到该属性指定的桶内的另一个对象或外部的URL。<br>类型：String |
| x-obs-version-id | 对象的版本号。如果该对象无版本号，则响应中不会出现该消息头。<br>类型：String<br>默认值：无 |

| 消息头名称 | 描述 |
| --- | --- |
| Access-Control-Allow-Origin | 当桶设置了CORS配置，如果请求的Origin满足服务端的CORS配置，则在响应中包含这个Origin。<br>类型：String |
| Access-Control-Allow-Headers | 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。<br>类型：String |
| Access-Control-Max-Age | 当桶设置了CORS配置，服务端CORS配置中的MaxAgeSeconds。<br>类型：Integer |
| Access-Control-Allow-Methods | 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。<br>类型：String<br>有效值：GET、PUT、HEAD、POST 、DELETE |
| Access-Control-Expose-Headers | 当桶设置了CORS配置，服务端CORS配置中的ExposeHeader。<br>类型：String |
| x-obs-object-type | 对象为非Normal对象时，会返回此头域，可取值为：Appendable<br>类型：String |
| x-obs-next-append-position | 对象为Appendable对象时，会返回此头域<br>类型：Integer |
| x-obs-uploadId | 对象为多段上传任务合并而来时，会返回此头域，头域值表示对应的多段任务ID<br>类型：String |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误；所有错误已经包含在表6-2中。

## 请求示例

```
HEAD /object1 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:19:25 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:/cARjk81l2iExMfQqn6iT3qEZ74=
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3E4BB5905C41B6E65B6
Accept-Ranges: bytes
ETag: "3b46eaf02d3b6b1206078bb86a7b7013"
Last-Modified: WED, 01 Jul 2015 01:19:21 GMT
Content-Type: binary/octet-stream
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSD3nAiTaBoeyt9oHp9vTYtXnLDmwV6D
Date: WED, 01 Jul 2015 04:19:21 GMT
Content-Length: 4572
```

## 5.4.6 删除对象

### 功能介绍

删除对象的操作。如果要删除的对象不存在，则仍然返回成功信息。

### 多版本

当桶的多版本状态是开启时，不指定版本删除对象将产生一个带唯一版本号的删除标记，并不删除对象；当桶的多版本状态是Suspended时，不指定版本删除将删除版本号为null的对象，并将产生一个版本号为null的删除标记。

如果要删除指定版本的对象，请求可携带versionId消息参数。

### 请求消息样式

```
DELETE /ObjectName HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

请求参数说明如**表5-80**所示。

**须知**

删除对象时，请求消息参数仅支持**表1 请求消息参数**中列出的参数信息，如果包含了OBS无法识别的参数信息，服务端将返回400错误。

**表 5-80** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 待删除对象的版本号。<br>类型：String | 否 |

### 请求消息头

该请求使用公共消息头，具体请参考**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

除公共响应消息头之外，如果开启了多版本功能，还可能使用如下表5-81中的消息头。

**表 5-81** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-delete-marker | 标识对象是否标记删除。如果不是，则响应中不会出现该消息头。<br>类型：Boolean<br>有效值：true\|false<br>默认值：false |
| x-obs-version-id | 对象的版本号。如果该对象无版本号，则响应中不会出现该消息头。<br>有效值：字符串<br>默认值：无 |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
DELETE /object2 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:19:21 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:MfK9JCnSFHCrJmjv7iRkRrrce2s=
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 8DF400000163D3F51DEA05AC9CA066F1
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCSgkM4Dij80gAeFY8pAZIwx72QhDeBZ5
Date: WED, 01 Jul 2015 04:19:21 GMT
```

# 5.4.7 批量删除对象

## 功能介绍

批量删除对象特性用于将一个桶内的部分对象一次性删除，删除后不可恢复。批量删除对象要求返回结果里包含每个对象的删除结果。OBS的批量删除对象使用同步删除对象的方式，每个对象的删除结果返回给请求用户。

批量删除对象支持两种响应方式：verbose和quiet。Verbose是指在返回响应时，不管对象是否删除成功都将删除结果包含在XML响应里；quiet是指在返回响应时，只返回删除失败的对象结果，没有返回的认为删除成功。OBS默认使用verbose模式，如果用户在请求消息体中指定quiet模式的话， 使用quiet模式。

批量删除的请求消息头中必须包含Content-MD5以及Content-Length，用以保证请求的消息体在服务端检测到网络传输如果有错，则可以检测出来。

## 请求消息样式

```
POST /?delete HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
Content-MD5: MD5
Content-Length: length

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
   <Quiet>true</Quiet>
   <Object>
      <Key>Key</Key>
      <VersionId>VersionId</VersionId>
   </Object>
   <Object>
      <Key>Key</Key>
   </Object>
</Delete>
```

## 请求消息参数

该请求的请求消息中不使用消息参数。

## 请求消息头

该请求使用公共消息头，具体请参考**表3-3**。

## 请求消息元素

该请求通过在请求消息的消息元素中指定要批量删除的对象列表，元素的具体含义如**表5-82**所示。

**表 5-82** 请求消息元素

| 元素名称 | 描述 | 是否必须 |
| --- | --- | --- |
| Quiet | 用于指定使用quiet模式，只返回删除失败的对象结果；如果有此字段，则必被置为True，如果为False则被系统忽略掉。<br>类型：Boolean | 否 |
| Delete | 待删除的对象列表。<br>类型：XML | 是 |
| Object | 待删除的对象。<br>类型：XML | 是 |
| Key | 待删除的对象Key。<br>类型：String | 是 |
| VersionId | 待删除的对象版本号。<br>类型：String | 否 |

批量删除对象一次能接收最大对象数目为1000个，如果超出限制，服务端会返回请求不合法。

并发任务分配后，在循环删除多个对象过程中， 如果发生内部错误， 有可能出现数据不一致的情况（某个对象索引数据删除但还有元数据）。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Type: application/xml
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeleteResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
<Deleted>
   <Key>Key</Key>
</Deleted>
<Error>
   <Key>Key</Key>
   <Code>ErrorCode</Code>
   <Message>Message</Message>
</Error>
</DeleteResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应通过消息元素返回删除的结果，元素的具体意义如**表5-83**中所示。

**表 5-83** 响应消息元素

| 字段名称 | 描述 |
| --- | --- |
| DeleteResult | 批删响应的根节点。<br>类型：Container |
| Deleted | 删除成功结果的Container。<br>类型：Container |
| Error | 删除失败结果的Container。<br>类型：Container |
| Key | 每个删除结果的对象名。<br>类型：String |
| Code | 删除失败结果的错误码。<br>类型：String |
| Message | 删除失败结果的错误消息。<br>类型：String |
| VersionId | 删除对象的版本号<br>类型：String |
| DeleteMarker | 当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，返回消息中该元素的值为true。<br>类型：Boolean |
| DeleteMarkerVersionId | 请求创建或删除的删除标记版本号。<br>当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，响应消息会返回该元素。该元素在以下两种情况中会出现：<br>• 用户发送不带版本删除请求，即请求只有对象名，无版本号。这种情况下，系统会创建一个删除标记，并在响应中返回该删除标记的版本号。<br>• 用户发送带版本删除请求，即请求同时包含对象名以及版本号，但是该版本号标识一个删除标记。这种情况下，系统会删除此删除标记，并在响应中返回该删除标记的版本号。<br>类型：Boolean |

## 错误响应消息

1、用户收到请求后首先进行XML的解析，如果超过1000个对象返回400 Bad Request。

2、如果XML消息体中包含的对象Key不合法，比如超过1024字节，OBS返回400 Bad Request。

3、如果请求消息头中不包含Content-MD5，OBS则返回400 Bad Request。

其他错误已经包含在表6-2中。

## 请求示例

```
POST /test333?delete HTTP/1.1
User-Agent: curl/7.29.0
Host: 127.0.0.1
Accept: */*
Date: WED, 01 Jul 2015 04:34:21 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:8sjZWJlWmYmYnK5JqXaFFQ+vHEg=
Content-MD5: ZPzz8L+hdRJ6qCqYbU/pCw==
Content-Length: 188

<?xml version="1.0" encoding="utf-8"?>
<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>obja02</Key>
  </Object>
  <Object>
    <Key>obja02</Key>
  </Object>
</Delete>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3FE4CE80340D30B0542
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCRhY0FBWRm6qjOE1ACBZwS+0KYlPBq0f
Content-Type: application/xml
Date: WED, 01 Jul 2015 04:34:21 GMT
Content-Length: 120

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeleteResult xmlns="http://obs.example.com/doc/2015-06-30/"/>
```

# 5.4.8 追加写对象

## 功能介绍

追加写对象操作是指在指定桶内的一个对象尾追加上传数据，不存在相同对象键值的对象则创建新对象。

通过Append Object操作创建的Object类型为Appendable Object，而通过Put Object上传的Object是Normal Object。

**说明**

用户上传的对象存储在桶中。用户必须对桶有WRITE权限，才可以在桶中上传对象。同一个桶中存储的对象名必须是唯一的。

为了确保数据在传输过程中没有遭到破坏，用户可以在请求消息头中加入Content-MD5参数，OBS收到上传数据后，会对数据进行MD5校验，如果不一致则返回出错信息。

该操作支持在创建Appendable对象时指定x-obs-acl参数，设置对象的权限控制策略。

## 和其他操作的关系

1. 对一个已经存在的Appendable对象进行Put Object操作，那么该Appendable对象会被新Object覆盖，类型变为Normal对象，反之出错。

2. Appendable对象复制后变成Normal对象，不支持Appenable对象复制成Appendable对象。

## 约束

1. 每次追加上传都会更新该对象的最后修改时间。
2. 每次追加上传的长度不能超过对象长度上限5G的限制。
3. 每个Appendable对象追加写次数最多为10000次。
4. 如果桶设置了跨区域复制配置，则不能调用该接口。
5. 并行文件系统不支持追加写对象。

## 请求消息样式

```
POST /ObjectName?append&position=Position HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Type: application/xml
Content-Length: length
Authorization: authorization
Date: date
<Optional Additional Header>
<object Content>
```

## 请求消息参数

该请求需要在消息中指定参数，表明这是追加写上传，同时指定本次追加上传位置，参数的具体意义如表5-84所示

**表 5-84** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| append | 表明这是以追加写方式上传。<br>类型：String | 是 |
| position | 追加写位置。需要追加写的对象首次上传时就必须指定position为0，下次追加需要填写的position会在服务端本次上传成功返回消息的头域x-obs-next-append-position中携带。<br>类型：Integer | 是 |

## 请求消息头

该请求使用公共消息头，具体请参考表3-3。

请求参数position=0时，该请求可以使用的附加消息头，具体如表5-85所示。

**表 5-85** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-acl | 第一次写时，可以加上此消息头设置对象的权限控制策略，使用的策略为预定义的常用策略，包括：private；public-read；public-read-write。<br>类型：String<br>说明：字符串形式的预定义策略。 | 否 |
| x-obs-grant-read | 第一次写时，使用此头域授权domain下所有用户有读对象和获取对象元数据的权限<br>类型：String | 否 |
| x-obs-grant-read-acp | 第一次写时，使用此头域授权domain下所有用户有获取对象ACL的权限。<br>类型：String | 否 |
| x-obs-grant-write-acp | 第一次写时，使用此头域授权domain下所有用户有写对象ACL的权限。<br>类型：String | 否 |
| x-obs-grant-full-control | 第一次写时，使用此头域授权domain下所有用户有读对象、获取对象元数据、获取对象ACL、写对象ACL的权限。<br>类型：String | 否 |
| x-obs-meta-* | 第一次写时，可以在HTTP请求中加入以“x-obs-meta-”开头的消息头，用来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回消息的头中出现。HTTP请求不包含消息体，长度不能超过8KB。<br>类型：String<br>示例：x-obs-meta-test：test metadata | 否 |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL，OBS将这个值从头域中取出，保存在对象的元数据中。<br>类型：String<br>默认值：无<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2K。 | 否 |
| x-obs-expires | 表示对象的过期时间，单位是天。过期之后对象会被自动删除。（从对象最后修改时间开始计算）<br>类型：Integer<br>示例：x-obs-expires：3 | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
ETag: etag
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

**说明**

ETag返回的是本次追加上传数据的Hash值，不是整个对象的Hash值。

**表 5-86** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 对象的版本号。如果桶的多版本状态为开启，则会返回对象的版本号。<br>类型：String |
| x-obs-next-append-position | 指明下一次请求应该提供的position。<br>类型：Integer |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

1. 如果本次追加上传使对象长度超过对象长度限制，OBS返回400 Bad Request，错误码为AppendTooLarge。
2. 如果position的值和当前对象的原始长度不一致，OBS返回409 Conflict，错误码为PositionNotEqualToLength。
3. 如果指定桶中存在相同对象键值的对象，对象类型非Appendable，OBS返回409 Conflict，错误码为ObjectNotAppendable。
4. 如果对象追加写次数超过10000次，OBS返回409 Conflict，错误码为ObjectNotAppendable。
5. 如果桶设置了跨区域复制配置，则不能调用该接口，否则OBS返回400 Bad Request，错误码为OperationNotSupported。

其他错误已包含在表6-2中。

## 请求示例：普通追加写对象

```
POST /object?append&position=0 HTTP/1.1
Host: examplebucket.obs.region.example.com
```

```
Expires: Wed, 27 Jun 2015 13:45:50 GMT
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1458
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:kZoYNv66bsmc10+dcGKw5x2PRrk=

[1458 bytes of object data]
```

## 响应示例：普通追加写对象

```
HTTP/1.1 200 OK
Date: Wed, 27 Jun 2015 13:45:50 GMT
ETag: "d41d8cd98f00b204e9800998ecf8427e"
Content-Length: 0
Server: OBS
x-obs-request-id: 8DF400000163D3F0FD2A03D2D30B0542
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCTjCqTmsA1XRpIrmrJdvcEWvZyjbztdd
x-obs-next-append-position: 1458
```

## 请求示例：带 redirect 和自定义头域追加写对象

用户存在桶examplebucket，对象obj001不存在，通过追加写接口创建新对象， 设置重定向头域"x-obs-website-redirect-location":"http://www.example.com/"，自定义头域"x-obs-meta-redirect":"redirect"，请求为

```
POST /obj001?append&position=0 HTTP/1.1
Host: examplebucket.obs.region.example.com
Expires: Wed, 27 Jun 2015 13:45:50 GMT
Date: Wed, 08 Jul 2015 06:57:01 GMT
x-obs-website-redirect-location: http://www.example.com/
x-obs-meta-redirect: redirect
Content-Length: 6
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:kZoYNv66bsmc10+dcGKw5x2PRrk=

[6 bytes of object data]
```

## 响应示例：带 redirect 和自定义头域追加写对象

```
HTTP/1.1 200 OK
Date: Wed, 27 Jun 2015 13:45:50 GMT
ETag: "9516dfb15f51c7ee19a4d46b8c0dbe1d"
Content-Length: 0
Server: OBS
x-obs-request-id: 5DEB00000164A3150AC36F8F0C120D50
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCSrVlTYwsA4p9GEW+LYqotSl5BYDxHfT
x-obs-next-append-position: 6
```

# 5.4.9 设置对象 ACL

## 功能介绍

OBS支持对对象的操作进行权限控制。默认情况下，只有对象的创建者才有该对象的读写权限。用户也可以设置其他的访问策略，比如对一个对象可以设置公共访问策略，允许所有人对其都有读权限。

OBS用户在上传对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的对象更改或者获取ACL(access control list) 。一个对象的ACL最多支持100条Grant授权。

本节将介绍如何更改对象ACL，改变对象的访问权限。

## 多版本

默认情况下，更改的是最新版本的对象ACL。要设置指定版本的对象ACL，请求可以带参数versionId。

## 请求消息格式

```
PUT /ObjectName?acl HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization

<AccessControlPolicy>
   <Owner>
      <ID>ID</ID>
   </Owner>
   <Delivered>true</Delivered>
   <AccessControlList>
      <Grant>
         <Grantee>
            <ID>ID</ID>
         </Grantee>
         <Permission>permission</Permission>
      </Grant>
   </AccessControlList>
</AccessControlPolicy>
```

## 请求消息参数

请求参数说明如表5-87所示。

**表 5-87** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 对象的版本号。标示更改指定版本的对象ACL。<br>类型：String | 否 |

## 请求消息头

该请求使用公共请求消息头，具体参见表3-3。

## 请求消息元素

该请求消息通过带消息元素来传递对象的ACL信息，元素的意义如表5-88所示。

**表 5-88** 请求消息元素

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Owner | 桶的所有者信息，包含ID。<br>类型：XML | 是 |
| ID | 用户DomainId。<br>类型：String | 是 |

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| Grant | 用于标记用户及用户的权限。单个对象的ACL，Grant元素不能超过100个。<br>类型：XML | 否 |
| Grantee | 记录用户信息。<br>类型：XML | 否 |
| Canned | 向所有人授予权限。<br>取值范围：Everyone。<br>类型：String | 否 |
| Delivered | 对象ACL是否继承桶的ACL。<br>类型：Boolean<br>默认：true。 | 否 |
| Permission | 授予权限。<br>取值范围：READ \| READ_ACP \| WRITE_ACP \| FULL_CONTROL<br>类型：String | 否 |
| AccessControlList | 访问控制列表，包含Grant、Grantee、Permission三个元素。<br>类型：XML | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Content-Length: length
Content-Type: application/xml
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

除公共响应消息头之外，还可能使用如**表5-89**中的消息头。

**表 5-89** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 被更改ACL的对象的版本号。<br>类型：String |

## 响应消息元素

该请求的响应消息中不带有消息元素。

## 错误响应消息

该请求的响应无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
PUT /obj2?acl HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:42:34 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:8xAODun1ofjkwHm8YhtN0QEcy9M=
Content-Length: 727

<AccessControlPolicy xmlns="http://obs.example.com/doc/2015-06-30/">
  <Owner>
    <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
  </Owner>
  <Delivered>false</Delivered>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <ID>783fc6652cf246c096ea836694f71855</ID>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <Canned>Everyone</Canned>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3F0FD2A03D2D30B0542
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCTjCqTmsA1XRpIrmrJdvcEWvZyjbztdd
Date: WED, 01 Jul 2015 04:42:34 GMT
Content-Length: 0
```

# 5.4.10 获取对象 ACL

## 功能介绍

用户执行获取对象ACL的操作，返回信息包含指定对象的权限控制列表信息。用户必须拥有对指定对象读ACP(access control policy)的权限，才能执行获取对象ACL的操作。

## 多版本

默认情况下，获取最新版本的对象ACL。如果最新版本的对象是删除标记，则返回404。如果要获取指定版本的对象ACL，请求可携带versionId消息参数。

## 请求消息样式

```
GET /ObjectName?acl HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

## 请求消息参数

该请求需要在请求消息参数中指定是在获取对象ACL，参数意义如表5-90所示。

**表 5-90** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 指定对象的版本号。<br>类型：String | 否 |

## 请求消息头

该请求使用公共消息头，具体请参考表3-3。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Owner>
      <ID>id</ID>
   </Owner>
   <Delivered>true</Delivered>
   <AccessControlList>
      <Grant>
         <Grantee>
            <ID>id</ID>
         </Grantee>
         <Permission>permission</Permission>
      </Grant>
   </AccessControlList>
</AccessControlPolicy>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

除公共响应消息头之外，还可能使用如下表5-91中的消息头。

**表 5-91** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 指定对象的版本号。<br>有效值：字符串<br>默认值：无 |

## 响应消息元素

该请求的响应消息中通过消息元素返回对象的ACL信息，元素的具体意义如表5-92所示。

**表 5-92** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ID | 用户的DomainId。<br>类型：String |
| AccessControlList | 访问控制列表，记录了对该桶有访问权限的用户列表。<br>类型：XML |
| Grant | 用于标记用户及用户的权限。<br>类型：XML |
| Grantee | 记录用户信息。<br>类型：XML |
| Delivered | 对象ACL是否继承桶的ACL。<br>类型：Boolean |
| Permission | 指定的用户对该对象所具有的操作权限。<br>类型：String |

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
GET /object011?acl HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:45:55 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:YcmvNQxItGjFeeC1K2HeUEp8MMM=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3E650F3065C2295674C
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCS+wsHqRuA2Tx+mXUpNtBbWLPMle9CIx
Content-Type: application/xml
Date: WED, 01 Jul 2015 04:45:55 GMT
Content-Length: 769

<?xml version="1.0" encoding="utf-8"?>
<AccessControlPolicy xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <Owner>
    <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
  </Owner>
  <Delivered>false</Delivered>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee>
        <ID>783fc6652cf246c096ea836694f71855</ID>
      </Grantee>
      <Permission>READ</Permission>
      </Grant>
    <Grant>
      <Grantee>
        <Canned>Everyone</Canned>
      </Grantee>
      <Permission>READ_ACP</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

# 5.4.11 修改对象元数据

## 功能介绍

用户可以通过本接口添加、修改或删除桶中已经上传的对象的元数据。

## 请求消息样式

```
PUT /ObjectName?metadata HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Type: application/xml
Content-Length: length
Authorization: authorization
Date: date
<Optional Additional Header>
<object Content>
```

## 请求消息参数

**表 5-93** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| versionId | 对象的版本号。<br>类型：String | 否 |

## 请求消息头

**说明**

OBS支持在修改对象元数据的请求里携带HTTP协议规定的6个请求头：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type、Content-Language，OBS会直接将这些头域的值保存在对象元数据中，在下载对象或者HEAD对象的时候，这些保存的值将会被设置到对应的HTTP头域中返回客户端。

**表 5-94** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-metadata-directive | 元数据操作指示符。<br>取值为REPLACE_NEW或REPLACE。<br>REPLACE_NEW表示：对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（备注：自定义头域作替换处理）。<br>REPLACE表示：使用当前请求中携带的头域完整替换，未指定的元数据会被删除。<br>类型：String | 是 |
| Cache-Control | 指定Object被下载时的网页的缓存行为。<br>类型：String | 否 |
| Content-Disposition | 指定Object被下载时的名称。<br>类型：String | 否 |
| Content-Encoding | 指定Object被下载时的内容编码格式。<br>类型：String | 否 |
| Content-Language | 指定Object被下载时的内容语言格式。<br>类型：String | 否 |
| Content-Type | Object文件类型。<br>类型：String | 否 |
| Expires | 指定Object被下载时的网页的缓存过期时间。<br>类型：String | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。<br>例如，重定向请求到桶内另一对象：<br>x-obs-website-redirect-location:/anotherPage.html<br>或重定向请求到一个外部URL：<br>x-obs-website-redirect-location:http://www.example.com/<br>类型：String<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2KB | 否 |
| x-obs-meta-* | 可以在请求中加入以“x-obs-meta-”开头的消息头，用来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回消息的头中出现。<br>类型：String<br>示例：x-obs-meta-test: test metadata | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
Etag: etag
Last-Modified: time
```

## 响应消息头

**表 5-95** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-metadata-directive | 元数据操作指示符。<br>取值为REPLACE_NEW或REPLACE。<br>类型：String |
| Cache-Control | 指定Object被下载时的网页的缓存行为。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |

| 消息头名称 | 描述 |
| --- | --- |
| Content-Disposition | 指定Object被下载时的名称。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |
| Content-Encoding | 指定Object被下载时的内容编码格式。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |
| Content-Language | 指定Object被下载时的内容语言格式。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |
| Expires | 指定Object被下载时的网页的缓存过期时间。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |
| x-obs-meta-* | 用户自定义的元数据，以便对对象进行自定义管理。如果请求携带了此头域，那么响应的消息中应该包含此消息头。<br>类型：String |

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误；所有错误已经包含在**表6-2**中。

## 请求示例：添加对象元数据

给对象object添加元数据：Content-Type:application/zip和x-obs-meta-test:meta。

```
PUT /object?metadata HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 14:24:33 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:NxtSMS0jaVxlLnxlO9awaMTn47s=
x-obs-metadata-directive:REPLACE_NEW
Content-Type:application/zip
x-obs-meta-test:meta
```

## 响应示例：添加对象元数据

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D3E4BB5905C41B6E65B6
```

```
Accept-Ranges: bytes
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSD3nAiTaBoeyt9oHp9vTYtXnLDmwV6D
Date: WED, 01 Jul 2015 04:19:21 GMT
Content-Length: 0
x-obs-metadata-directive:REPLACE_NEW
x-obs-meta-test:meta
```

# 5.4.12 修改写对象

## 功能介绍

修改写对象操作是指将指定文件桶内的一个对象从指定位置起修改为其他内容。

**说明**

目前接口仅在并行文件系统支持，创建并行文件系统的方法详见**请求示例：创建并行文件系统**。

## 请求消息样式

```
PUT /ObjectName?modify&position=Position HTTP/1.1
Host: bucketname.obs.region.example.com
Content-Type: type
Content-Length: length
Authorization: authorization
Date: date
<object Content>
```

## 请求消息参数

该请求需要在消息中指定参数，表明这是修改写上传，同时指定本次修改上传位置。参数说明如**表5-96**所示。

**表 5-96** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| modify | 表明这是以修改写方式上传。<br>类型：String | 是 |
| position | 写操作的位置。<br>类型：Integer | 是 |

## 请求消息头

该请求使用公共的请求消息头，具体如**表3-3**所示。

## 请求消息元素

该请求消息头中不带消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: Date
ETag: etag
Content-Length: length
```

```
Server: OBS
x-obs-request-id: request-id
x-obs-id-2: id
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在**表6-2**中。

## 请求示例

```
PUT /ObjectName?modify&position=Position HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1458
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:kZoYNv66bsmc10+dcGKw5x2PRrk=

[1458 bytes of object data]
```

## 响应示例

```
HTTP/1.1 200
Date: Wed, 08 Jul 2015 06:57:02 GMT
ETag: "d41d8cd98f00b204e9800998ecf8427e"
Content-Length: 0
Server: OBS
x-obs-request-id: 8DF400000163D3F0FD2A03D2D30B0542
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCTjCqTmsA1XRpIrmrJdvcEWvZyjbztd
```

# 5.4.13 截断对象

## 功能介绍

截断对象操作是指将指定文件桶内的一个对象截断到指定大小。

**说明**

目前接口仅在并行文件系统支持，创建并行文件系统的方法详见**请求示例：创建并行文件系统**。

## 请求消息样式

```
PUT /ObjectName?truncate&length=Length HTTP/1.1
Host: bucketname.obs.region.example.com
Authorization: authorization
Content-Length: length
Date: date
```

## 请求消息参数

该请求需要在消息中指定参数，表明这是截断写上传，同时指定本次截断后的对象大小。参数说明如**表5-97**所示。

**表 5-97** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| truncate | 表明这是以截断方式上传。<br>类型：String | 是 |
| length | 截断后对象的大小。<br>类型：Integer | 是 |

## 请求消息头

该请求使用公共的请求消息头，具体如表3-3所示。

## 请求消息元素

该请求消息头中不带消息元素。

## 响应消息样式

```
HTTP/1.1 204 status_code
Server: OBS
x-obs-request-id: request-id
x-obs-id-2: id
Date: Date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

## 请求示例

```
PUT /ObjectName?truncate&length=1000 HTTP/1.1
Host: examplebucket.obs.region.example.com
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:75/Y4Ng1izvzc1nTGxpMXTE6ynw=
Content-Length: 1
Date: WED, 01 Jul 2015 04:19:20 GMT
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 8DF400000163D3F51DEA05AC9CA066F1
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCSgkM4Dij80gAeFY8pAZIwx72QhDeBZ5
Date: WED, 01 Jul 2015 04:19:21 GMT
```

# 5.4.14 重命名对象

## 功能介绍

重命名对象操作是指将指定文件桶内的一个对象重命名为其他对象名。

**说明**

目前接口仅在并行文件系统支持，创建并行文件系统的方法详见**请求示例：创建并行文件系统**。
重命名对象操作为非幂等操作。

## 请求消息样式

```
POST /ObjectName?name=Name&rename HTTP/1.1
Host: bucketname.obs.region.example.com
Authorization: authorization
Date: date
```

## 请求消息参数

该请求需要在消息中指定参数，表明这是重命名操作，同时指定本次重命名后的对象名称。参数说明如**表5-98**所示。

**表 5-98** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| name | 重命名后的对象名称，请使用绝对路径。<br>类型：String | 是 |
| rename | 表明这是重命名操作。<br>类型：String | 是 |

## 请求消息头

该请求使用公共的请求消息头，具体如**表3-3**所示。

## 请求消息元素

该请求消息头中不带消息元素。

## 响应消息样式

```
HTTP/1.1 204 status_code
Server: OBS
x-obs-request-id: request-id
x-obs-id-2: id
Date: Date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

### 响应消息元素

该请求的响应消息中不带消息元素。

### 错误响应消息

无特殊错误，所有错误已经包含在表6-2中。

### 请求示例

```
POST /ObjectName?name=file2&rename HTTP/1.1
Host: examplebucket.obs.region.example.com
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:75/Y4Ng1izvzc1nTGxpMXTE6ynw=
Date: WED, 01 Jul 2015 04:19:20 GMT
```

### 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 8DF400000163D3F51DEA05AC9CA066F1
x-obs-id-2: 32AAAUgAIAABAAAQAAEAABAAAQAAEAABCSgkM4Dij80gAeFY8pAZIwx72QhDeBZ5
Date: WED, 01 Jul 2015 04:19:21 GMT
```

## 5.5 多段操作

## 5.5.1 列举桶中已初始化多段任务

### 功能介绍

用户可以通过本接口查询一个桶中所有的初始化后还未合并以及未取消的多段上传任务。

### 请求消息样式

```
GET /?uploads&max-uploads=max HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: authorization
```

### 请求消息参数

该请求可以通过在请求消息中指定参数，查询指定范围的多段上传任务，请求参数说明如表5-99所示。

**表 5-99** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| delimiter | 对于名字中包含delimiter的对象的任务，其对象名（如果请求中指定了prefix，则此处的对象名需要去掉prefix）中从首字符至第一个delimiter之间的字符串将作为CommonPrefix在响应中返回。对象名包含CommonPrefix的任务被视为一个分组，作为一条记录在响应中返回，该记录不包含任务的信息，仅用于提示用户该分组下存在多段上传任务。<br>类型：String | 否 |
| prefix | 如果请求中指定了prefix，则响应中仅包含对象名以prefix开始的任务信息。<br>类型：String | 否 |
| max-uploads | 列举的多段任务的最大条目，取值范围为[1,1000]，当超出范围时，按照默认的1000进行处理。<br>类型：Integer | 否 |
| key-marker | 列举时返回指定的key-marker之后的多段任务。<br>类型：String | 否 |
| upload-id-marker | 只有和key-marker一起使用才有意义， 列举时返回指定的key-marker的upload-id-marker之后的多段任务。<br>类型：String | 否 |

## 请求消息头

该请求使用公共请求消息头，具体请见表3-3。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListMultipartUploadsResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Bucket>bucketname</Bucket>
   <KeyMarker/>
   <UploadIdMarker/>
   <NextKeyMarker>nextMarker</NextKeyMarker>
   <NextUploadIdMarker>idMarker</NextUploadIdMarker>
   <MaxUploads>maxUploads</MaxUploads>
   <IsTruncated>true</IsTruncated>
   <Upload>
      <Key>key</Key>
      <UploadId>uploadID</UploadId>
```

```
    <Initiator>
      <ID>domainID/domainID:userID/userID</ID>
    </Initiator>
    <Owner>
      <ID>ownerID</ID>
    </Owner>

    <Initiated>initiatedDate</Initiated>
  </Upload>
</ListMultipartUploadsResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中通过消息元素返回多段上传任务，元素的具体意义如**表5-100**所示。

**表 5-100** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| ListMultipartUploadsResult | 保存List Multipart Upload请求结果的容器。<br>类型：Container<br>子节点： Bucket, KeyMarker , UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MaxUploads, Delimiter, Prefix, Upload, CommonPrefixes, IsTruncated。<br>父节点：None。 |
| Bucket | 初始化任务所在的桶名。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |
| KeyMarker | 列举时的起始对象位置。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |
| UploadIdMarker | 列举时的起始UploadId位置。<br>类型：String<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字段，用于标明接下来请求的KeyMarker值。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |

| 元素名称 | 描述 |
| --- | --- |
| NextUploadIdMarker | 如果本次没有返回全部结果，响应请求中将包含NextUploadIdMarker元素，用于标明接下来请求的UploadMarker值。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |
| MaxUploads | 返回的最大多段上传任务数目。<br>类型：Integer<br>父节点: ListMultipartUploadsResult。 |
| IsTruncated | 表明是否本次返回的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：Boolean。<br>父节点: ListMultipartUploadsResult。 |
| Upload | 保存Multipart Upload任务信息的容器。<br>类型：Container<br>子节点: Key, UploadId, InitiatorOwner, Initiated。<br>父节点: ListMultipartUploadsResult。 |
| Key | 初始化Multipart Upload任务的Object名字。<br>类型：String<br>父节点: Upload。 |
| UploadId | Multipart Upload任务的ID。<br>类型：String<br>父节点: Upload。 |
| Initiator | Multipart Upload任务的创建者。<br>子节点: ID。<br>类型：Container<br>父节点: Upload。 |
| ID | 创建者的DomainId。<br>类型：String<br>父节点: Initiator, Owner。 |
| Owner | 段的所有者。<br>类型：Container<br>子节点: ID<br>父节点: Upload。 |
| Initiated | Multipart Upload任务的初始化时间。<br>类型：Date。<br>父节点: Upload。 |

| 元素名称 | 描述 |
| --- | --- |
| ListMultipartUploadsResult.Prefix | 请求中带的Prefix。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |
| Delimiter | 请求中带的Delimiter。<br>类型：String<br>父节点: ListMultipartUploadsResult。 |
| CommonPrefixes | 请求中带Delimiter参数时，返回消息带CommonPrefixes分组信息。<br>类型：Container<br>父节点: ListMultipartUploadsResult。 |
| CommonPrefixes. Prefix | CommonPrefixes分组信息中，表明不同的Prefix。<br>类型：String<br>父节点: CommonPrefixes。 |

## 错误响应消息

OBS系统对maxUploads进行判断，如果maxUploads不为整数类型或者小于0，OBS返回400 Bad Request。

其他错误已经包含在**表6-2**中。

## 请求示例：不带任何参数列举已初始化的段任务

```
GET /?uploads HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:51:21 GMT
Authorization: OBS UDSIAMSTUBTEST000008:XdmZgYQ+ZVy1rjxJ9/KpKq+wrU0=
```

## 响应示例：不带任何参数列举已初始化的段任务

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D405534D046A2295674C
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSDaHP+a+Bp0RI6Mm9XvCOrf7q3qvBQW
Content-Type: application/xml
Date: WED, 01 Jul 2015 04:51:21 GMT
Content-Length: 681

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListMultipartUploadsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Bucket>examplebucket</Bucket>
  <KeyMarker/>
  <UploadIdMarker/>
  <Delimiter/>
  <Prefix/>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <Upload>
    <Key>obj2</Key>
    <UploadId>00000163D40171ED8DF4050919BD02B8</UploadId>
```

```
  <Initiator>
    <ID>domainID/b4bf1b36d9ca43d984fbcb9491b6fce9:userID/71f390117351534r88115ea2c26d1999</ID>
  </Initiator>
  <Owner>
    <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
  </Owner>
  <Initiated>2015-07-01T02:30:54.582Z</Initiated>
 </Upload>
</ListMultipartUploadsResult>
```

### 请求示例：带 prefix 和 delimeter 列举已初始化的段任务

例如，用户桶examplebucket中2个段任务，对象名分别为multipart-object001和part2-key02，列举段任务时，设置prefix为“multipart"，delimeter设置为object001，列举已初始化的段任务。

```
GET /?uploads&delimiter=object001&prefix=multipart HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 04:51:21 GMT
Authorization: OBS UDSIAMSTUBTEST000008:XdmZgYQ+ZVy1rjxJ9/KpKq+wrU0=
```

### 响应示例：带 prefix 和 delimeter 列举已初始化的段任务

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 5DEB00000164A27A1610B8250790D703
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSq3ls2ZtLDD6pQLcJq1yGITXgspSvBR
Content-Type: application/xml
Date: WED, 01 Jul 2015 04:51:21 GMT
Content-Length: 681
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListMultipartUploadsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Bucket>newbucket0001</Bucket>
  <KeyMarker></KeyMarker>
  <UploadIdMarker>
  </UploadIdMarker>
  <Delimiter>object</Delimiter>
  <Prefix>multipart</Prefix>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <CommonPrefixes>
    <Prefix>multipart-object001</Prefix>
  </CommonPrefixes>
</ListMultipartUploadsResult>
```

## 5.5.2 初始化上传段任务

### 功能介绍

使用多段上传特性时，用户必须首先调用创建多段上传任务接口创建任务，系统会给用户返回一个全局唯一的多段上传任务号，作为任务标识。后续用户可以根据这个标识发起相关的请求，如：上传段、合并段、列举段等。创建多段上传任务不影响已有的同名对象；同一个对象可以同时存在多个多段上传任务；每个多段上传任务在初始化时可以附加消息头信息，包括acl、用户自定义元数据和通用的HTTP消息头contentType、contentEncoding等，这些附加的消息头信息将先记录在多段上传任务元数据中。

### 请求消息样式

```
POST /ObjectName?uploads  HTTP/1.1
Host: bucketname.obs.region.example.com
```

```
Date: date
Authorization: authorization
```

## 请求消息参数

该请求需要在消息中指定参数，表明这是多段上传，参数意义如表5-101所示。

**表 5-101** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| uploads | 表明这是多段上传任务。<br>类型：String<br>**说明**<br>• 该参数为空字符串。<br>• 如果请求未设置此参数，则为普通POST上传任务。 | 是 |

## 请求消息头

该请求可以使用附加的消息头，具体如表5-102所示。

**表 5-102** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-acl | 初始化多段上传任务时，可以加上此消息头设置对象的权限控制策略，使用的策略为预定义的常用策略，包括：private；public-read；public-read-write。<br>类型：String<br>说明：字符串形式的预定义策略。<br>示例：x-obs-acl:public-read-write。 | 否 |
| x-obs-grant-read | 初始化多段上传任务时，使用此头域授权domain下所有用户有读对象和获取对象元数据的权限<br>类型：String<br>示例：x-obs-grant-read: ID=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-grant-read-acp | 初始化多段上传任务时，使用此头域授权domain下所有用户有获取对象ACL的权限。<br>类型：String<br>示例：x-obs-grant-read-acp: ID=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-grant-write-acp | 初始化多段上传任务时，使用此头域授权domain下所有用户有写对象ACL的权限。<br>类型：String<br>实例x-obs-grant-write-acp: ID=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-grant-full-control | 初始化多段上传任务时，使用此头域授权domain下所有用户有读对象、获取对象元数据、获取对象ACL、写对象ACL的权限。<br>类型：String<br>示例：x-obs-grant-full-control: ID=domainID。如果授权给多个租户，需要通过“,”分割。 | 否 |
| x-obs-website-redirect-location | 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL，OBS将这个值从头域中取出，保存在对象的元数据中。<br>类型：String<br>默认值：无<br>约束：必须以“/”、“http://”或“https://”开头，长度不超过2K。 | 否 |
| x-obs-expires | 表示对象的过期时间，单位是天。过期之后对象会被自动删除。（从对象最后修改时间开始计算）<br>类型：Integer<br>示例：x-obs-expires:3 | 否 |
| x-obs-meta-* | 初始化上传段任务时，可以在HTTP请求中加入以“x-obs-meta-”开头的消息头，用来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回消息的头中出现。<br>类型：String<br>示例：x-obs-meta-test: test metadata | 否 |

其他公共消息头请参考**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length
Connection: status
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InitiateMultipartUploadResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Bucket>BucketName</Bucket>
   <Key>ObjectName</Key>
   <UploadId>uploadID</UploadId>
</InitiateMultipartUploadResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

该请求响应消息中通过返回消息元素，返回本次多段上传任务的多段上传任务号、桶名、对象名，供后续上传段、合并段使用，元素的具体意义如表5-103所示。

**表 5-103** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| InitiateMultipartUploadResult | 描述多段上传任务的容器。<br>类型：XML |
| Bucket | 多段上传对象所在桶的桶名。<br>类型：String |
| Key | 多段上传对象的key。<br>类型：String |
| UploadId | 多段上传id，后面进行多段上传时，利用这个id指定多段上传任务。<br>类型：String |

## 错误响应消息

1、如果AccessKey或签名无效，OBS返回403 Forbidden，错误码为AccessDenied。

2、如果桶不存在，OBS返回404 Not Found，错误码为NoSuchBucket。

3、检查用户是否具有指定桶的写权限，如果没有权限，OBS返回403 Forbidden，错误码为AccessDenied。

其他错误已包含在表6-2中。

## 请求示例：初始化段

```
POST /objectkey?uploads  HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 05:14:52 GMT
Authorization: OBS AKIAIOSFODNN7EXAMPLE:VGhpcyBtZXNzYWdlIHNpZ25lZGGieSRlbHZpbmc=
```

## 响应示例：初始化段

```
HTTP/1.1 200 OK
Server: OBS
x-obs-id-2: Weag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
```

```
x-obs-request-id: 996c76696e6727732072657175657374
Date: WED, 01 Jul 2015 05:14:52 GMT
Content-Length: 303

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InitiateMultipartUploadResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <Bucket>bucketname</Bucket>
  <Key>objectkey</Key>
  <UploadId>DCD2FC98B4F70000013DF578ACA318E7</UploadId>
</InitiateMultipartUploadResult>
```

### 请求示例：初始化段的同时携带 ACL

```
POST /objectkey?uploads  HTTP/1.1
Host: examplebucket.obs.region.example.com
Date: WED, 01 Jul 2015 05:15:43 GMT
x-obs-grant-write-acp:ID=52f24s3593as5730ea4f722483579ai7,ID=a93fcas852f24s3596ea8366794f7224
Authorization: OBS AKIAIOSFODNN7EXAMPLE:VGhpcyBtZXNzYWdlIHNpZ25lZGGieSRlbHZpbmc=
```

### 响应示例：初始化段的同时携带 ACL

```
HTTP/1.1 200 OK
Server: OBS
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCTnv+daB51p+IVhAvWN7s5rSKhcWqDFs
x-obs-request-id: BB78000001648457112DF37FDFADD7AD
Date: WED, 01 Jul 2015 05:15:43 GMT
Content-Length: 303

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InitiateMultipartUploadResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
  <Bucket>bucketname</Bucket>
  <Key>objectkey</Key>
  <UploadId>000001648453845DBB78F2340DD460D8</UploadId>
</InitiateMultipartUploadResult>
```

## 5.5.3 上传段

### 功能介绍

多段上传任务创建后，用户可以通过指定多段上传任务号，通过上传段接口为特定的任务上传段，从客户端上传新数据。同一个对象的同一个多段上传任务在上传段时，上传的顺序对后续的合并操作没有影响，也即支持多个段并发上传。

段大小范围是[100KB，5GB]，但在进行合并段操作时，最后一个段的大小范围为[0,5GB]。上传的段的编号也有范围限制，其范围是[1,10000]。

**须知**

段任务中的partNumber是唯一的，重复上传相同partNumber的段，后一次上传会覆盖前一次上传内容。多并发上传同一对象的同一partNumber时，服务端遵循Last Write Win策略，但“Last Write”的时间定义为段元数据创建时间。为了保证数据准确性，客户端需要加锁保证同一对象的同一个段上传的并发性。同一对象的不同段并发上传不需要加锁。

### 请求消息样式

```
PUT /ObjectName?partNumber=partNum&uploadId=uploadID  HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Content-Length: length
```

```
Authorization: authorization
Content-MD5：md5
<object Content>
```

## 请求消息参数

在上传段的时候需要通过在消息参数中指定多段上传任务号和段号来上传指定段，参数的具体意义如表5-104所示。

**表 5-104** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| partNumber | 上传段的段号。取值为从1到10000的整数。<br>类型：Integer | 是 |
| uploadId | 多段上传任务Id。<br>类型：String | 是 |

## 请求消息头

该请求使用公共消息头，具体请参考表3-3。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
ETag: etag
Content-Length: length
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息说明

1. 如果段序号超过范围[1,10000]，则返回错误400 Bad Request。
2. 如果段大小超过5G，则返回错误400 Bad Request。
3. 如果AccessKey或签名无效，OBS返回403 Forbidden， 错误码为AccessDenied。
4. 查询桶是否存在，如果桶不存在，OBS返回404 Not Found，错误码为NoSuchBucket。
5. 检查桶的ACL，判断用户DomainId是否具有指定桶的写权限，如果没有权限，则OBS返回403 Forbidden，错误码为AccessDenied。

6. 检查多段上传任务是否存在，如果不存在，OBS返回404 Not Found，错误码为NoSuchUpload。
7. 检查请求用户是否是多段上传任务的发起者（Initiator），如果不是，OBS返回403 Forbidden， 错误码为AccessDenied。

其他错误已包含在表6-2中。

## 请求示例

```
PUT /object02?partNumber=1&uploadId=00000163D40171ED8DF4050919BD02B8 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 05:15:55 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:ZB0hFwaHubi1aKHv7dSZjJts40g=
Content-Length: 102015348

[102015348 Byte part content]
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D40956A703289CA066F1
ETag: "b026324c6904b2a9cb4b88d6d61c81d1"
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCUQu/EOEVSMa04GXVwy0z9WI+BsDKvfh
Date: WED, 01 Jul 2015 05:15:55 GMT
Content-Length: 0
```

# 5.5.4 拷贝段

## 功能介绍

多段上传任务创建后，用户可以通过指定多段上传任务号，为特定的任务上传段。添加段的方式还包括调用段拷贝接口。允许客户将已上传对象的一部分或全部拷贝为段。

**须知**

拷贝段的结果不能仅根据HTTP返回头域中的status_code来判断请求是否成功，头域中status_code返回200时表示服务端已经收到请求，且开始处理拷贝段请求。拷贝是否成功会在响应消息的body中，只有body体中有ETag标签才表示成功，否则表示拷贝失败。

将源对象object拷贝为一个段part1，如果在拷贝操作之前part1已经存在，拷贝操作执行之后老段数据part1会被新拷贝的段数据覆盖。拷贝成功后，只能列举到最新的段part1，老段数据将会被删除。因此在使用拷贝段接口时请确保目标段不存在或者已无价值，避免因拷贝段导致数据误删除。拷贝过程中源对象object无任何变化。

## 请求消息样式

```
PUT /ObjectName?partNumber=partNum&uploadId=UploadID HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
x-obs-copy-source: sourceobject
x-obs-copy-source-range:bytes=start-end
Authorization: authorization
Content-Length: length
```

## 请求消息参数

拷贝段需要在参数中指定目标段的段号和多段上传任务号，参数的具体意义如表5-105所示。

**表 5-105** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| partNumber | 上传段的段号。<br>类型：Integer | 是 |
| uploadId | 多段上传任务Id。<br>类型：String | 是 |

## 请求消息头

该请求的除了使用公共消息头外，还使用了两个扩展的消息头。公共消息头如表3-3所示。

**表 5-106** 请求消息头

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-copy-source | 拷贝的源对象。<br>类型：String | 是 |
| x-obs-copy-source-range | 源对象中待拷贝的段的字节范围（start - end），start为段起始字节，end为段结束字节。<br>类型：Integer | 否 |
| x-obs-copy-source-if-match | 只有当源对象的Etag与此参数指定的值相等时才进行复制对象操作，否则返回412（前置条件不满足）。<br>类型：String<br>示例：x-obs-copy-source-if-match: etag<br>约束条件：此参数可与x-obs-copy-source-if-unmodified-since一起使用，但不能与其它条件复制参数一起使用。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-copy-source-if-none-match | 只有当源对象的Etag与此参数指定的值不相等时才进行复制对象操作，否则返回412（前置条件不满足）。<br>类型：String<br>示例：x-obs-copy-source-if-none-match: etag<br>约束条件：此参数可与x-obs-copy-source-if-modified-since一起使用，但不能与其它条件复制参数一起使用。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-copy-source-if-unmodified-since | 只有当源对象在此参数指定的时间之后没有修改过才进行复制对象操作，否则返回412（前置条件不满足），此参数可与x-obs-copy-source-if-match一起使用，但不能与其它条件复制参数一起使用。<br>类型：String<br>格式：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。<br>1. EEE, dd MMM yyyy HH:mm:ss z<br>2. EEEE, dd-MMM-yy HH:mm:ss z<br>3. EEE MMM dd HH:mm:ss yyyy<br>对应示例：<br>1. x-obs-copy-source-if-unmodified-since: Sun, 06 Nov 1994 08:49:37 GMT<br>2. x-obs-copy-source-if-unmodified-since: Sunday, 06-Nov-94 08:49:37 GMT<br>3. x-obs-copy-source-if-unmodified-since: Sun Nov 6 08:49:37 1994<br>约束条件：此参数指定的时间不能晚于当前的服务器时间（GMT时间），否则参数不生效。 | 否 |

| 消息头名称 | 描述 | 是否必选 |
| --- | --- | --- |
| x-obs-copy-source-if-modified-since | 只有当源对象在此参数指定的时间之后修改过才进行复制对象操作，否则返回412（前置条件不满足），此参数可与x-obs-copy-source-if-none-match一起使用，但不能与其它条件复制参数一起使用。<br>类型：String<br>格式：符合http://www.ietf.org/rfc/rfc2616.txt规定格式的HTTP时间字符串。<br>1. EEE, dd MMM yyyy HH:mm:ss z<br>2. EEEE, dd-MMM-yy HH:mm:ss z<br>3. EEE MMM dd HH:mm:ss yyyy<br>对应示例：<br>1. x-obs-copy-source-if-unmodified-since: Sun, 06 Nov 1994 08:49:37 GMT<br>2. x-obs-copy-source-if-unmodified-since: Sunday, 06-Nov-94 08:49:37 GMT<br>3. x-obs-copy-source-if-unmodified-since: Sun Nov 6 08:49:37 1994<br>约束条件：此参数指定的时间不能晚于当前的服务器时间（GMT时间），否则参数不生效。 | 否 |

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyPartResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
```

```
  <LastModified>modifiedDate</LastModified>
  <ETag>etag</ETag>
</CopyPartResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息使用消息元素来返回段拷贝的结果，元素的意义如**表5-107**所示。

**表 5-107** 响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| LastModified | 对象上次修改时间。<br>类型：String |
| ETag | 目标段的ETag值，是段内容的唯一标识，用于段合并时校验数据一致性。<br>类型：String |

## 错误响应消息

1. 如果AccessKey或签名无效，OBS返回403 Forbidden，错误码为AccessDenied。
2. 查询源桶或目的桶是否存在，如果不存在，OBS返回404 Not Found，错误码为NoSuchBucket。
3. 如果源对象不存在，OBS返回404 Not Found，错误码为NoSuchKey。
4. 如果用户对指定对象没有读权限，OBS返回403 Forbidden，错误码为AccessDenied。
5. 如果用户对目的桶没有写权限，OBS返回403 Forbidden，错误码为AccessDenied。
6. 查询指定的任务不存在，OBS返回404 Not Found，错误码为NoSuchUpload。
7. 如果用户不是多段上传任务的发起者，OBS返回403 Forbidden，错误码为AccessDenied。
8. 当拷贝的单段超过5G时，OBS返回400 Bad Request。
9. 如果段序号超过范围[1,10000]，OBS返回错误400 Bad Request。

其他错误已包含在**表6-2**中。

## 请求示例

```
PUT /tobject02?partNumber=2&uploadId=00000163D40171ED8DF4050919BD02B8 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 05:16:32 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:dSnpnNpawDSsLg/xXxaqFzrAmMw=
x-obs-copy-source: /destbucket/object01
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D40ABBD20405D30B0542
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCTIJpD2efLy5o8sTTComwBb2He0j11Ne
Content-Type: application/xml
Date: WED, 01 Jul 2015 05:16:32 GMT
Transfer-Encoding: chunked

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyPartResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <LastModified>2015-07-01T05:16:32.344Z</LastModified>
  <ETag>"3b46eaf02d3b6b1206078bb86a7b7013"</ETag>
</CopyPartResult>
```

## 5.5.5 列举已上传的段

### 功能介绍

用户可以通过本接口查询一个任务所属的所有段信息。此接口列举的各个段大小和分段上传的各个段大小一致。

### 请求消息样式

```
GET /ObjectName?uploadId=uploadid&max-parts=max&part-number-marker=marker HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: auth
```

### 请求消息参数

该请求通过请求消息参数指定多段上传任务以及列出的段数量，参数的具体含义如表5-108所示。

**表 5-108** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| uploadId | 多段上传任务的id。<br>类型：String<br>默认值：无。 | 是 |
| max-parts | 规定在列举已上传段响应中的最大Part数目。<br>类型：Integer<br>默认值：1,000。 | 否 |
| 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：Integer<br>默认值：无。 | 否 |

### 请求消息头

该请求使用公共消息头，具体请参考表3-3。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListPartsResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Bucket>BucketName</Bucket>
   <Key>object</Key>
   <UploadId>uploadid</UploadId>
   <Initiator>
      <ID>id</ID>
   </Initiator>
   <Owner>
      <ID>ownerid</ID>
   </Owner>
   <PartNumberMarker>partNmebermarker</PartNumberMarker>
   <NextPartNumberMarker>nextPartnumberMarker</NextPartNumberMarker>
   <MaxParts>maxParts</MaxParts>
   <IsTruncated>true</IsTruncated>
   <Part>
      <PartNumber>partNumber</PartNumber>
      <LastModified>modifiedDate</LastModified>
      <ETag>etag</ETag>
      <Size>size</Size>
   </Part>
</ListPartsResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应通过消息元素返回已上传了的段信息，元素的具体含义如**表5-109**所示。

**表 5-109** 响应消息元素

| 响应字段名称 | 描述 |
| --- | --- |
| ListPartsResult | 保存List Part请求结果的容器。<br>类型：Container<br>子节点： Bucket, Key, UploadId, PartNumberMarker,NextPartNumberMarker, MaxParts, IsTruncated, Part。<br>父节点：无。 |
| Bucket | Bucket名称。<br>类型：String<br>父节点：ListPartsResult。 |

| 响应字段名称 | 描述 |
| --- | --- |
| Key | Object名称。<br>类型：String<br>父节点：ListPartsResult。 |
| UploadId | Upload任务ID。<br>类型：String<br>父节点：ListPartsResult。 |
| Initiator | Upload任务的创建者。<br>类型：Container<br>子节点：ID。<br>父节点：ListPartsResult。 |
| Owner | 和Initiator相同。<br>类型：Container<br>子节点: ID。<br>父节点: ListPartsResult。 |
| ID | 创建者的DomainId。<br>类型：String<br>父节点: Initiator、Owner。 |
| PartNumberMarker | 本次List结果的Part Number起始位置。<br>类型：Integer<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：Integer<br>父节点：ListPartsResult。 |
| MaxParts | 返回请求中最大的Part数目。<br>类型：Integer<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：Boolean。<br>父节点：ListPartsResult。 |

| 响应字段名称 | 描述 |
| --- | --- |
| Part | 保存Part信息的容器。<br>类型：String<br>子节点：PartNumber, LastModified, ETag, Size。<br>父节点：ListPartsResult。<br>（PartNumber表示Part的数字。） |
| PartNumber | 已上传Part的编号。<br>类型：Integer<br>父节点：ListPartsResult.Part。 |
| LastModified | Part上传的时间。<br>类型：Date<br>父节点：ListPartsResult.part。 |
| ETag | 已上传段内容的ETag，是段内容的唯一标识，用于段合并时校验数据一致性。<br>类型：String<br>父节点：ListPartsResult.Part |
| Size | 已上传Part大小。<br>类型：Integer<br>父节点：ListPartsResult.Part。 |

## 错误响应消息

1. 如果AccessKey或签名无效，OBS返回403 Forbidden，错误码为AccessDenied。
2. 如果请求的桶不存在，OBS返回404 Not Found，错误码为NoSuchBucket。
3. 如果请求的多段上传任务不存在，OBS返回404 Not Found，错误码为NoSuchUpload。
4. OBS判断用户DomainId是否具有指定桶的读权限，如果没有权限，则OBS返回403 Forbidden，错误码为AccessDenied。

其他错误已经包含在**表6-2**中。

## 请求示例

```
GET /object02?uploadId=00000163D40171ED8DF4050919BD02B8 HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 05:20:35 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:xkABdSrBPrz5yqzuZdJnK5oL/yU=
```

## 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D40C099A04EF4DD1BDD9
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSK71fr+hDnzB0JBvQC1B9+S12AWxC41
```

```
Content-Type: application/xml
Date: WED, 01 Jul 2015 05:20:35 GMT
Content-Length: 888

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListPartsResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Bucket>test333</Bucket>
  <Key>obj2</Key>
  <UploadId>00000163D40171ED8DF4050919BD02B8</UploadId>
  <Initiator>
    <ID>domainID/domainiddomainiddomainiddo000008:userID/useriduseriduseriduseridus000008</ID>
  </Initiator>
  <Owner>
    <ID>domainiddomainiddomainiddo000008</ID>
  </Owner>
  <PartNumberMarker>0</PartNumberMarker>
  <NextPartNumberMarker>2</NextPartNumberMarker>
  <MaxParts>1000</MaxParts>
  <IsTruncated>false</IsTruncated>
  <Part>
    <PartNumber>1</PartNumber>
    <LastModified>2018-06-06T07:39:32.522Z</LastModified>
    <ETag>"b026324c6904b2a9cb4b88d6d61c81d1"</ETag>
    <Size>2058462721</Size>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2018-06-06T07:41:03.344Z</LastModified>
    <ETag>"3b46eaf02d3b6b1206078bb86a7b7013"</ETag>
    <Size>4572</Size>
  </Part>
</ListPartsResult>
```

## 5.5.6 合并段

### 功能介绍

如果用户上传完所有的段，就可以调用合并段接口，系统将在服务端将用户指定的段合并成一个完整的对象。在执行“合并段”操作以前，用户不能下载已经上传的数据。在合并段时需要将多段上传任务初始化时记录的附加消息头信息拷贝到对象元数据中，其处理过程和普通上传对象带这些消息头的处理过程相同。在并发合并段的情况下，仍然遵循Last Write Win策略，但“Last Write”的时间定义为段任务的初始化时间。

已经上传的段，只要没有取消对应的多段上传任务，都要占用用户的容量配额；对应的多段上传任务“合并段”操作完成后，只有指定的多段数据占用容量配额，用户上传的其他此多段任务对应的段数据如果没有包含在“合并段”操作制定的段列表中，“合并段”完成后删除多余的段数据，且同时释放容量配额。

合并完成的多段上传数据可以通过已有的下载对象接口，下载整个多段上传对象或者指定Range下载整个多段上传对象的某部分数据。

合并完成的多段上传数据可以通过已有的删除对象接口，删除整个多段上传对象的所有分段数据，删除后不可恢复。

合并完成的多段上传数据不记录整个对象的MD5作为Etag，在下载多段数据或List桶内对象看到的多段数据其Etag的生成方式为：$MD5（M_1M_2……M_N）-N$，其中，$M_n$表示第n段的MD5值， 如**请求示例**所示，有3个分段，每个分段都有对应的MD5值，合并段ETag的生成是先将3个分段的MD5合并起来再进行MD5计算得到一个新值，再拼接-N作为合并段的ETag值，-N表示总共有多少段，在该示例中即为-3。

合并段请求如果出现等待响应超时、服务端返回500或503报错时，建议客户端可以先尝试获取多段任务对应的对象元数据，并比较响应的x-obs-uploadId头域的值，是否

与本次要合并的段任务ID相同，如果相同，说明服务端实际已经合并段成功，无需再进行重试。关于并发一致性说明，参见**7.5-并发一致性说明**。

## 多版本

如果桶的多版本状态是开启的，则合并段后得到的对象生成一个唯一的版本号，并且会在响应报头x-obs-version-id返回该版本号。如果桶的多版本状态是暂停的，则合并段后得到的对象版本号为**null**。关于桶的多版本状态，参见**设置桶的多版本状态**。

**须知**

如果上传了10个段，但合并时只选择了9个段进行合并，那么未被合并的段将会被系统自动删除，未被合并的段删除后不能恢复。在进行合并之前请使用列出已上传的段接口进行查询，仔细核对所有段，确保没有段被遗漏。

## 请求消息样式

```
POST /ObjectName?uploadId=uploadID HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Content-Length: length
Authorization: authorization
<CompleteMultipartUpload>
   <Part>
      <PartNumber>partNum</PartNumber>
      <ETag>etag</ETag>
   </Part>
   <Part>
      <PartNumber>partNum</PartNumber>
      <ETag>etag</ETag>
   </Part>
   <Part>
      <PartNumber>partNum</PartNumber>
      <ETag>etag</ETag>
   </Part>
</CompleteMultipartUpload>
```

## 请求消息参数

该请求在消息参数中指定多段上传任务号来标明它要合并哪一个上传段任务，参数意义如**表5-110**所示。

**表 5-110** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| uploadId | 指明多段上传任务。<br>类型：String | 是 |

## 请求消息头

该请求使用公共消息头，具体请参考**表3-3**。

## 请求消息元素

该请求需要在消息中带消息元素，指定要合并的段列表，元素的具体意义如表5-111中所示

**表 5-111** 请求消息元素

| 元素名称 | 描述 | 是否必选 |
| --- | --- | --- |
| CompleteMultipartUpload | 合并的段列表。<br>类型：XML | 是 |
| PartNumber | 段号。<br>类型：Integer | 是 |
| ETag | 上传段成功后返回的ETag值，是段内容的唯一标识。该值用于合并段时进行数据一致性校验。<br>类型：String | 是 |

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CompleteMultipartUploadResult xmlns="http://obs.region.example.com/doc/2015-06-30/">
   <Location>http://example-Bucket.obs.region.example.com/example-Object</Location>
   <Bucket>bucketname</Bucket>
   <Key>ObjectName</Key>
   <ETag>ETag</ETag>
</CompleteMultipartUploadResult>
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考表3-19。

除公共响应消息头之外，还可能使用如表5-112中的消息头。

**表 5-112** 附加响应消息头

| 消息头名称 | 描述 |
| --- | --- |
| x-obs-version-id | 合并得到的对象的版本号。<br>类型：String |

## 响应消息元素

该请求的响应消息中通过返回消息元素来返回合并段的结果，元素的具体意义如表5-113所示。

**表 5-113** 响应消息元素

| 元素 | 描述 |
| --- | --- |
| Location | 合并后得到对象的路径。<br>类型：String |
| Bucket | 合并段所在的桶。<br>类型：String |
| Key | 合并得到对象的key。<br>类型：String |
| ETag | 根据各个段的ETag计算得出的结果，是对象内容的唯一标识。<br>类型：String |

## 错误响应消息

1. 如果没有消息体，OBS返回400 Bad Request。
2. 如果消息体格式不正确，OBS返回400 Bad Request。
3. 消息体中如果段信息未按照段序号升序排列，OBS返回400 Bad Request，错误码为InvalidPartOrder。
4. 如果AccessKey或签名无效，OBS返回403 Forbidden， 错误码为AccessDenied。
5. 如果请求的桶不存在，OBS返回404 Not Found，错误码为NoSuchBucket。
6. 如果请求的多段上传任务不存在，OBS返回404 Not Found，包含错误信息NoSuchUpload。
7. 如果用户不是该任务的发起者（initiator），OBS返回403 Forbidden，错误码为AccessDenied。
8. 在合并段时如果请求段列表中包含了不存在的段，OBS返回400 Bad Request，错误码为InvalidPart。
9. 如果请求段列表中包含的段的Etag错误，OBS返回400 Bad Request，错误码为InvalidPart。
10. 除最后一个段之外的其它段尺寸过小（小于100KB），OBS返回400 Bad Request。
11. 对象在合并完成后总大小如果超过48.8TB，OBS返回400 Bad Request。

其他错误已包含在表6-2中。

## 请求示例

```
POST /object02?uploadId=00000163D46218698DF407362295674C HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 05:23:46 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:dOfK9iILcKxo58tRp3fWeDoYzKA=
Content-Length: 422

<?xml version="1.0" encoding="utf-8"?>
<CompleteMultipartUpload>
  <Part>
```

```
  <PartNumber>1</PartNumber>
  <ETag>a54357aff0632cce46d942af68356b38</ETag>
 </Part>
 <Part>
  <PartNumber>2</PartNumber>
  <ETag>0c78aef83f66abc1fa1e8477f296d394</ETag>
 </Part>
 <Part>
  <PartNumber>3</PartNumber>
  <ETag>acbd18db4cc2f85cedef654fccc4a4d8</ETag>
 </Part>
</CompleteMultipartUpload>
```

### 响应示例

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: 8DF400000163D4625BE3075019BD02B8
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSN8D1AfQcIvyGBZ9+Ee+jU6zv1iYdO4
Content-Type: application/xml
Date: WED, 01 Jul 2015 05:23:46 GMT
Content-Length: 326

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CompleteMultipartUploadResult xmlns="http://obs.example.com/doc/2015-06-30/">
  <Location>/examplebucket/object02</Location>
  <Bucket>examplebucket</Bucket>
  <Key>object02</Key>
  <ETag>"03f814825e5a691489b947a2e120b2d3-3"</ETag>
</CompleteMultipartUploadResult>
```

## 5.5.7 取消多段上传任务

### 功能介绍

如果用户希望取消一个任务，可以调用取消多段上传任务接口取消任务。合并段或取消任务接口被调用后，用户不能再对任务进行上传段和列举段的操作。

### 请求消息样式

```
DELETE /ObjectName?uploadId=uplaodID HTTP/1.1
Host: bucketname.obs.region.example.com
Date: date
Authorization: auth
```

### 请求消息参数

该请求通过消息参数，指定要取消的段任务的多段上传任务号，参数的意义如表5-114所示。

**表 5-114** 请求消息参数

| 参数名称 | 描述 | 是否必选 |
| --- | --- | --- |
| uploadId | 指明多段上传任务。<br>类型：String | 是 |

## 请求消息头

该请求使用公共消息头，具体请参考**表3-3**。

## 请求消息元素

该请求消息中不使用消息元素。

## 响应消息样式

```
HTTP/1.1 status_code
Date: date
```

## 响应消息头

该请求的响应消息使用公共消息头，具体请参考**表3-19**。

## 响应消息元素

该请求的响应消息中不带消息元素。

## 错误响应消息

1. 如果AccessKey或签名无效，OBS返回403 Forbidden，错误码为AccessDenied。
2. 如果请求的桶不存在，OBS返回404 Not Found，错误码为NoSuchBucket。
3. 用户执行取消多段上传任务操作时判断用户是否是任务初始化者或是桶的所有者，如果不是则OBS则返回403 Forbidden。
4. 操作成功，OBS向用户返回204 No Content。

其他错误已包含在**表6-2**中。

## 请求示例

```
DELETE /object02?uploadId=00000163D46218698DF407362295674C HTTP/1.1
User-Agent: curl/7.29.0
Host: examplebucket.obs.region.example.com
Accept: */*
Date: WED, 01 Jul 2015 05:28:27 GMT
Authorization: OBS H4IPJX0TQTHTHEBQQCEC:QmM2d1DBXZ/b8drqtEv1QJHPbM0=
```

## 响应示例

```
HTTP/1.1 204 No Content
Server: OBS
x-obs-request-id: 8DF400000163D463E02A07EC2295674C
x-obs-id-2: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCTp5YDlzn0UgqG3laRfkHLGyz7RpR9ON
Date: WED, 01 Jul 2015 05:28:27 GMT
```

# 6 错误码

调用接口出错后，将不会返回结果数据。调用方可根据每个接口对应的错误码来定位错误原因。 当调用出错时，HTTP请求返回一个3xx，4xx或5xx的HTTP状态码。返回的消息体中是具体的错误代码及错误信息。

## 错误响应消息格式

当错误发生时，响应消息头中都会包含：

- Content-Type: application/xml
- 错误对应的3xx，4xx或5xx的HTTP状态码。

响应消息体中同样会包含对错误的描述信息。下面的错误响应消息体示例展示了所有REST错误响应中公共的元素。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
<Code>NoSuchKey</Code>
<Message>The resource you requested does not exist</Message>
<Resource>/example-bucket/object</Resource>
<RequestId>001B21A61C6C0000013402C4616D5285</RequestId>
<HostId>RkRCRDJENDc5MzdGQkQ4OUY3MTI4NTQ3NDk2Mjg0M0FB
QUFBQUFBYmJiYmJiYmJD</HostId>
</Error>
```

各元素的具体含义如表6-1所示。

**表 6-1** 错误响应消息元素

| 元素名称 | 描述 |
| --- | --- |
| Error | 错误响应消息体XML结构中描述错误信息的根节点元素 。 |
| Code | 错误响应消息体XML中错误响应对应的HTTP消息返回码，具体的错误码请参见表6-2。 |
| Message | 错误响应消息体XML中具体错误更全面、详细的英文解释，具体的错误消息请参见表6-2。 |
| RequestId | 本次错误请求的请求ID，用于错误定位。 |

| 元素名称 | 描述 |
| --- | --- |
| HostId | 返回该消息的服务端ID。 |
| Resource | 该错误相关的桶或对象资源。 |

**说明**

许多错误响应包含其他的更丰富的错误信息， 建议将所有错误信息记入日志，方便程序员在诊断程序错误时阅读和理解。

## 错误码说明

在向OBS系统发出请求后，如果遇到错误，会在响应中包含响应的错误码描述错误信息。对象存储访问服务的错误码如表6-2所示。

**表 6-2** 错误码

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 301 Moved Permanently | PermanentRedirect | 尝试访问的桶必须使用指定的地址，请将以后的请求发送到这个地址。 | 按照返回的重定向地址发送请求。 |
| 301 Moved Permanently | WebsiteRedirect | Website请求缺少bucketName。 | 携带桶名后重试。 |
| 307 Moved Temporarily | TemporaryRedirect | 临时重定向，当DNS更新时，请求将被重定向到桶。 | 会自动重定向，也可以将请求发送到重定向地址。 |
| 400 Bad Request | BadDigest | 客户端指定的对象内容的MD5值与系统接收到的内容MD5值不一致。 | 检查头域中携带的MD5与消息体计算出来的MD5是否一致。 |
| 400 Bad Request | BadDomainName | 域名不合法。 | 使用合法的域名。 |
| 400 Bad Request | BadRequest | 请求参数不合法。 | 根据返回的错误消息体提示进行修改。 |
| 400 Bad Request | CustomDomainAreadyExist | 配置了已存在的域。 | 已经配置过了，不需要再配置。 |
| 400 Bad Request | CustomDomainNotExist | 删除不存在的域。 | 未配置或已经删除，无需删除。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 400 Bad Request | EntityTooLarge | • 通过SDK或API的PUT上传、POST上传和追加写超过5GB的文件<br>• 上传段超过5GB<br>• 设置的桶配置超过20KB<br>• 超过post表单中policy限制的文件大小（用户可自定义）<br>• 通过SDK或API的多段上传以及SDK的断点续传超过48.8TB的文件 | 修改上传的policy中的条件或者减少对象大小。 |
| 400 Bad Request | EntityTooSmall | • 上传段除最后一段小于100KB<br>• 小于post表单中policy限制的文件大小（用户可自定义） | 修改上传的policy中的条件或者增加对象大小。 |
| 400 Bad Request | IllegalLocationConstraintException | 用户未带Location在非默认Region创桶。 | 请求发往默认Region创桶或带非默认Region的Location创桶。 |
| 400 Bad Request | IncompleteBody | 由于网络原因或其他问题导致请求体未接受完整。 | 重新上传对象。 |
| 400 Bad Request | IncorrectNumberOfFilesInPost Request | 每个POST请求都需要带一个上传的文件。 | 带上一个上传文件。 |
| 400 Bad Request | InvalidArgument | 无效的参数。 | 根据返回的错误消息体提示进行修改。 |
| 400 Bad Request | InvalidBucket | 请求访问的桶已不存在。 | 更换桶名。 |
| 400 Bad Request | InvalidBucketName | 请求中指定的桶名无效，超长或带不允许的特殊字符。 | 更换桶名。 |
| 400 Bad Request | InvalidContentLength | Content-Length头域内容有误。 | 请检查封装头域或联系技术支持。 |
| 400 Bad Request | InvalidLocationConstraint | 创建桶时，指定的Location不合法或不存在。 | 指定正确的Location创桶。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 400 Bad Request | InvalidPart | 一个或多个指定的段无法找到。这些段可能没有上传，或者指定的entity tag与段的entity tag不一致。 | 按照正确的段和entity tag合并段。 |
| 400 Bad Request | InvalidPartOrder | 段列表的顺序不是升序，段列表必须按段号升序排列。 | 按段号升序排列后重新合并。 |
| 400 Bad Request | InvalidPolicyDocument | 表单中的内容与策略文档中指定的条件不一致。 | 根据返回的错误消息体提示修改构造表单的policy重试。 |
| 400 Bad Request | InvalidRedirectLocation | 无效的重定向地址。 | 指定正确的地址。 |
| 400 Bad Request | InvalidRequest | 无效请求。 | 根据返回的错误消息体提示进行修改。 |
| 400 Bad Request | InvalidRequestBody | 请求体无效，需要消息体的请求没有上传消息体。 | 按照正确的格式上传消息体。 |
| 400 Bad Request | InvalidTargetBucketForLogging | delivery group对目标桶无ACL权限。 | 对目标桶配置ACL权限后重试。 |
| 400 Bad Request | KeyTooLongError | 提供的Key过长。 | 使用较短的Key。 |
| 400 Bad Request | MalformedACLError | 提供的XML格式错误，或者不符合要求的格式。 | 使用正确的XML格式重试。 |
| 400 Bad Request | MalformedError | 请求中携带的XML格式不正确。 | 使用正确的XML格式重试。 |
| 400 Bad Request | MalformedLoggingStatus | Logging的XML格式不正确。 | 使用正确的XML格式重试。 |
| 400 Bad Request | MalformedPolicy | Bucket policy检查不通过。 | 根据返回的错误消息体提示结合桶policy的要求进行修改。 |
| 400 Bad Request | MalformedQuotaError | Quota的XML格式不正确。 | 使用正确的XML格式重试。 |
| 400 Bad Request | MalformedXML | 当用户发送了一个配置项的错误格式的XML会出现这样的错误。 | 使用正确的XML格式重试。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 400 Bad Request | MaxMessageLengthExceeded | 拷贝对象，带请求消息体。 | 拷贝对象不带消息体重试。 |
| 400 Bad Request | MetadataTooLarge | 元数据消息头超过了允许的最大元数据大小。 | 减少元数据消息头。 |
| 400 Bad Request | MissingRegion | 请求中缺少Region信息，且系统无默认Region。 | 请求中携带Region信息。 |
| 400 Bad Request | MissingRequestBodyError | 当用户发送一个空的XML文档作为请求时会发生。 | 提供正确的XML文档。 |
| 400 Bad Request | MissingRequiredHeader | 请求中缺少必要的头域。 | 提供必要的头域。 |
| 400 Bad Request | MissingSecurityHeader | 请求缺少一个必须的头。 | 提供必要的头域。 |
| 400 Bad Request | MultipleContentLengths | 多个Content-Length头域。 | 请检查封装头域或者联系技术支持。 |
| 400 Bad Request | TooManyBuckets | 用户拥有的桶的数量达到了系统的上限，并且请求试图创建一个新桶。 | 删除部分桶后重试。 |
| 400 Bad Request | TooManyCustomDomains | 配置了过多的用户域。 | 删除部分用户域后重试。 |
| 400 Bad Request | TooManyWrongSignature | 因高频错误请求被拒绝服务。 | 更换正确的Access Key后重试。 |
| 400 Bad Request | UnexpectedContent | 该请求需要消息体而客户端没带，或该请求不需要消息体而客户端带了。 | 根据说明重试。 |
| 400 Bad Request | UserKeyMustBeSpecified | 该操作只有特殊用户可使用。 | 请联系技术支持。 |
| 403 Forbidden | AccessDenied | 拒绝访问，请求没有携带日期头域或者头域格式错误。 | 请求携带正确的日期头域。 |
| 403 Forbidden | AccessForbidden | 权限不足，桶未配置CORS或者CORS规则不匹配。 | 修改桶的CORS配置，或者根据桶的CORS配置发送匹配的OPTIONS请求。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 403 Forbidden | AllAccessDisabled | 用户无权限执行某操作。桶名为禁用关键字。 | 更换桶名。 |
| 403 Forbidden | InsufficientStorageSpace | 存储空间不足。 | 超过配额限制，增加配额或删除部分对象。 |
| 403 Forbidden | InvalidAccessKeyId | 系统记录中不存在客户提供的Access Key Id。 | 携带正确的Access Key Id。 |
| 403 Forbidden | RequestTimeTooSkewed | 客户端发起请求的时间与OBS服务端的时间相差太大。<br>出于安全目的，OBS会校验客户端与OBS服务端的时间差，当该时间差大于15分钟时，OBS服务端会拒绝您的请求，从而出现此报错。 | 请检查客户端时间是否与当前OBS服务端时间相差太大。请根据本地UTC时间调整客户端时间后再访问。 |
| 403 Forbidden | SignatureDoesNotMatch | 请求中带的签名与系统计算得到的签名不一致。 | 检查你的Secret Access Key和签名计算方法。 |
| 403 Forbidden | VirtualHostDomainRequired | 未使用虚拟主机访问域名。 | Host使用**虚拟主机访问域名**。 |
| 403 Forbidden | Unauthorized | 用户未实名认证。 | 请实名认证后重试。 |
| 404 Not Found | NoSuchBucket | 指定的桶不存在。 | 先创桶再操作。 |
| 404 Not Found | NoSuchBucketPolicy | 桶policy不存在。 | 先配置桶policy。 |
| 404 Not Found | NoSuchCORSConfiguration | CORS配置不存在。 | 先配置CORS。 |
| 404 Not Found | NoSuchCustomDomain | 请求的用户域不存在。 | 先设置用户域。 |
| 404 Not Found | NoSuchKey | 指定的Key不存在。 | 先上传对象。 |
| 404 Not Found | NoSuchLifecycleConfiguration | 请求的LifeCycle不存在。 | 先配置LifeCycle。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 404 Not Found | NoSuchUpload | 指定的多段上传不存在。Upload ID不存在，或者多段上传已经终止或完成。 | 使用存在的段或重新初始化段。 |
| 404 Not Found | NoSuchVersion | 请求中指定的version ID与现存的所有版本都不匹配。 | 使用正确的version ID。 |
| 404 Not Found | NoSuchWebsiteConfiguration | 请求的Website不存在。 | 先配置Website。 |
| 405 Method Not Allowed | MethodNotAllowed | 指定的方法不允许操作在请求的资源上。<br>对应返回的Message为：Specified method is not supported. | 方法不允许。 |
| 405 Method Not Allowed | FsNotSupport | posix桶不支持该API。 | 方法不允许。 |
| 408 Request Timeout | RequestTimeout | 用户与Server之间的socket连接在超时时间内没有进行读写操作。 | 检查网络后重试，或联系技术支持。 |
| 409 Conflict | BucketAlreadyExists | 请求的桶名已经存在。桶的命名空间是系统中所有用户共用的，选择一个不同的桶名再重试一次。 | 更换桶名。 |
| 409 Conflict | BucketAlreadyOwnedByYou | 发起该请求的用户已经创建过了这个名字的桶，并拥有这个桶。 | 不需要再创桶了。 |
| 409 Conflict | BucketNotEmpty | 用户尝试删除的桶不为空。 | 先删除桶中对象，然后再删桶。 |
| 409 Conflict | InvalidBucketState | 无效的桶状态，配置跨Region复制后不允许关闭桶多版本。 | 不关闭桶的多版本或取消跨Region复制。 |
| 409 Conflict | OperationAborted | 另外一个冲突的操作当前正作用在这个资源上，请重试。 | 等待一段时间后重试。 |
| 409 Conflict | ServiceNotSupported | 请求的方法服务端不支持。 | 服务端不支持，请联系技术支持。 |

| 状态码 | 错误码 | 错误信息 | 处理措施 |
| --- | --- | --- | --- |
| 409 ObjectNotAppendable | ObjectNotAppendable | 该对象不支持追加上传 | 请确认桶类型，并行文件系统不支持追加上传。 |
| 411 Length Required | MissingContentLength | 必须要提供HTTP消息头中的Content-Length字段。 | 提供Content-Length消息头。 |
| 412 Precondition Failed | PreconditionFailed | 用户指定的先决条件中至少有一项没有包含。 | 根据返回消息体中的Condition提示进行修改。 |
| 414 URI Too Long | Request-URI Too Large | 请求使用的URI过长 | 请减少URI的长度。 |
| 416 Client Requested Range Not Satisfiable | InvalidRange | 请求的range不可获得。 | 携带正确的range重试。 |
| 500 Internal Server Error | InternalError | 系统遇到内部错误，请重试。 | 请联系技术支持。 |
| 501 Not Implemented | ServiceNotImplemented | 请求的方法服务端没有实现。 | 当前不支持，请联系技术支持。 |
| 503 Service Unavailable | ServiceUnavailable | 服务器过载或者内部错误异常。 | 等待一段时间后重试，或联系技术支持。 |
| 503 Service Unavailable | SlowDown | 请降低请求频率。 | 请降低请求频率。 |

# 7 IAM 策略和授权项

## 7.1 IAM 策略及授权项说明

如果您需要对您所拥有的对象存储服务（OBS）进行精细的权限管理，您可以使用统一身份认证服务（Identity and Access Management，简称IAM），如果当前的账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用OBS的其它功能。

默认情况下，新建的IAM用户没有任何权限，您需要将其加入用户组，并给用户组授予IAM策略，才能使用户组中的用户获得IAM策略定义的权限，这一过程称为授权。授权后，用户就可以基于IAM策略对云服务进行操作。

关于IAM中和OBS相关的用户策略介绍，请参见《对象存储服务用户指南》的“产品介绍 > 权限管理”章节。关于IAM策略的语法结构及示例，请参见《对象存储服务用户指南》的“控制台指南 > 权限控制 > 权限控制方式介绍 > IAM策略”章节。

策略根据授权精度分为细粒度策略和RBAC策略。RBAC策略是将服务作为一个整体进行授权，授权后，用户可以拥有这个服务的所有权限。细粒度策略以API接口为粒度进行权限拆分，授权更加精细，可以精确到某个操作。

**说明**

- 如果您要允许或是禁止某个接口的操作权限，请使用细粒度策略。
- 由于缓存的存在，对用户、用户组授予OBS相关的RBAC策略后，大概需要等待10~15分钟RBAC策略才能生效；授予OBS相关的细粒度策略后，大概需要等待5分钟细粒度策略才能生效。

账号具备所有接口的调用权限，如果使用账号下的IAM用户发起API请求时，该IAM用户必须具备调用该接口所需的权限，否则，API请求将调用失败。每个接口所需要的权限，与各个接口所对应的授权项相对应，只有发起请求的用户被授予授权项所对应的策略，该用户才能成功调用该接口。例如，用户要调用接口来创建桶，那么这个IAM用户被授予的策略中必须包含允许“obs:bucket:CreateBucket”的授权项，该接口才能调用成功。

### 支持的授权项

细粒度策略支持的操作与API相对应，授权项列表说明如下：

- 权限：自定义策略中授权项定义的内容即为权限。
- 对应API接口：自定义策略实际调用的API接口。
- 授权项：自定义策略中支持的Action，在自定义策略中的Action中写入授权项，可以实现授权项对应的权限功能。

OBS支持的自定义策略授权项如下所示：

- **桶相关授权项**：包括OBS所有面向桶的接口所对应的授权项，如列举全部桶、创建桶、删除桶、设置桶策略、设置桶的日志记录、设置桶的事件通知、设置桶的跨区域复制配置等接口。
- **对象相关授权项**：包括上传对象、下载对象、删除对象等接口。

# 7.2 桶相关授权项

**表 7-1** 桶相关授权项列表

| 权限 | 对应API接口 | 授权项（Action） | IAM项目（Project） |
| --- | --- | --- | --- |
| 列举全部桶 | **获取桶列表** | obs:bucket:ListAllMyBuckets | √ |
| 创建桶 | **创建桶** | obs:bucket:CreateBucket | √ |
| 列举桶内对象 | **列举桶内对象** | obs:bucket:ListBucket | √ |
| 列举桶内多版本对象 | **列举桶内对象** | obs:bucket:ListBucketVersions | √ |
| 判断桶是否存在并获取桶的元数据 | **获取桶元数据** | obs:bucket:HeadBucket | √ |
| 获取桶位置 | **获取桶区域位置** | obs:bucket:GetBucketLocation | √ |
| 删除桶 | **删除桶** | obs:bucket:DeleteBucket | √ |
| 设置桶策略 | **设置桶策略** | obs:bucket:PutBucketPolicy | √ |
| 获取桶策略配置的相关信息 | **获取桶策略** | obs:bucket:GetBucketPolicy | √ |
| 删除桶策略 | **删除桶策略** | obs:bucket:DeleteBucketPolicy | √ |
| 设置桶ACL | **设置桶ACL** | obs:bucket:PutBucketAcl | √ |
| 获取桶ACL的相关信息 | **获取桶ACL** | obs:bucket:GetBucketAcl | √ |

| 权限 | 对应API接口 | 授权项（Action） | IAM项目（Project） |
| --- | --- | --- | --- |
| 设置桶日志记录 | 设置桶日志管理配置 | obs:bucket:PutBucketLogging | √ |
| 获取桶日志记录的相关信息 | 获取桶日志管理配置 | obs:bucket:GetBucketLogging | √ |
| 设置和删除桶生命周期规则 | 设置桶的生命周期配置<br>删除桶的生命周期配置 | obs:bucket:PutLifecycleConfiguration | √ |
| 获取桶生命周期规则 | 获取桶的生命周期配置 | obs:bucket:GetLifecycleConfiguration | √ |
| 设置多版本 | 设置桶的多版本状态 | obs:bucket:PutBucketVersioning | √ |
| 获取桶多版本的相关信息 | 获取桶的多版本状态 | obs:bucket:GetBucketVersioning | √ |
| 设置桶的事件通知 | 设置桶的消息通知配置 | obs:bucket:PutBucketNotification | √ |
| 获取桶的事件通知 | 获取桶的消息通知配置 | obs:bucket:GetBucketNotification | √ |
| 设置桶的跨区域复制配置 | 设置桶的跨区域复制配置 | obs:bucket:PutReplicationConfiguration | √ |
| 获取桶的跨区域复制配置 | 获取桶的跨区域复制配置 | obs:bucket:GetReplicationConfiguration | √ |
| 删除桶的跨区域复制配置 | 删除桶的跨区域复制配置 | obs:bucket:DeleteReplicationConfiguration | √ |
| 设置桶标签 | 设置桶标签 | obs:bucket:PutBucketTagging | √ |
| 获取桶标签 | 获取桶标签 | obs:bucket:GetBucketTagging | √ |
| 删除桶标签 | 删除桶标签 | obs:bucket:DeleteBucketTagging | √ |
| 设置桶配额 | 设置桶配额 | obs:bucket:PutBucketQuota | √ |
| 获取桶配额 | 获取桶配额 | obs:bucket:GetBucketQuota | √ |
| 获取桶存量信息 | 获取桶存量信息 | obs:bucket:GetBucketStorage | √ |

| 权限 | 对应API接口 | 授权项（Action） | IAM项目（Project） |
| --- | --- | --- | --- |
| 设置桶清单 | 设置桶清单 | obs:bucket:PutBucketInventoryConfiguration | √ |
| 获取和列举桶清单 | 获取桶清单<br>列举桶清单 | obs:bucket:GetBucketInventoryConfiguration | √ |
| 删除桶清单 | 删除桶清单 | obs:bucket:DeleteBucketInventoryConfiguration | √ |
| 设置桶的自定义域名 | 设置桶的自定义域名 | obs:bucket:PutBucketCustomDomainConfiguration | √ |
| 获取桶的自定义域名 | 获取桶的自定义域名 | obs:bucket:GetBucketCustomDomainConfiguration | √ |
| 删除桶的自定义域名 | 删除桶的自定义域名 | obs:bucket:DeleteBucketCustomDomainConfiguration | √ |
| 设置桶的静态网站托管 | 设置桶的网站配置 | obs:bucket:PutBucketWebsite | √ |
| 获取桶的静态网站配置的相关信息 | 获取桶的网站配置 | obs:bucket:GetBucketWebsite | √ |
| 删除桶的静态网站托管配置 | 删除桶的网站配置 | obs:bucket:DeleteBucketWebsite | √ |
| 设置和删除桶CORS | 设置桶的CORS配置<br>删除桶的CORS配置 | obs:bucket:PutBucketCORS | √ |
| 获取桶CORS配置的相关信息 | 获取桶的CORS配置 | obs:bucket:GetBucketCORS | √ |
| 列举桶中已初始化多段任务 | 列举桶中已初始化多段任务 | obs:bucket:ListBucketMultipartUploads | √ |

# 7.3 对象相关授权项

**表 7-2** 对象相关授权项列表

| 权限 | 对应API接口 | 授权项（Action） | IAM项目（Project） |
| --- | --- | --- | --- |
| 可用作于PUT上传对象，POST上传对象，复制对象，追加写对象，初始化上传段任务，上传段，拷贝段，合并段 | **PUT上传**<br>**POST上传**<br>**复制对象**<br>**追加写对象**<br>**初始化上传段任务**<br>**上传段**<br>**合并段** | obs:object:PutObject | √ |
| 获取对象内容和对象元数据 | **下载对象**<br>**获取对象元数据** | obs:object:GetObject | √ |
| 获取指定版本对象内容和对象元数据 | **下载对象**<br>**获取对象元数据** | obs:object:GetObjectVersion | √ |
| 单个删除和批量删除对象 | **删除对象**<br>**批量删除对象** | obs:object:DeleteObject | √ |
| 单个删除和批量删除指定版本对象 | **删除对象**<br>**批量删除对象** | obs:object:DeleteObjectVersion | √ |
| 设置对象ACL | **设置对象ACL** | obs:object:PutObjectAcl | √ |
| 设置指定版本对象ACL | **设置对象ACL** | obs:object:PutObjectVersionAcl | √ |
| 获取对象ACL的相关信息 | **获取对象ACL** | obs:object:GetObjectAcl | √ |
| 获取指定版本对象ACL的相关信息 | **获取对象ACL** | obs:object:GetObjectVersionAcl | √ |
| 修改对象元数据 | **修改对象元数据** | obs:object:ModifyObjectMetaData | √ |
| 列举已上传段 | **列举已上传的段** | obs:object:ListMultipartUploadParts | √ |
| 取消多段上传任务 | **取消多段上传任务** | obs:object:AbortMultipartUpload | √ |

# 8 附录

## 8.1 状态码

服务器向用户返回的状态码和提示信息如表8-1所示：

**表 8-1** 状态码

| 状态码 | 说明 |
| --- | --- |
| 2xx | 服务器成功返回用户请求的数据。 |
| 4xx | 客户端发出的请求有错误，服务器没有进行新建或修改数据的操作。 |
| 5xx | 服务器发生错误，用户将无法判断发出的请求是否成功。 |

**说明**

注：请使用符合https://www.ietf.org/rfc/rfc2616.txt规定的HTTP/HTTPS请求格式发送API请求。

## 8.2 获取访问密钥（AK/SK）

在调用接口的时候，需要使用AK/SK进行签名验证。AK/SK获取步骤如下：

**步骤1** 登录控制台。

**步骤2** 单击界面右上角的登录用户名，在下拉列表中单击“我的凭证”。

**步骤3** 在左侧导航栏单击“访问密钥”。

**步骤4** 单击“新增访问密钥”，进入“新增访问密钥”页面。

**步骤5** 输入访问密钥描述信息（非必填），单击“确定”。

**步骤6** 通过手机短信、邮箱或者虚拟MFA进行验证，输入对应的验证码，单击“确定”。

**说明**

如果您已开启操作保护，需要通过手机短信、邮箱或者虚拟MFA进行验证，输入对应的验证码。

**步骤7** 单击“立即下载”，浏览器下载访问密钥。

**说明**

为防止访问密钥泄露，建议您将其保存到安全的位置。

**----结束**

# 8.3 获取账号 ID 和用户 ID

在调用接口的时候，部分请求中需要指定账号ID（DomainID）和用户ID（UserID），所以需要先在控制台上获取。获取步骤如下：

**步骤1** 登录控制台。

**步骤2** 单击用户名，在下拉列表中单击“我的凭证”。

在“我的凭证”页面查看账号ID和用户ID。

**----结束**

# 8.4 并发一致性说明

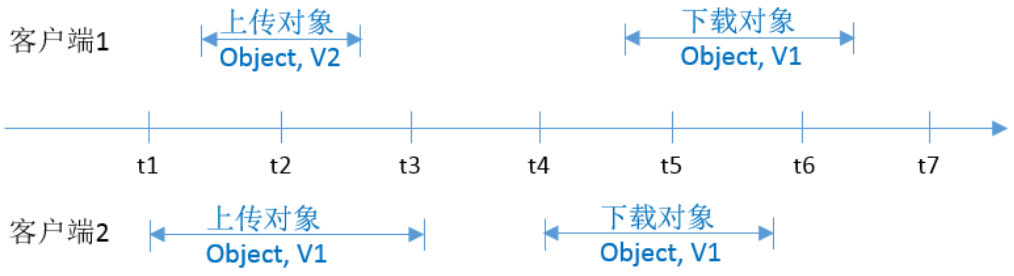
当客户端发起的写/删除请求返回成功之后，客户端可以获取到最新数据。当写操作客户端等待超时、服务端返回500或者503的HTTP响应错误码时，之后的读取操作有可能成功读取到数据，也有可能读不到数据。建议客户端在出现上述错误时，查询数据是否已经上传成功，如果不成功则重新上传。

针对同一个对象或桶的操作，比如多个客户端对同一个对象并行上传、查询和删除时，具体操作结果依赖于操作到达系统的时间和系统内部处理的时延，可能返回不一致的结果。比如，当多个客户端并行上传同一个对象时，系统最后收到的上传请求会覆盖前一个上传的对象。如果需要避免同一个对象被并行访问，需要在上层应用中增加对象的锁机制。

## 并发操作举例

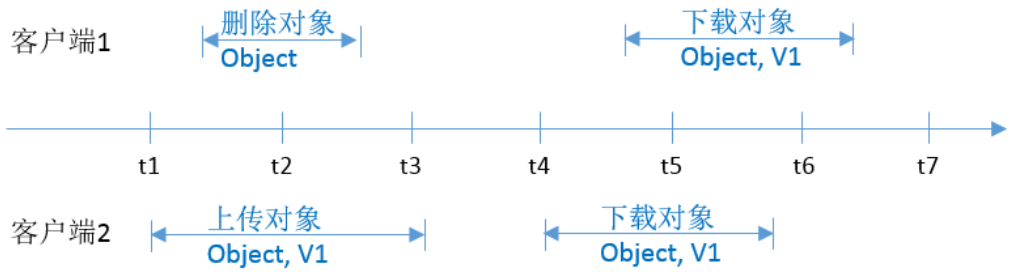
1. 当客户端2正在上传一个对象v1时，客户端1同时上传一个同名的对象v2成功后，不管是客户端1还是客户端2都能够读取最新的对象数据v1，如图8-1所示。

**图 8-1** 并发成功上传同一个对象



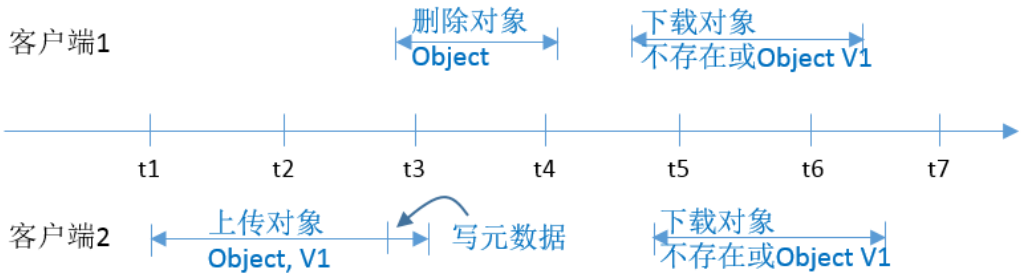
2. 当客户端2上传一个对象v1的时候，如果在对象数据上传且还没有写入对象元数据的过程中，客户端1删除同名的对象成功后，客户端2的上传对象操作仍然返回成功，并且不论客户端1还是客户端2都能读取到对象数据v1，如图8-2所示。

**图 8-2** 并发上传和删除同一个对象（1）



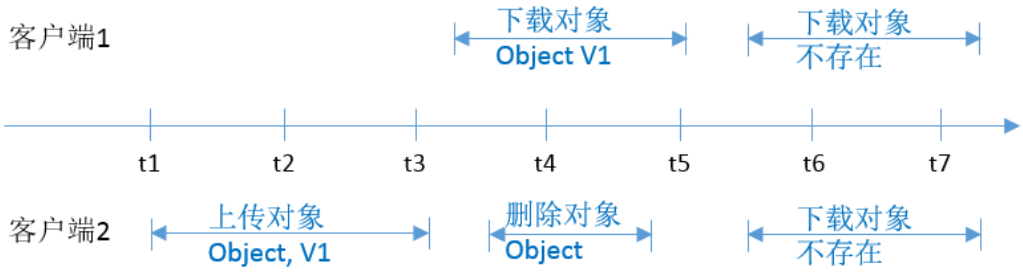

3. 当客户端2上传一个对象v1的时候，如果在对象数据上传完成，系统写入对象元数据的短暂过程中，客户端1发起删除同名的对象成功后，客户端2的上传对象操作仍然返回成功，但是客户端1和客户端2下载对象Object1时，有可能读到对象数据v1，也有可能返回对象不存在，如图8-3所示。

**图 8-3** 并发上传和删除同一个对象（2）



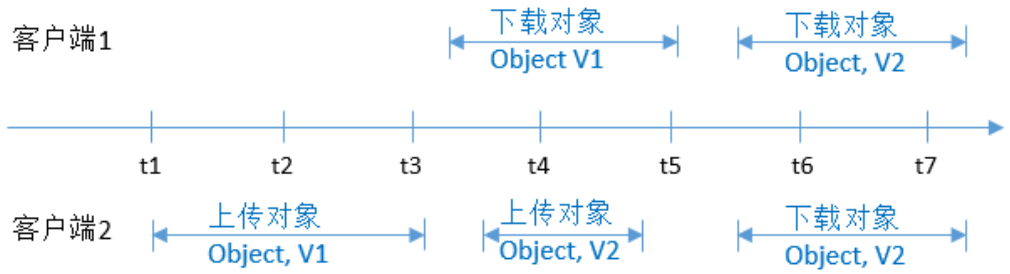

4. 当客户端1下载一个对象的过程中，客户端2发起删除同名对象操作，此时客户端1可能下载到完整的对象数据，也有可能只能下载到对象的一部分数据。当客户端2的删除操作返回成功后，再发起下载对象请求，则返回对象不存在，如图8-4所示。

**图 8-4** 并发下载和删除同一个对象



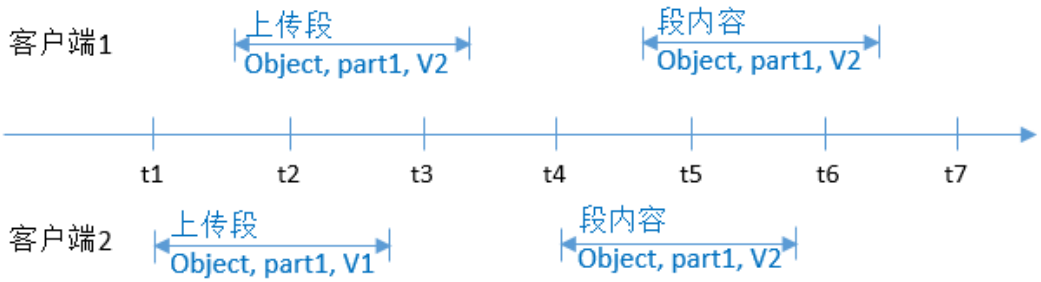

5. 当客户端1下载一个对象的过程中，客户端2发起更新同名对象操作，此时客户端1可能下载到完整的对象数据，也有可能只能下载到对象的一部分数据。当客户端2的更新操作返回成功后，再发起下载对象请求，则返回最新的对象数据，如图8-5所示。

**图 8-5** 并发下载和更新同一个对象




6. 当客户端2正在上传一个对象的段v1时，客户端1同时上传同一个对象的相同段号的段v2成功后，不管是客户端1还是客户端2列举段时都能够列举etag为v2的段信息，如图8-6所示。

**图 8-6** 并发上传同名对象的同名段



# A 修订记录

| 发布日期 | 修订记录 |
| --- | --- |
| 2024-04-15 | 第一次正式发布。 |