Alibaba Cloud
对象存储

API参考

檔案版本：20200330

目錄

1 OSS API 文檔簡介

阿里雲對象儲存服務（Object Storage Service，簡稱OSS），是阿里雲對外提供的海量、安全、低成本、高可靠的雲端儲存體服務。您可以通過本文檔提供的簡單的REST介面，在任何時間、任何地點、任何互連網裝置上進行上傳和下載數據。基於OSS，您可以搭建出各種多媒體分享網站、網盤、個人和企業資料備份等基於大規模數據的服務。

請確保在使用這些介面前，已充分了解了OSS產品說明、使用協議和收費方式。

2 API概覽

OSS提供的API介面如下：

關於Service操作

API	描述
GetService	得到該賬戶下所有Bucket

關於Bucket的操作

API	描述
Put Bucket	建立Bucket
Put Bucket ACL	設定Bucket存取權限
Put Bucket Logging	開啟Bucket日誌
Put Bucket Website	設定Bucket為靜態網站託管模式
Put Bucket Referer	設定Bucket的防盜鏈規則
Put Bucket Lifecycle	設定Bucket中Object的生命週期規則
Get Bucket Acl	獲得Bucket存取權限
Get Bucket Location	獲得Bucket所屬的資料中心位置資訊
Get Bucket Logging	查看Bucket的訪問日誌配置情況
Get Bucket Website	查看Bucket的靜態網站託管狀態
Get Bucket Referer	查看Bucket的防盜鏈規則
Get Bucket Lifecycle	查看Bucket中Object的生命週期規則
Delete Bucket	刪除Bucket
Delete Bucket Logging	關閉Bucket訪問日誌記錄功能
Delete Bucket Website	關閉Bucket的靜態網站託管模式
Delete Bucket Lifecycle	刪除Bucket中Object的生命週期規則
Get Bucket(List Object)	獲得Bucket中所有Object的資訊
Get Bucket Info	獲取Bucket資訊

關於Object的操作

API	描述
Put Object	上傳object
Copy Object	拷貝一個object成另外一個object
Get Object	獲取Object
Delete Object	刪除Object
Delete Multiple Objects	刪除多個Object
Head Object	獲得Object的meta資訊
Post Object	使用Post上傳Object
Append Object	在Object尾追加上傳數據
Put Object ACL	設定Object ACL
Get Object ACL	獲取Object ACL資訊
Callback	上傳回調

關於Multipart Upload的操作

API	描述
Initiate Multipart Upload	初始化MultipartUpload事件
Upload Part	分塊上傳檔案
Upload Part Copy	分塊複製上傳檔案
Complete Multipart Upload	完成整個檔案的Multipart Upload上傳
Abort Multipart Upload	取消Multipart Upload事件
List Multipart Uploads	羅列出所有執行中的Multipart Upload事件
List Parts	羅列出指定Upload ID所屬的所有已經上傳成功Part

跨域資源共用(CORS)

API	描述
Put Bucket cors	在指定Bucket設定一個CORS的規則
Get Bucket cors	獲取指定的Bucket目前的CORS規則
Delete Bucket cors	關閉指定Bucket對應的CORS功能並清空所有規則

API	描述
Option Object	跨域訪問preflight請求

3 公共HTTP頭定義

公共要求標頭（Common Request Headers）

OSS的RESTful介面中使用了一些公共要求標頭。這些要求標頭可以被所有的OSS請求所使用，其詳細定義如下：

名稱	類型	描述
Authorization	字元串	用於驗證請求合法性的認證資訊。 預設值：無 使用場景：非匿名請求
Content-Length	字元串	*RFC2616*中定義的HTTP請求內容長度。 預設值：無 使用場景：需要向OSS提交數據的請求
Content-Type	字元串	*RFC2616*中定義的HTTP請求內容類型。 預設值：無 使用場景：需要向OSS提交數據的請求
Date	字元串	HTTP 1.1協議中規定的GMT時間，例如：Wed, 05 Sep. 2012 23:00:00 GMT 預設值：無
Host	字元串	訪問Host值，格式為：`<bucketname>.oss-cn-hangzhou.aliyuncs.com`。 預設值：無

公共回應標頭（Common Response Headers）

OSS的RESTful介面中使用了一些公共回應標頭。這些回應標頭可以被所有的OSS請求所使用，其詳細定義如下：

名稱	類型	描述
Content-Length	字元串	*RFC2616*中定義的HTTP請求內容長度。 預設值：無 使用場景：需要向OSS提交數據的請求
Connection	枚舉	標明客戶端和OSS伺服器之間的連結狀態。 有效值：open、close 預設值：無
Date	字元串	HTTP 1.1協議中規定的GMT時間，例如：Wed, 05 Sep. 2012 23:00:00 GMT 預設值：無
ETag	字元串	ETag (entity tag) 在每個Object生成的時候被建立，用於標示一個Object的內容。對於Put Object請求建立的Object，ETag值是其內容的MD5值；對於其他方式建立的Object，ETag值是其內容的UUID。ETag值可以用於檢查Object內容是否發生變化。 預設值：無
Server	字元串	生成Response的伺服器。 預設值：AliyunOSS

名稱	類型	描述
x-oss-request-id	字元串	x-oss-request-id是由Aliyun OSS建立，並唯一標識這個response的UUID。如果在使用OSS服務時遇到問題，可以憑藉該欄位聯繫OSS工作人員，快速定位問題。 預設值：無

4 存取控制

4.1 用戶簽名驗證

OSS通過使用AccessKeyId/ AccessKeySecret對稱式加密的方法來驗證某個請求的寄件者身份。AccessKeyId用於標示用戶，AccessKeySecret是用戶用於加密簽名字元串和OSS用來驗證簽名字元串的密鑰，其中AccessKeySecret必須保密，只有用戶和OSS知道。AccessKey 根據所屬帳號的類型有所區分

- **阿里雲賬戶AccessKey：每個阿里雲帳號提供的AccessKey擁有對擁有的資源有完全的許可權**
- **RAM賬戶AccessKey：RAM賬戶由阿里雲帳號授權生成，所擁的AccessKey擁有對特定資源限定的操作許可權**
- **STS臨時訪問憑證：由阿里雲帳號或RAM帳號生成，所擁的AccessKey在限定時間內擁有對特定資源限定的操作許可權。過期許可權收回。**

詳情請參考OSS產品文檔中訪問身分識別驗證訪問身分識別驗證。

當用戶想以個人身份向OSS發送請求時，需要首先將發送的請求按照OSS指定的格式生成簽名字元串；然後使用AccessKeySecret對簽名字元串進行加密產生驗證碼。 OSS收到請求以後，會通過AccessKeyId找到對應的AccessKeySecret，以同樣的方法提取簽名字元串和驗證碼，如果計算出來的驗證碼和提供的一樣即認為該請求是有效；否則，OSS將拒絕處理這次請求，並返回HTTP 403錯誤。

4.2 在Header中包含簽名

用戶可以在HTTP請求中增加 `Authorization` 的Header來包含簽名（Signature）資訊，表明這個消息已被授權。

Authorization欄位計算的方法

```
Authorization = "OSS " + AccessKeyId + ":" + Signature
Signature = base64(hmac-sha1(AccessKeySecret,
            VERB + "\n"
            + Content-MD5 + "\n"
            + Content-Type + "\n"
            + Date + "\n"
            + CanonicalizedOSSHeaders
            + CanonicalizedResource))
```

- `AccessKeySecret` **表示簽名所需的密鑰**
- `VERB`**表示HTTP 要求的Method，主要有PUT，GET，POST，HEAD，DELETE等**

- `\n` 表示分行符號
- `Content-MD5` 表示請求內容數據的MD5值，對消息內容（不包括頭部）計算MD5值獲得128位元位元字，對該數字進行base64編碼而得到。該要求標頭可用於消息合法性的檢查（消息內容是否與發送時一致），如”eB5eJF1ptWaXm4bijSPyxw==”，也可以為空。詳情參看 *RFC2616 Content-MD5*。
- `Content-Type` 表示請求內容的類型，如”application/octet-stream”，也可以為空
- `Date` 表示此次操作的時間，且必須為GMT格式，如”Sun, 22 Nov 2015 08:16:38 GMT”
- `CanonicalizedOSSHeaders` 表示以 x-oss- 為首碼的http header的字典序排列
- `CanonicalizedResource` 表示用戶想要訪問的OSS資源

其中，Date和CanonicalizedResource不能為空；如果請求中的Date時間和OSS伺服器的時間差15分鐘以上，OSS伺服器將拒絕該服務，並返回HTTP 403錯誤。

構建CanonicalizedOSSHeaders的方法

所有以 x-oss- 為首碼的HTTP Header被稱為CanonicalizedOSSHeaders。它的構建方法如下：

1. 將所有以 x-oss- 為首碼的HTTP要求標頭的名字轉換成 小寫 。如`X-OSS-Meta-Name: TaoBao`轉換成`x-oss-meta-name: TaoBao`。
2. 如果請求是以STS獲得的AccessKeyId和AccessKeySecret發送時，還需要將獲得的security-token值，以 `x-oss-security-token:security-token` 的形式加入到簽名字元串中。
3. 將上一步得到的所有HTTP要求標頭按照名字的字典序進行升序排列。
4. 刪除要求標頭和內容之間分隔符號兩端出現的任何空格。如`x-oss-meta-name: TaoBao`轉換成：`x-oss-meta-name:TaoBao`。
5. 將每一個頭和內容用 `\n` 分隔符號分隔拼成最後的CanonicalizedOSSHeaders。

> **說明:**
>
> - CanonicalizedOSSHeaders可以為空，無需添加最後的 `\n`。
> - 如果只有一個，則如 `x-oss-meta-a\n`，注意最後的`\n`。
> - 如果有多個，則如 `x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n`，注意最後的`\n`。

構建CanonicalizedResource的方法

用戶發送請求中想訪問的OSS目標資源被稱為CanonicalizedResource。它的構建方法如下：

1. 將CanonicalizedResource置成Null 字元串 "";
2. 放入要訪問的OSS資源 /BucketName/ObjectName（無ObjectName則CanonicalizedResource為”/BucketName/“，如果同時也沒有BucketName則為“/”）
3. 如果請求的資源套件括子資源(SubResource)，那麼將所有的子資源按照字典序，從小到大排列並以 & 為分隔符號生成子資源字元串。在CanonicalizedResource字元串尾添加 ? 和子資源字元串。此時的CanonicalizedResource如：/BucketName/ObjectName?acl&uploadId=UploadId
4. 如果用戶請求在指定了查詢字元串(QueryString，也叫Http Request Parameters)，那麼將這些查詢字元串及其請求值按照 字典序，從小到大排列，以 & 為分隔符號，按參數添加到CanonicalizedResource中。此時的CanonicalizedResource如：/BucketName/ObjectName?acl&response-content-type=ContentType&uploadId=UploadId。

 说明:

- OSS目前支援的子資源(sub-resource)包括：acl，uploads，location，cors，logging，website，referer，lifecycle，delete，append，tagging，objectMeta，uploadId，partNumber，security-token，position，img，style，styleName，replication，replicationProgress，replicationLocation，cname，bucketInfo，comp，qos，live，status，vod，startTime，endTime，symlink，x-oss-process，response-content-type，response-content-language，response-expires，response-cache-control，response-content-disposition，response-content-encoding等
- 子資源(sub-resource)有三種類型：
  - 資源標識，如子資源中的acl，append，uploadId，symlink等，詳見關於*Bucket*的操作和關於*Object*的操作。
  - 指定返回Header欄位，如 response-***，詳見*GetObject*的Request Parameters。
  - 檔案（Object）處理方式，如 x-oss-process，用於檔案的處理方式，如圖片處理。

計算簽名頭規則

- 簽名的字元串必須為 UTF-8 格式。含有中文字元的簽名字元串必須先進行 UTF-8 編碼，再與 AccessKeySecret計算最終簽名。
- 簽名的方法用*RFC 2104*中定義的HMAC-SHA1方法，其中Key為 AccessKeySecret` 。
- Content-Type 和 Content-MD5 在請求中不是必須的，但是如果請求需要簽名驗證，空值的話以分行符號 \n 代替。

- 在所有非HTTP標準定義的header中，只有以 `x-oss-` 開頭的header，需要加入簽名字元串；其他非HTTP標準header將被OSS忽略（如上例中的x-oss-magic是需要加入簽名字元串的）。
- 以 `x-oss-` 開頭的header在簽名驗證前需要符合以下規範：
  - header的名字需要變成小寫。
  - header按字典序自小到大排序。
  - 分割header name和value的冒號前後不能有空格。
  - 每個Header之後都有一個分行符號“\n”，如果沒有Header，CanonicalizedOSSHeaders就設定為空。

簽名樣本

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

請求	簽名字元串計算公式	簽名字元串
PUT /nelson HTTP /1.0 Content-MD5: eB5eJF1ptWaXm4bijSPyxw== Content-Type: text /html Date: Thu, 17 Nov 2005 18:49:58 GMT Host : oss-example.oss-cn-hangzhou.aliyuncs.com X-OSS-Meta-Author: foo @bar.com X-OSS-Magic: abracadabra	Signature = base64(hmac-sha1(AccessKeySecret,VERB + “\n” + Content-MD5 + “\n ” + Content-Type + “\ n” + Date + “\n” + CanonicalizedOSSHeaders + CanonicalizedResource))	“PUT\n eB5eJF1ptW aXm4bijSPyxw==\n text/ html\n Thu, 17 Nov 2005 18:49:58 GMT\n x-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nels

可用以下方法計算簽名(Signature)：

python範例程式碼：

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

簽名(Signature)計算結果應該為 26NBxoKdsyly4EDv6inkoDft/yA=，因為Authorization = “OSS “ + AccessKeyId + “:” + Signature所以最後Authorization為 “OSS 44CF959000

6BF252F707:26NBxoKdsyly4EDv6inkoDft/yA=”然後加上Authorization頭來組成最後需要發送的消息：

```
PUT /nelson HTTP/1.0
Authorization:OSS 44CF9590006BF252F707:26NBxoKdsyly4EDv6inkoDft/yA=
Content-Md5: eB5eJF1ptWaXm4bijSPyxw==
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
X-OSS-Meta-Author: foo@bar.com
X-OSS-Magic: abracadabra
```

細節分析

- 如果傳入的AccessKeyId不存在或inactive，返回403 Forbidden。錯誤碼：InvalidAccessKeyId。
- 若用戶要求標頭中Authorization值的格式不對，返回400 Bad Request。錯誤碼：InvalidArgument。
- OSS所有的請求都必須使用HTTP 1.1協議規定的GMT時間格式。其中，日期的格式為：

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

  上述日期格式中，“天”所佔位元都是“2 DIGIT”。因此，“Jun 2”、“2 Jun 1982”和“2-Jun-82”都是非法日期格式。
- 如果簽名驗證的時候，頭中沒有傳入Date或者格式不正確，返回403 Forbidden錯誤。錯誤碼：AccessDenied。
- 傳入請求的時間必須在OSS伺服器目前時間之後的15分鐘以內，否則返回403 Forbidden。錯誤碼：RequestTimeTooSkewed。
- 如果AccessKeyId是active的，但OSS判斷用戶的請求發生簽名錯誤，則返回403 Forbidden，並在返回給用戶的response中告訴用戶正確的用於驗證加密的簽名字元串。用戶可以根據OSS的response來檢查自己的簽名字元串是否正確。返回樣本：

  ```
  <?xml version="1.0" ?>
  <Error>
   <Code>
       SignatureDoesNotMatch
   </Code>
   <Message>
       The request signature we calculated does not match the
  signature you provided. Check your key and signing method.
   </Message>
   <StringToSignBytes>
       47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31
  31 20 30 37 3a 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72 65 61 6c 74
   65 73 74 3f 61 63 6c
   </StringToSignBytes>
   <RequestId>
       1E446260FF9B10C2
  ```

```
 </RequestId>
 <HostId>
     oss-cn-hangzhou.aliyuncs.com
 </HostId>
 <SignatureProvided>
     y5H7yzPsA/tP4+0tH1HHvPEwUv8=
 </SignatureProvided>
 <StringToSign>
     GET
Wed, 11 May 2011 07:59:25 GMT
/oss-example?acl
 </StringToSign>
 <OSSAccessKeyId>
     AKIAIVAKMSMOY7VOMRWQ
 </OSSAccessKeyId>
</Error>
```

说明:

- **OSS SDK已經實現簽名，用戶使用OSS SDK不需要關注簽名問題。如果您想了解具體語言的簽名實現，請參考OSS SDK的代碼。OSS SDK簽名實現的檔案如下表：**

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

- **當您自己實現簽名，訪問OSS報 `SignatureDoesNotMatch` 錯誤時，請使用可視化簽名工具確認簽名並排除錯誤。**

常見問題

Content-MD5的計算方法

```
Content-MD5的計算
以消息內容為"123456789"來說，計算這個字元串的Content-MD5
正確的計算方式：
標準中定義的演算法簡單點說就是：
1. 先計算MD5加密的位元組（128位）。
2. 再對這個二進位進行base64編碼（而不是對32位字元串編碼）。
```

以Python為例子：
正確計算的代碼為：
```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```
需要注意
正確的是：hash.digest()，計算出進位數組（128位）
```
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```
常見錯誤是直接對計算出的32位字元串編碼進行base64編碼。
例如，錯誤的是：hash.hexdigest()，計算得到可見的32位字元串編碼
```
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
```
錯誤的MD5值進行base64編碼後的結果：
```
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

4.3 在URL中包含簽名

除了使用Authorization Head，用戶還可以在URL中加入簽名資訊，這樣用戶就可以把該URL轉給第三方實現授權訪問

實現方式

URL簽名樣本:

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

URL簽名，必須至少包含Signature、Expires和OSSAccessKeyId三個參數。

- `Expires` **這個參數的值是一個UNIX時間（自UTC時間1970年1月1號開始的秒數，詳見** *Wikipedia*），**用於標識該URL的逾時時間。如果OSS接收到這個URL請求的時候晚於簽名中包含的Expires參數時，則返回請求逾時的錯誤碼。例如：目前時間是1141889060，開發人員希望建立一個60秒後自動失效的URL，則可以設定Expires時間為1141889120。**
- `OSSAccessKeyId` **即密鑰中的AccessKeyId。**
- `Signature` **表示簽名資訊。所有的OSS支援的請求和各種Header參數，在URL中進行簽名的演算法和在**[*Header*中包含簽名]**的演算法基本一樣。**

```
Signature = urlencode(base64(hmac-sha1(AccessKeySecret,
          VERB + "\n"
          + CONTENT-MD5 + "\n"
          + CONTENT-TYPE + "\n"
          + EXPIRES + "\n"
          + CanonicalizedOSSHeaders
```

```
            + CanonicalizedResource)))
```

其中，與header中包含簽名相比主要區別如下：

- 通過URL包含簽名時，之前的Date參數換成Expires參數。
- 不支援同時在URL和Head中包含簽名。
- 如果傳入的Signature，Expires，OSSAccessKeyId出現不止一次，以第一次為準。
- 請求先驗證請求時間是否晚於Expires時間，然後再驗證簽名。
- 將簽名字元串放到url時，注意要對url進行urlencode

- 臨時用戶URL簽名時，需要攜帶`security-token`，格式如下：

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

範例程式碼

URL中添加簽名的python範例程式碼：

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

說明：

- 上面為python的範例程式碼。
- OSS SDK中提供了提供URL簽名的方法，使用方法請參看SDK檔案中的授權訪問章節。
- OSS SDK的URL簽名實現，請參看下錶：

SDK	URL簽名方法	實現檔案
Java SDK	OSSClient.generatePresignedUrl	*OSSClient.java*
Python SDK	Bucket.sign_url	*api.py*
.Net SDK	OssClient.GeneratePresignedUri	*OssClient.cs*
PHP SDK	OssClient.signUrl	*OssClient.php*
JavaScript SDK	signatureUrl	*object.js*
C SDK	oss_gen_signed_url	*oss_object.c*

細節分析

- 使用在URL中簽名的方式，會將你授權的數據在過期時間以內曝露在互連網上，請預先評估使用風險。
- PUT和GET請求都支援在URL中簽名。
- 在URL中添加簽名時，Signature，Expires，OSSAccessKeyId順序可以交換，但是如果Signature，Expires，OSSAccessKeyId缺少其中的一個或者多個，返回403 Forbidden。錯誤碼：AccessDenied。
- 如果訪問的目前時間晚於請求中設定的Expires時間，返回403 Forbidden。錯誤碼：AccessDenied。
- 如果Expires時間格式錯誤，返回403 Forbidden。錯誤碼：AccessDenied。
- 如果URL中包含參數Signature，Expires，OSSAccessKeyId中的一個或者多個，並且Head中也包含簽名消息，返回消息400 Bad Request。錯誤碼：InvalidArgument。
- 生成簽名字元串時，除Date被替換成Expires參數外，仍然包含content-type、content-md5等上節中定義的Header（請求中雖然仍然有Date這個要求標頭，但不需要將Date加入簽名字元串中）。

4.4 Bucket許可權控制

OSS提供ACL（Access Control List）許可權控制方法，OSS ACL提供Bucket等級的許可權存取控制，Bucket目前有三種存取權限：public-read-write，public-read和private，它們的含義如下：

許可權值	中文名稱	許可權對訪問者的限制
public-read-write	公共讀寫	· 任何人（包括匿名訪問）都可以對該Bucket中的Object進行讀/寫/刪除操作。 · 所有這些操作產生的費用由該Bucket的Owner承擔。
public-read	公共讀，私有寫	· 只有該Bucket的Owner或者授權對象可以對存放在其中的Object進行寫/刪除操作。 · 任何人（包括匿名訪問）可以對Object進行讀操作。

許可權值	中文名稱	許可權對訪問者的限制
private	私有讀寫	· 只有該Bucket的Owner或者授權對象可以對存放在其中的Object進行讀/寫/刪除操作。 · 其他人在未經授權的情況下無法訪問該Bucket內的Object。

说明:

- 用戶建立一個Bucket時，如果不指定Bucket許可權，OSS會自動為該Bucket設定private許可權。
- 對於一個已經存在的Bucket，只有它的建立者可以通過OSS的 Put Bucket Acl介面修改該Bucket的許可權。

5 關於Service操作

5.1 GetService (ListBuckets)

對於服務地址作Get請求可以返回要求者擁有的所有Bucket，其中“/”表示根目錄。

請求文法

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

請求參數

GetService(ListBucket)時，可以通過prefix，marker和max-keys對list做限定，返回部分結果。

表 5-1: 請求參數

名稱	類型	是否必需	描述
prefix	**字元串**	**否**	**限定返回的bucket name必須以prefix作為首碼，可以不設定，不設定時不過濾首碼資訊** **預設值：無**
marker	**字元串**	**否**	**設定結果從marker之後按字母排序的第一個開始返回，可以不設定，不設定時從頭開始返回數據** **預設值：無**
max-keys	**字元串**	**否**	**限定此次返回bucket的最大數，如果不設定，預設為100，max-keys取值不能大於1000** **預設值：100**

響應元素(Response Elements)

表 5-2: 響應元素

名稱	類型	描述
ListAllMyBucketsResult	容器	保存Get Service請求結果的容器。 子節點：Owner, Buckets 父節點：None
Prefix	字元串	本次查詢結果的首碼，當bucket未全部返回時才有此節點。 父節點：ListAllMyBucketsResult
Marker	字元串	標明這次GetService(ListBucket)的起點，當bucket未全部返回時才有此節點。 父節點：ListAllMyBucketsResult
MaxKeys	字元串	響應請求內返回結果的最大數目，當bucket未全部返回時才有此節點。 父節點：ListAllMyBucketsResult
IsTruncated	枚舉字元串	指明是否所有的結果都已經返回：“true”表示本次沒有返回全部結果；“false”表示本次已經返回了全部結果。當bucket未全部返回時才有此節點。 有效值：true、false 父節點：ListAllMyBucketsResult
NextMarker	字元串	表示下一次GetService(ListBucket)可以以此為marker，將未返回的結果返回。當bucket未全部返回時才有此節點。 父節點：ListAllMyBucketsResult
Owner	容器	用於存放Bucket擁有者資訊的容器。 父節點：ListAllMyBucketsResult
ID	字元串	Bucket擁有者的用戶ID。 父節點：ListAllMyBucketsResult.Owner

名稱	類型	描述
DisplayName	字元串	Bucket擁有者的名稱 (目前和ID一致)。 父節點：ListAllMyBucketsResult.Owner
Buckets	容器	保存多個Bucket資訊的容器。 子節點：Bucket 父節點：ListAllMyBucketsResult
Bucket	容器	保存bucket資訊的容器。 子節點：Name, CreationDate, Location 父節點：ListAllMyBucketsResult.Buckets
Name	字元串	Bucket名稱。 父節點：ListAllMyBucketsResult.Buckets.Bucket
CreateDate	時間 (格式：yyyy-mm-ddThh:mm:ss.timezone, e.g., 2011-12-01T12:27:13.000Z)	Bucket建立時間 父節點：ListAllMyBucketsResult.Buckets.Bucket
Location	字元串	Bucket所在的資料中心。 父節點：ListAllMyBucketsResult.Buckets.Bucket
ExtranetEndpoint	字元串	Bucket訪問的外網網域名稱。 父節點：ListAllMyBucketsResult.Buckets.Bucket
IntranetEndpoint	字元串	同區域ECS訪問Bucket的內網網域名稱。 父節點：ListAllMyBucketsResult.Buckets.Bucket
StorageClass	字元串	Bucket儲存類型，支援“Standard”、“IA”、“Archive”。（目前只有部分區域支援“Archive”類型） 父節點：ListAllMyBucketsResult.Buckets.Bucket

細節分析

- **GetService這個API只對驗證通過的用戶有效。**
- **如果請求中沒有用戶驗證資訊（即匿名訪問），返回403 Forbidden。錯誤碼：AccessDenied。**
- **當所有的bucket都返回時，返回的xml中不包含Prefix、Marker、MaxKeys、IsTruncated、NextMarker節點，如果還有部分結果未返回，則增加上述節點，其中NextMarker用於繼續查詢時給marker賦值。**

樣本

請求樣本 I

```
GET / HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
Authorization: OSS nxj7dtl******hcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv
2qUl5jI=
```

返回樣本 I

```
HTTP/1.1 200 OK
Date: Thu, 15 May 2014 11:18:32 GMT
Content-Type: application/xml
Content-Length: 556
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Owner>
    <ID>51264</ID>
    <DisplayName>51264</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <CreationDate>2015-12-17T18:12:43.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-shanghai.aliyuncs.com</ExtranetEndpoint
>
      <IntranetEndpoint>oss-cn-shanghai-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-shanghai</Location>
      <Name>app-base-oss</Name>
      <StorageClass>Standard</StorageClass>
    </Bucket>
    <Bucket>
      <CreationDate>2014-12-25T11:21:04.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEndpoint
>
      <IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-hangzhou</Location>
      <Name>atestleo23</Name>
      <StorageClass>IA</StorageClass>
    </Bucket>
  </Buckets>
```

```
</ListAllMyBucketsResult>
```

請求樣本II

```
GET /?prefix=xz02tphky6fjfiuc&max-keys=1 HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
Authorization: OSS nxj7dtl******hcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv
2qUl5jI=
```

返回樣本II

```
HTTP/1.1 200 OK
Date: Thu, 15 May 2014 11:18:32 GMT
Content-Type: application/xml
Content-Length: 545
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D75
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Prefix>xz02tphky6fjfiuc</Prefix>
  <Marker></Marker>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>true</IsTruncated>
  <NextMarker>xz02tphky6fjfiuc0</NextMarker>
  <Owner>
    <ID>ut_test_put_bucket</ID>
    <DisplayName>ut_test_put_bucket</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <CreationDate>2014-05-15T11:18:32.000Z</CreationDate>
      <ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEndpoint
>
      <IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</
IntranetEndpoint>
      <Location>oss-cn-hangzhou</Location>
      <Name>xz02tphky6fjfiuc0</Name>
      <StorageClass>Standard</StorageClass>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

6 關於Bucket的操作

7 關於Object操作

7.1 基礎操作

7.1.1 PutObject

PutObject**介面用於上傳檔案**。

請求文法

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

請求Header

表 7-1: 請求Header

名稱	類型	描述
Cache-Control	字元串	指定該Object被下載時的網頁的緩存行為；更詳細描述請參照*RFC2616*。 預設值：無
Content-Disposition	字元串	指定該Object被下載時的名稱；更詳細描述請參照*RFC2616*。 預設值：無
Content-Encoding	字元串	指定該Object被下載時的內容編碼格式；更詳細描述請參照*RFC2616*。 預設值：無

名稱	類型	描述
Content-Md5	字元串	根據協議RFC 1864對消息內容（不包括頭部）計算Md5值獲得128位元位元字，對該數字進行base64編碼為一個消息的Content-Md5值。該要求標頭可用於消息合法性的檢查（消息內容是否與發送時一致）。雖然該要求標頭是可選項，OSS建議用戶使用該要求標頭進行端到端檢查。 預設值：無 限制：無
Expires	字元串	過期時間；更詳細描述請參照*RFC2616*。 預設值：無 說明: OSS不會對這個值進行限制和驗證。
x-oss-server-side-encryption	字元串	指定oss建立object時的伺服器端加密編碼演算法。 合法值：AES256 或 KMS 說明: 您需要購買KMS套件，才可以使用KMS密碼編譯演算法，否則會報KmsServiceNotEnabled錯誤碼
x-oss-object-acl	字元串	指定oss建立object時的存取權限。 合法值：public-read，private，public-read-write

細節分析

- 如果用戶上傳了Content-Md5要求標頭，OSS會計算body的Content-Md5並檢查一致性，如果不一致，將返回InvalidDigest錯誤碼。
- 如果要求標頭中的“Content-Length”值小於實際請求體（body）中傳輸的數據長度，OSS仍將成功建立檔案；但Object大小隻等於“Content-Length”中定義的大小，其他數據將被丟棄。
- 如果試圖添加的Object的同名檔案已經存在，並且有存取權限。新添加的檔案將覆蓋原來的檔案，成功返回200 OK。
- 如果在PutObject的時候，攜帶以x-oss-meta-為首碼的參數，則視為user meta，比如x-oss-meta-location。一個Object可以有多個類似的參數，但所有的user meta總大小不能超過8k。

- 如果Header不是*chunked encoding*編碼方式，且沒有加入Content length參數，會返回411 Length Required錯誤。錯誤碼：MissingContentLength。
- 如果設定了長度，但是沒有發送消息Body，或者發送的body大小小於給定大小，伺服器會一直等待，直到time out，返回400 Bad Request消息。錯誤碼：RequestTimeout。
- 如果試圖添加的Object所在的Bucket不存在，返回404 Not Found錯誤。錯誤碼：NoSuchBucket。
- 如果試圖添加的Object所在的Bucket沒有存取權限，返回403 Forbidden錯誤。錯誤碼：AccessDenied。
- 如果添加檔案長度超過5G，返回錯誤訊息400 Bad Request。錯誤碼：InvalidArgument。
- 如果傳入的Object key長度大於1023位元組，返回400 Bad Request。錯誤碼：InvalidObjectName。
- PUT一個Object的時候，OSS支援5個 HTTP *RFC2616*協議規定的Header 欄位：Cache-Control、Expires、Content-Encoding、Content-Disposition、Content-Type。如果上傳Object時設定了這些Header，則這個Object被下載時，相應的Header值會被自動化佈建成上傳時的值。
- 如果上傳Object時指定了x-oss-server-side-encryption Header，則必須設定其值為AES256，否則會返回400和相應錯誤提示：InvalidEncryptionAlgorithmError。指定該Header後，在回應標頭中也會返回該Header，OSS會對上傳的Object進行加密編碼儲存，當這個Object被下載時，回應標頭中會包含x-oss-server-side-encryption，值被設定成該Object的密碼編譯演算法。

常見問題

- Content-Md5計算方式錯誤

  以上傳的內容為 `0123456789` 為例，計算這個字元串的Content-Md5。

以上傳的內容為"123456789"來說，計算這個字元串的Content-Md5 正確的計算方式：標準中定義的演算法為： 先計算Md5加密的位元組（128位），再對這個二進位進行base64編碼（而不是對32位字元串編碼）。 以Python為例子，正確計算的代碼為：

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

需要注意正確的是：`hash.digest()`，計算出進位數組（128位） `>>> hash.digest() 'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'`。常見錯誤是直接對計算出的32位字元串編碼進行base64編碼。 例如，錯誤的是：`hash.hexdigest()`，計算得到可見的32位字元串

編碼 `>>> hash.hexdigest() '781e5e245d69b566979b86e28d23f2c7'` **錯誤的Md5值進行base64編碼後的結果：** `>>> base64.b64encode(hash.hexdigest()) 'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='`

樣本

請求樣本：

```
PUT /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
Expires: Fri, 28 Feb 2012 05:38:42 GMT
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Fri, 24 Feb 2012 06:03:28 GMT
Content-Type: image/jpg
Content-Length: 344606
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PR
rk=

[344606 bytes of object data]
```

返回樣本：

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

7.1.2 CopyObject

`CopyObject`**介面用於拷貝一個在OSS上已經存在的object成另外一個object。**

該介面可以發送一個PUT請求給OSS，並在PUT要求標頭中添加元素`x-oss-copy-source`**來指定拷貝源。OSS會自動判斷出這是一個Copy操作，並直接在伺服器端執行該操作。如果拷貝成功，則返回新的object資訊給用戶。**

該操作適用於拷貝小於1GB的檔案，當拷貝一個大於1GB的檔案時，必須使用Multipart Upload操作，具體見Upload Part Copy。

Copy操作的源Bucket和目標Bucket必須是同一個Region，不同Region的Bucket不能執行Copy Object操作。

請求文法

```
PUT /DestObjectName HTTP/1.1
Host: DestBucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

```
x-oss-copy-source: /SourceBucketName/SourceObjectName
```

請求Header

名稱	類型	描述
x-oss-copy-source	字元串	複製源地址（必須有可讀許可權） 預設值：無
x-oss-copy-source-if-match	字元串	如果源Object的ETag值和用戶提供的ETag相等，則執行拷貝操作，並返回200；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-none-match	字元串	如果源Object的ETag值和用戶提供的ETag不相等，則執行拷貝操作，並返回200；否則返回304 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-unmodified-since	字元串	如果傳入參數中的時間等於或者晚於檔案實際修改時間，則正常傳輸檔案，並返回200 OK；否則返回412 precondition failed錯誤。 預設值：無
x-oss-copy-source-if-modified-since	字元串	如果源Object自從用戶指定的時間以後被修改過，則執行拷貝操作；否則返回304 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-metadata-directive	字元串	有效值為COPY和REPLACE。如果該值設為COPY，則新的Object的meta都從源Object複製過來；如果設為REPLACE，則忽視所有源Object的meta值，而採用用戶這次請求中指定的meta值；其他值則返回400 HTTP錯誤碼。注意該值為COPY時，源Object的x-oss-server-side-encryption的meta值不會進行拷貝。 取值： · COPY （預設值） · REPLACE

名稱	類型	描述
x-oss-server-side-encryption	字元串	指定oss建立目標object時的伺服器端熵編碼密碼編譯演算法 取值：AES256 或 KMS **说明:** 您需要購買KMS套件，才可以使用KMS密碼編譯演算法，否則會報KmsServiceNotEnabled錯誤碼。
x-oss-object-acl	字元串	指定oss建立object時的存取權限。 取值：public-read，private，public-read-write

響應元素(Response Elements)

表 7-2: 響應元素

名稱	類型	描述
CopyObjectResult	字元串	Copy Object的結果。 預設值：無
ETag	字元串	新Object的ETag值。 父元素：CopyObjectResult
LastModified	字元串	新Object最後更新時間。 父元素：CopyObjectResult

細節分析

- 可以通過拷貝操作來實現修改已有Object的meta資訊。
- 如果拷貝操作的源Object地址和目標Object地址相同，則無論x-oss-metadata-directive為何值，都會直接替換源Object的meta資訊。
- OSS支援拷貝操作的四個預判斷Header任意個同時出現，相應邏輯參見Get Object操作的細節分析。
- 拷貝操作需要要求者對源Object有讀許可權。
- 源Object和目標Object必須屬於同一個資料中心，否則返回403 AccessDenied，錯誤資訊為：Target object does not reside in the same data center as source object。
- 拷貝操作的計費統計會對源Object所在的Bucket增加一次Get請求次數，並對目標Object所在的Bucket增加一次Put請求次數，以及相應的新增儲存空間。

- 拷貝操作涉及到的要求標頭，都是以“x-oss-”開頭的，所以要加入簽名字元串中。
- 若在拷貝操作中指定了x-oss-server-side-encryption要求標頭，並且請求值合法（為AES256），則無論源Object是否進行過伺服器端加密編碼，拷貝之後的目標Object都會進行伺服器端加密編碼。並且拷貝操作的回應標頭中會包含x-oss-server-side-encryption，值被設定成目標Object的密碼編譯演算法。在這個目標Object被下載時，回應標頭中也會包含x-oss-server-side-encryption，值被設定成該Object的密碼編譯演算法；若拷貝操作中未指定x-oss-server-side-encryption要求標頭，則無論源Object是否進行過伺服器端加密編碼，拷貝之後的目標Object都是未進行過伺服器端加密編碼加密的數據。
- 拷貝操作中x-oss-metadata-directive要求標頭為COPY（預設值）時，並不拷貝源Object的x-oss-server-side-encryption值，即目標Object是否進行伺服器端加密編碼只根據COPY操作是否指定了x-oss-server-side-encryption要求標頭來決定。
- 若在拷貝操作中指定了x-oss-server-side-encryption要求標頭，並且請求值非AES256，則返回400和相應的錯誤提示：InvalidEncryptionAlgorithmError。
- 如果拷貝的檔案大小大於1GB，會返回400和錯誤提示：EntityTooLarge。
- 該操作不能拷貝通過Append追加上傳方式產生的object。
- 如果檔案類型為符號連結，只拷貝符號連結。

樣本

請求樣本：

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

返回樣本：

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

7.1.3 GetObject

用於獲取某個Object，此操作要求用戶對該Object有讀許可權。

請求文法

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

請求參數(Request Parameters)

OSS支援用戶在發送GET請求時，可以自訂OSS返回請求中的一些Header，前提條件是用戶發送的GET請求必須攜帶簽名。這些Header包括：

名稱	類型	描述
response-content-type	字元串	設定OSS返回請求的content-type頭 預設值：無
response-content-language	字元串	設定OSS返回請求的content-language頭 預設值：無
response-expires	字元串	設定OSS返回請求的expires頭 預設值：無
response-cache-control	字元串	設定OSS返回請求的cache-control頭 預設值：無
response-content-disposition	字元串	設定OSS返回請求的content-disposition頭 預設值：無
response-content-encoding	字元串	設定OSS返回請求的content-encoding頭 預設值：無

請求Header

表 7-3: 請求Header

名稱	類型	描述
Range	字元串	指定檔案傳輸的範圍。如，設定 bytes=0-9，表示傳送第0到第9這10個字元。 預設值：無

名稱	類型	描述
If-Modified-Since	字元串	如果指定的時間早於實際修改時間，則正常傳送檔案，並返回200 OK；否則返回304 not modified 預設值：無 時間格式：GMT時間，例如Fri, 13 Nov 2015 14:47:53 GMT
If-Unmodified-Since	字元串	如果傳入參數中的時間等於或者晚於檔案實際修改時間，則正常傳輸檔案，並返回200 OK；否則返回412 precondition failed錯誤 預設值：無 時間格式：GMT時間，例如Fri, 13 Nov 2015 14:47:53 GMT
If-Match	字元串	如果傳入期望的ETag和object的 ETag匹配，則正常傳輸檔案，並返回200 OK；否則返回412 precondition failed錯誤 預設值：無
If-None-Match	字元串	如果傳入的ETag值和Object的ETag不匹配，則正常傳輸檔案，並返回200 OK；否則返回304 Not Modified 預設值：無

細節分析

- GetObject通過range參數可以支援斷點續傳，對於比較大的Object建議使用該功能。
- 如果在要求標頭中使用Range參數；則返回消息中會包含整個檔案的長度和此次返回的範圍，例如：Content-Range: bytes 0-9/44，表示整個檔案長度為44，此次返回的範圍為0-9。如果不符合範圍規範，則傳送整個檔案，並且不在結果中提及Content-Range。
- 如果“If-Modified-Since”元素中設定的時間不符合規範，直接返迴文件，並返回200 OK。
- If-Modified-Since和If-Unmodified-Since可以同時存在，If-Match和If-None-Match也可以同時存在。
- 如果包含If-Unmodified-Since並且不符合或者包含If-Match並且不符合，返回412 precondition failed
- 如果包含If-Modified-Since並且不符合或者包含If-None-Match並且不符合，返回304 Not Modified
- 如果檔案不存在返回404 Not Found錯誤。錯誤碼：NoSuchKey。

- OSS不支援在匿名訪問的GET請求中，通過請求參數來自訂返回請求的header。
- 在自訂OSS返回請求中的一些Header時，只有請求處理成功（即返回碼為200時），OSS才會將請求的header設定成用戶GET請求參數中指定的值。
- 若該Object為進行伺服器端熵編碼加密儲存的，則在GET請求時會自動解密返回給用戶，並且在回應標頭中，會返回x-oss-server-side-encryption，其值表明該Object的伺服器端密碼編譯演算法。
- 需要將返回內容進行 GZIP壓縮傳輸的用戶，需要在請求的Header中顯示方式加入 Accept-Encoding:gzip，OSS會根據檔案的Content-Type和檔案大小，判斷是否返回給用戶經過 GZIP 壓縮的數據。如果採用了GZIP壓縮則不會附帶etag 資訊。目前OSS支援GZIP壓縮的Content-Type為HTML、Javascript、CSS、XML、RSS、Json，檔案大小需不小於1k。
- 如果檔案類型為符號連結，返回目標檔案的內容。並且， 回應標頭中`Content-Length`、`ETag`、`Content-Md5`均為目標檔案的元資訊；`Last-Modified`是目標檔案和符號連結的最大值；其他均為符號連結的元資訊。
- 如果檔案類型為符號連結，並且目標檔案不存在，返回404 Not Found錯誤。錯誤碼：SymlinkTargetNotExist。
- 如果檔案類型為符號連結，並且目標檔案類型是符號連結，返回400 Bad request錯誤。錯誤碼：InvalidTargetType。
- 對於Archive歸檔類型，Object下載需要提交Restore請求，並等待Restore完成；只有在Object的Restore操作完成且逾時前，Object才能被下載：
  - 如果沒有提交Restore請求，或者上一次提交Restore已經逾時， 則返回403錯，錯誤碼為：InvalidObjectState。
  - 或者已經提交Restore請求，但數據的Restore操作還沒有完成， 則返回403錯，錯誤碼為：InvalidObjectState。
  - 只有Restore完成，且沒有逾時，數據才能直接下載。

樣本

請求樣本：

```
GET /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 06:38:30 GMT
Authorization:OSS qn6qrrqxo2oawuk53otfjbyc:UNQDb7GapEgJCZkcde6OhZ9Jfe8
=
```

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 3a89276f-2e2d-7965-3ff9-51c875b99c41
x-oss-object-type: Normal
Date: Fri, 24 Feb 2012 06:38:30 GMT
```

```
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
[344606 bytes of object data]
```

Range請求樣本：

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

返回樣本：

```
HTTP/1.1 206 Partial Content
x-oss-request-id: 28f6508f-15ea-8224-234e-c0ce40734b89
x-oss-object-type: Normal
Date: Fri, 28 Feb 2012 05:38:42 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Accept-Ranges: bytes
Content-Range: bytes 100-900/344606
Content-Type: image/jpg
Content-Length: 801
Server: AliyunOSS
[801 bytes of object data]
```

自訂返回消息頭的請求樣本：

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

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
x-oss-object-type: Normal
Date: Fri, 24 Feb 2012 06:09:48 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE "
Content-Length: 344606
Connection: keep-alive
Content-disposition: attachment; filename:testing.txt
Content-language: 中文
Content-encoding: utf-8
Content-type: text
Cache-control: no-cache
Expires: Fri, 24 Feb 2012 17:00:00 GMT
Server: AliyunOSS
```

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

符號連結的請求樣本：

```
GET /link-to-oss.jpg HTTP/1.1
Accept-Encoding: identity
Date: Tue, 08 Nov 2016 03:17:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Authorization:OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM
=
```

返回樣本：

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

Archive類型Object的Restore操作已經完成時的請求樣本：

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

返回樣本

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

7.1.4 AppendObject

AppendObject**介面用於以追加寫的方式上傳檔案**。

通過Append Object操作建立的Object類型為Appendable Object，而通過Put Object上傳的Object是Normal Object。

請求文法

```
POST /ObjectName?append&position=Position HTTP/1.1
Content-Length: ContentLength
Content-Type: ContentType
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

請求Header

名稱	類型	描述
Cache-Control	字元串	指定該Object被下載時的網頁的緩存行為；更詳細描述請參照*RFC2616*。 預設值：無
Content-Disposition	字元串	指定該Object被下載時的名稱；更詳細描述請參照*RFC2616*。 預設值：無
Content-Encoding	字元串	指定該Object被下載時的內容編碼格式；更詳細描述請參照*RFC2616*。 預設值：無
Content-MD5	字元串	根據協議RFC 1864對消息內容（不包括頭部）計算MD5值獲得128位元位元字，對該數字進行base64編碼為一個消息的Content-MD5值。該要求標頭可用於消息合法性的檢查（消息內容是否與發送時一致）。雖然該要求標頭是可選項，OSS建議用戶使用該要求標頭進行端到端檢查。 預設值：無 限制：無
Expires	整數	過期時間；更詳細描述請參照*RFC2616*。 預設值：無
x-oss-server-side-encryption	字元串	指定oss建立object時的伺服器端加密編碼演算法。 合法值：AES256 或 KMS **说明:** 您需要購買KMS套件，才可以使用KMS密碼編譯演算法，否則會報KmsServiceNotEnabled錯誤碼。

名稱	類型	描述
x-oss-object-acl	字元串	指定oss建立object時的存取權限。 合法值：public-read，private，public-read-write

響應Header

名稱	類型	描述
x-oss-next-append-position	64位整型	指明下一次請求應當提供的position。實際上就是當前Object長度。當Append Object成功返回，或是因position和Object長度不匹配而引起的409錯誤時，會包含此header。
x-oss-hash-crc64ecma	64位整型	表明Object的64位CRC值。該64位CRC根據*ECMA-182*標準計算得出。

和其他動作的關係

- 不能對一個非Appendable Object進行Append Object操作。例如，已經存在一個同名Normal Object時，Append Object調用返回409，錯誤碼ObjectNotAppendable。
- 對一個已經存在的Appendable Object進行Put Object操作，那麼該Appendable Object會被新的Object覆蓋，類型變為Normal Object。
- Head Object操作會返回x-oss-object-type，用於表明Object的類型。對於Appendable Object來說，該值為Appendable。對Appendable Object，Head Object也會返回上述的x-oss-next-append-position和x-oss-hash-crc64ecma。
- Get Bucket（List Objects）請求的響應XML中，會把Appendable Object的Type設為Appendable
- 不能使用Copy Object來拷貝一個Appendable Object，也不能改變它的伺服器端加密的屬性。可以使用Copy Object來改變用戶自訂元資訊。

細節分析

- URL參數append和position均為CanonicalizedResource，需要包含在簽名中。
- URL的參數必須包含append，用來指定這是一個Append Object操作。
- URL查詢參數還必須包含position，其值指定從何處進行追加。首次追加操作的position必須為0，後續追加操作的position是Object的當前長度。例如，第一次Append Object請求指定position值為0，content-length是65536；那麼，第二次Append Object需要指定position為65536。每次操作成功後，回應標頭部x-oss-next-append-position也會標明下一次追加的position。

- 如果position的值和當前Object的長度不一致，OSS會返回409錯誤，錯誤碼為PositionNotEqualToLength。發生上述錯誤時，用戶可以通過回應標頭部x-oss-next-append-position來得到下一次position，並再次進行請求。
- 當Position值為0時，如果沒有同名Appendable Object，或者同名Appendable Object長度為0，該請求成功；其他情況均視為Position和Object長度不匹配的情形。
- 當Position值為0，且沒有同名Object存在，那麼Append Object可以和Put Object請求一樣，設定諸如x-oss-server-side-encryption之類的請求Header。這一點和Initiate Multipart Upload是一樣的。如果在Position為0的請求時，加入了正確的x-oss-server-side-encryption頭，那麼後續的Append Object回應標頭部也會包含x-oss-server-side-encryption頭，其值表明密碼編譯演算法。後續如果需要更改meta，可以使用Copy Object請求。
- 由於並發的關係，即使用戶把position的值設為了x-oss-next-append-position，該請求依然可能因為PositionNotEqualToLength而失敗。
- Append Object產生的Object長度限制和Put Object一樣。
- 每次Append Object都會更新該Object的最後修改時間。
- 在position值正確的情況下，對已存在的Appendable Object追加一個長度為0的內容，該操作不會改變Object的狀態。

CRC64的計算方式

Appendable Object的CRC採用*ECMA-182*標準，和XZ的計算方式一樣。用Boost CRC模組的方式來定義則有：

```
typedef boost::crc_optimal<64, 0x42F0E1EBA9EA3693ULL, 0xffffffff
ffffffffULL, 0xffffffffffffffffULL, true, true> boost_ecma;

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

或是用Python crcmod的方式為：

```
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)

print do_crc64(“123456789”)
```

樣本

請求樣本：

```
POST /oss.jpg?append&position=0 HTTP/1.1
```

```
Host: oss-example.oss.aliyuncs.com
Cache-control: no-cache
Expires: Wed, 08 Jul 2015 16:57:01 GMT
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1717
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PR
rk=

[1717 bytes of object data]
```

返回樣本：

```
HTTP/1.1 200 OK
Date: Wed, 08 Jul 2015 06:57:01 GMT
ETag: "0F7230CAA4BE94CCBDC99C5500000000"
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
x-oss-hash-crc64ecma: 14741617095266562575
x-oss-next-append-position: 1717
x-oss-request-id: 559CC9BDC755F95A64485981
```

7.1.5 DeleteObject

DeleteObject**用於刪除某個Object**。

請求文法

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

細節分析

- **DeleteObject要求對該Object要有寫入權限**。
- **如果要刪除的Object不存在，OSS也返回狀態碼204（No Content）**。
- **如果Bucket不存在，返回404 Not Found**。
- **如果檔案類型為符號連結，只刪除符號連結自身**。

樣本

請求樣本：

```
DELETE /copy_oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 07:45:28 GMT
```

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:zUglwRPGkbByZxm1+y4eyu+
NIUs=
```

返回樣本：

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

7.1.6 DeleteMultipleObjects

DeleteMultipleObjects支援用戶通過一個HTTP請求刪除同一個Bucket中的多個Object。

Delete Multiple Objects操作支援一次請求內最多刪除1000個Object，並提供兩種返回模式：詳細(verbose)模式和簡單(quiet)模式。

- **詳細模式：OSS返回的消息體中會包含每一個刪除Object的結果。**
- **簡單模式：OSS返回的消息體中只包含刪除過程中出錯的Object結果；如果所有刪除都成功的話，則沒有消息體。**

請求文法

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

請求參數(Request Parameters)

Delete Multiple Objects時，可以通過encoding-type對返回結果中的Key進行編碼。

名稱	描述
encoding-type	**指定對返回的Key進行編碼，目前支援url編碼。Key使用UTF-8字元，但xml 1.0標準不支援解析一些控制字元，比如ascii值從0到10的字元。對於Key中包含xml 1.0標準不支援的控制字元，可以通過指定encoding-type對返回的Key進行編碼。** **資料類型：字元串** **預設值：無** **可選值：**url

請求元素(Request Elements)

名稱	類型	描述
Delete	**容器**	**保存Delete Multiple Object請求的容器。** **子節點：一個或多個Object元素，可選的Quiet元素** **父節點： None**
Key	**字元串**	**被刪除Object的名字。** **父節點：Object**
Object	**容器**	**保存一個Object資訊的容器。** **子節點：key** **父節點：Delete**
Quiet	**枚舉字元串**	**開啟“簡單”響應模式的開關。** **有效值：**true、false **預設值：**false **父節點：Delete**

響應元素(Response Elements)

名稱	類型	描述
Deleted	容器	保存被成功刪除的Object的容器。 子節點：Key 父節點：DeleteResult
DeleteResult	容器	保存Delete Multiple Object請求結果的容器。 子節點：Deleted 父節點：None
Key	字元串	OSS執行刪除操作的Object名字。 父節點：Deleted
EncodingType	字元串	指明返回結果中編碼使用的類型。如果請求的參數中指定了encoding-type，那返回的結果會對Key進行編碼。 父節點：容器

細節分析

- Delete Multiple Objects請求必須填Content-Length和Content-MD5欄位。OSS會根據這些欄位驗證收到的消息體是正確的，之後才會執行刪除操作。
- 生成Content-MD5欄位內容方法：首先將Delete Multiple Object請求內容經過MD5加密後得到一個128位位元組數組；再將該位元組數組用base64演算法編碼；最後得到的字元串即是Content-MD5欄位內容。
- Delete Multiple Objects請求預設是詳細(verbose)模式。
- 在Delete Multiple Objects請求中刪除一個不存在的Object，仍然認為是成功的。
- Delete Multiple Objects的消息體最大允許2MB的內容，超過2MB會返回MalformedXML錯誤碼。
- Delete Multiple Objects請求最多允許一次刪除1000個Object；超過1000個Object會返回MalformedXML錯誤碼。
- 如果用戶上傳了Content-MD5要求標頭，OSS會計算body的Content-MD5並檢查一致性，如果不一致，將返回InvalidDigest錯誤碼。

樣本

請求樣本I:

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

返回樣本:

```
HTTP/1.1 200 OK
x-oss-request-id: 78320852-7eee-b697-75e1-b6db0f4849e7
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length: 244
Content-Type: application/xml
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <Deleted>
       <Key>multipart.data</Key>
    </Deleted>
    <Deleted>
       <Key>test.jpg</Key>
    </Deleted>
    <Deleted>
       <Key>demo.jpg</Key>
    </Deleted>
</DeleteResult>
```

請求樣本II:

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:WuV0Jks8RyGSNQrBca64
kEExJDs=
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Quiet>true</Quiet>
  <Object>
    <Key>multipart.data</Key>
```

```
  </Object>
  <Object>
    <Key>test.jpg</Key>
  </Object>
  <Object>
    <Key>demo.jpg</Key>
  </Object>
</Delete>
```

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

7.1.7 HeadObject

HeadObject**只返回某個Object的meta資訊，不返迴文件內容**。

請求文法

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

請求Header

名稱	類型	描述
If-Modified-Since	字元串	如果指定的時間早於實際修改時間，則返回200 OK和Object Meta；否則返回304 not modified 預設值：無
If-Unmodified-Since	字元串	如果傳入參數中的時間等於或者晚於檔案實際修改時間，則返回200 OK和Object Meta；否則返回412 precondition failed錯誤 預設值：無

名稱	類型	描述
If-Match	字元串	如果傳入期望的ETag和object的 ETag匹配，則返回200 OK和Object Meta；否則返回412 precondition failed錯誤 預設值：無
If-None-Match	字元串	如果傳入的ETag值和Object的ETag不匹配，則返回200 OK和Object Meta；否則返回304 Not Modified 預設值：無

細節分析

- 不論正常返回200 OK還是非正常返回，Head Object都不返回消息體。
- HeadObject支援在頭中設定If-Modified-Since, If-Unmodified-Since, If-Match，If-None-Match的查詢條件。具體規則請參見GetObject中對應的選項。如果沒有修改，返回304 Not Modified。
- 如果用戶在PutObject的時候傳入以x-oss-meta-為開頭的user meta，比如x-oss-meta-location，返回消息時，返回這些user meta。
- 如果檔案不存在返回404 Not Found錯誤。
- 若該Object為進行伺服器端熵編碼加密儲存的，則在Head請求回應標頭中，會返回x-oss-server-side-encryption，其值表明該Object的伺服器端密碼編譯演算法。
- 如果檔案類型為符號連結， 回應標頭中`Content-Length`、`ETag`、`Content-Md5`均為目標檔案的元資訊；`Last-Modified`是目標檔案和符號連結的最大值；其他均為符號連結元資訊。
- 如果檔案類型為符號連結，並且目標檔案不存在，返回404 Not Found錯誤。錯誤碼：SymlinkTargetNotExist。
- 如果檔案類型為符號連結，並且目標檔案類型是符號連結，返回400 Bad request錯誤。錯誤碼：InvalidTargetType。
- x-oss-storage-class展示Object的儲存類型：Standard，IA，Archive。

- **如果Bucket類型為Archive，且用戶已經提交Restore請求，則回應標頭中會以x-oss-restore返回該Object的Restore狀態，分如下幾種情況：**
  - **如果沒有提交Restore或者Restore已經逾時，則不返回該欄位。**
  - **如果已經提交Restore，且Restore沒有時完成，則返回的x-oss-restore值為: ongoing-request=”true”。**
  - **如果已經提交Restore，且Restore已經完成，則返回的x-oss-restore值為: ongoing-request=”false”, expiry-date=”Sun, 16 Apr 2017 08:12:33 GMT”。**

樣本

請求樣本：

```
HEAD /oss.jpg HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 07:32:52 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:JbzF2LxZUtanlJ5dLA092wpDC/
E=
```

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
x-oss-object-type: Normal
x-oss-storage-class: Archive
Date: Fri, 24 Feb 2012 07:32:52 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
Content-Type: image/jpg
Connection: keep-alive
Server: AliyunOSS
```

提交Restore請求但restore沒有完成時的請求樣本

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

返回樣本

```
HTTP/1.1 200 OK
x-oss-request-id: 58F71A164529F18D7F000045
x-oss-object-type: Normal
x-oss-storage-class: Archive
x-oss-restore: ongoing-request="true"
Date: Sat, 15 Apr 2017 07:32:52 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
Content-Type: image/jpg
Connection: keep-alive
```

```
Server: AliyunOSS
```

提交Restore請求且restore已經完成時的請求樣本

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

返回樣本

```
HTTP/1.1 200 OK
x-oss-request-id: 58F725344529F18D7F000055
x-oss-object-type: Normal
x-oss-storage-class: Archive
x-oss-restore: ongoing-request="false", expiry-date="Sun, 16 Apr 2017
08:12:33 GMT"
Date: Sat, 15 Apr 2017 09:35:51 GMT
Last-Modified: Sat, 15 Apr 2017 06:07:48 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 344606
```

7.1.8 GetObjectMeta

GetObjectMeta**用來獲取某個Bucket下的某個Object的基本meta資訊，包括該Object的ETag、Size（檔案大小）、LastModified，並不返回其內容。**

請求文法

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

細節分析

- **無論正常返回還是非正常返回，Get Object Meta均不返回消息體。**
- **Get Object Meta需包含objectMeta請求參數，否則表示Get Object請求。**
- **如果檔案不存在返回404 Not Found錯誤。**
- **Get Object Meta相比Head Object更輕量，僅返回指定Object的少量基本meta資訊，包括該Object的ETag、Size（檔案大小）、LastModified，其中Size由回應標頭Content-Length的數值表示。**
- **如果檔案類型為符號連結，返回符號連結自身資訊。**

樣本

請求樣本：

```
GET /oss.jpg?objectMeta HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
```

```
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0Fmg
brQ0=
```

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
ETag: "5B3C1A2E053D763E1B002CC607C5A0FE"
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 344606
Connection: keep-alive
Server: AliyunOSS
```

7.1.9 PostObject

PostObject**使用HTML表單上傳檔案到指定bucket**。

Post作為Put的替代品，使得基於瀏覽器上傳檔案到bucket成為可能。Post Object的消息實體通過多重表單格式（multipart/form-data）編碼，在Put Object操作中參數通過HTTP要求標頭傳遞，在Post操作中參數則作為消息實體中的表單域傳遞。

Post object

請求文法

```
POST / HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
User-Agent: browser_data
Content-Length: ContentLength
Content-Type: multipart/form-data; boundary=9431149156168
--9431149156168
Content-Disposition: form-data; name="key"
key
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"
success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
attachment;filename=oss_download.jpg
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-uuid"
myuuid
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-tag"
mytag
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
access-key-id
--9431149156168
Content-Disposition: form-data; name="policy"
encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"
signature
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg
file_content
```

```
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OSS
--9431149156168--
```

表單域

名稱	類型	描述	必須
OSSAccessKeyId	字元串	Bucket 擁有者的 Access Key Id。 預設值：無 限制：當bucket非public-read-write或者提供了policy（或Signature）表單域時，必須提供該表單域。	有條件的
policy	字元串	policy規定了請求的表單域的合法性。不包含policy表單域的請求被認為是匿名請求，並且只能訪問public-read-write的bucket。更詳細描述請參考下文 Post Policy。 預設值：無 限制：當bucket非public-read-write或者提供了OSSAccessKeyId（或Signature）表單域時，必須提供該表單域。	有條件的

名稱	類型	描述	必須
Signature	字元串	根據Access Key Secret和policy計算的簽名資訊，OSS驗證該簽名資訊從而驗證該Post請求的合法性。更詳細描述請參考下文Post Signature。 預設值：無 限制：當bucket非public-read-write或者提供了OSSAccessKeyId（或policy）表單域時，必須提供該表單域。	有條件的
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	字元串	REST要求標頭，更多的資訊見Put Object。 預設值：無	可選
file	字元串	檔案或常值內容，必須是表單中的最後一個域。瀏覽器會自動根據檔案類型來設定Content-Type，會覆蓋用戶的設定。 OSS一次只能上傳一個檔案。 預設值：無	必須

名稱	類型	描述	必須
key	字元串	上傳檔案的object名稱。如果需要使用用戶上傳的檔案名稱作為object名，使用${filename}變數。例如：如果用戶上傳了檔案b.jpg，而key域的值設定為/user/a/${filename}，最終的object名為/user/a/b.jpg。如果檔案名包含路徑，則去除檔案名中的路徑，例如用戶上傳了檔案a/b/c/b.jpg，則取檔案名為b.jpg，若key域的值設定為/user/a/${filename}，最終的object名為/user/a/b.jpg 預設值：無	必須
success_action_redirect	字元串	上傳成功後客戶端跳轉到的URL，如果未指定該表單域，返回結果由success_action_status表單域指定。如果上傳失敗，OSS返回錯誤碼，並不進行跳轉。 預設值：無	可選

名稱	類型	描述	必須
success_action_status	字元串	未指定success_action_redirect表單域時，該表單域指定了上傳成功後返回給客戶端的狀態碼。接受值為200, 201, 204（預設）。 如果該域的值為200或者204，OSS返回一個空文檔和相應的狀態碼。 如果該域的值設定為201，OSS返回一個XML檔案和201狀態碼。 如果其值未設定或者設定成一個非法值，OSS返回一個空文檔和204狀態碼。 預設值：無	
x-oss-meta-*	字元串	用戶指定的user meta值。 OSS不會檢查或者使用該值。 預設值：無	可選
x-oss-server-side-encryption	字元串	指定OSS建立object時的伺服器端加密編碼演算法。 合法值：AES256	可選
x-oss-object-acl	字元串	指定oss建立object時的存取權限。 合法值：public-read，private，public-read-write	可選

名稱	類型	描述	必須
x-oss-security-token	字元串	若本次訪問是使用STS臨時授權方式，則需要指定該項為SecurityToken的值，同時OSSAccessKeyId需要使用與之配對的臨時AccessKeyId，計算簽名時，與使用普通AccessKeyId簽名方式一致。 預設值：無	可選

響應Header

名稱	類型	描述
x-oss-server-side-encryption	字元串	如果請求指定了x-oss-server-side-encryption熵編碼，則響應Header中包含了該頭部，指明了所使用的密碼編譯演算法。

響應元素(Response Elements)

名稱	類型	描述
PostResponse	容器	保持Post請求結果的容器。 子節點：Bucket, ETag, Key, Location
Bucket	字元串	Bucket名稱。 父節點：PostResponse
ETag	字元串	ETag (entity tag) 在每個Object生成的時候被建立，Post請求建立的Object，ETag值是該Object內容的uuid，可以用於檢查該Object內容是否發生變化。 父節點：PostResponse

名稱	類型	描述
Location	字元串	新建立Object的URL。 父節點：PostResponse

細節分析

- 進行Post操作要求對bucket有寫入權限，如果bucket為public-read-write，可以不上傳簽名資訊，否則要求對該操作進行簽名驗證。與Put操作不同，Post操作使用AccessKeySecret對policy進行簽名計算出簽名字元串作為Signature表單域的值，OSS會驗證該值從而判斷簽名的合法性。
- 無論bucket是否為public-read-write，一旦上傳OSSAccessKeyId, policy, Signature表單域中的任意一個，則另兩個表單域為必選項，缺失時OSS會返回錯誤碼：InvalidArgument。
- post操作提交表單編碼必須為“multipart/form-data”，即header中Content-Type為`multipart/form-data;boundary=xxxxxx`這樣的形式，boundary為邊界字元串。
- 提交表單的URL為bucket網域名稱即可，不需要在URL中指定object。即請求行是`POST / HTTP/1.1`，不能寫成`POST /ObjectName HTTP/1.1`。
- policy規定了該次Post請求中表單域的合法值，OSS會根據policy判斷請求的合法性，如果不合法會返回錯誤碼：AccessDenied。在檢查policy合法性時，policy中不涉及的表單域不進行檢查。
- 表單和policy必須使用UTF-8編碼，policy為經過UTF-8編碼和base64編碼的JSON。
- Post請求中可以包含額外的表單域，OSS會根據policy對這些表單域檢查合法性。
- 如果用戶上傳了Content-MD5要求標頭，OSS會計算body的Content-MD5並檢查一致性，如果不一致，將返回InvalidDigest錯誤碼。
- 如果POST請求中包含Header簽名資訊或URL簽名資訊，OSS不會對它們做檢查。
- 如果請求中攜帶以x-oss-meta-為首碼的表單域，則視為user meta，比如x-oss-meta-location。一個Object可以有多個類似的參數，但所有的user meta總大小不能超過8k。
- Post請求的body總長度不允許超過5G。若檔案長度過大，會返回錯誤碼：EntityTooLarge。
- 如果上傳指定了x-oss-server-side-encryption Header請求域，則必須設定其值為AES256，否則會返回400和錯誤碼：InvalidEncryptionAlgorithmError。指定該Header後，在回應標頭中也會返回該Header，OSS會對上傳的Object進行加密編碼儲存，當這個Object被下載時，回應標頭中會包含x-oss-server-side-encryption，值被設定成該Object的密碼編譯演算法。
- 表單域為大小寫不敏感的，但是表單域的值為大小寫敏感的。

樣本

- **請求樣本：**

```
POST / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 344606
Content-Type: multipart/form-data; boundary=9431149156168
--9431149156168
Content-Disposition: form-data; name="key"
/user/a/${filename}
--9431149156168
Content-Disposition: form-data; name="success_action_status"
200
--9431149156168
Content-Disposition: form-data; name="Content-Disposition"
content_disposition
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-uuid"
uuid
--9431149156168
Content-Disposition: form-data; name="x-oss-meta-tag"
metadata
--9431149156168
Content-Disposition: form-data; name="OSSAccessKeyId"
44CF9590006BF252F707
--9431149156168
Content-Disposition: form-data; name="policy"
eyJleHBpcmF0aW9uIjoiMjAxMy0xMi0wMVQxMjowMDowMFoiLCJjb25kaXRp
b25zIjpbWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDAsIDEwNDg1NzYwXSx7
ImJ1Y2tldCI6ImFoYWhhIn0sIHsiQSI6ICJhIn0seyJrZXkiOiAiQUJDIn1dfQ==
--9431149156168
Content-Disposition: form-data; name="Signature"
kZoYNv66bsmc10+dcGKw5x2PRrk=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.
txt"
Content-Type: text/plain
abcdefg
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OSS
--9431149156168--
```

- **返回樣本：**

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

Post Policy

Post請求的policy表單域用於驗證請求的合法性。 policy為一段經過UTF-8和base64編碼的JSON文本，聲明了Post請求必須滿足的條件。雖然對於public-read-write的bucket上傳時，post表單域為可選項，我們強烈建議使用該域來限制Post請求。

policy樣本

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

Post policy中必須包含expiration和condtions。

Expiration

Expiration項指定了policy的過期時間，以ISO8601 GMT時間表示。例如”2014-12-01T12:00:00.000Z”指定了Post請求必鬚髮生在2014年12月1日12點之前。

Conditions

Conditions是一個列表，可以用於指定Post請求的表單域的合法值。注意：表單域對應的值在檢查policy之後進行擴充，因此，policy中設定的表單域的合法值應當對應於擴充之前的表單域的值。Policy中支援的conditions項見下表：

名稱	描述
content-length-range	上傳檔案的最小和最大允許大小。 該condition支援contion-length-range匹配方式。
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	HTTP要求標頭。 該condition支援精確匹配和starts-with匹配方式。
key	上傳檔案的object名稱。 該condition支援精確匹配和starts-with匹配方式。
success_action_redirect	上傳成功後的跳轉URL地址。 該condition支援精確匹配和starts-with匹配方式。
success_action_status	未指定success_action_redirect時，上傳成功後的返回狀態碼。 該condition支援精確匹配和starts-with匹配方式。
x-oss-meta-*	用戶指定的user meta。 該condition支援精確匹配和starts-with匹配方式。

如果Post請求中包含其他的表單域，可以將這些額外的表單網域加入到policy的conditions中，conditions不涉及的表單域將不會進行合法性檢查。

Conditions匹配方式

Conditions匹配方式	描述
精確匹配	表單域的值必須精確匹配conditions中聲明的值。如指定key表單域的值必須為a：{“key”：“a”}同樣可以寫為：[“eq”，“$key”，“a”]
Starts With	表單域的值必須以指定值開始。例如指定key的值必須以/user/user1開始：[“starts-with”，“$key”，“/user/user1”]
指定檔案大小	指定所允許上傳的檔案最大大小和最小大小，例如允許的檔案大小為1到10位元組：[“content-length-range”, 1, 10]

逸出字元

於在 Post policy 中 $ 表示變數，所以如果要描述 $，需要使用逸出字元\$。除此之外，JSON 將對一些字元進行轉義。下圖描述了 Post policy 的 JSON 中需要進行轉義的字元。

逸出字元	描述
\/	斜杠
\	反斜線
\”	雙引號
\$	美元符
\b	空格
\f	換頁
\n	換行
\r	回車
\t	水平定位字元
\uxxxx	Unicode 字元

Post Signature

對於驗證的Post請求，HTML表單中必須包含policy和Signature資訊。policy控制請求中那些值是允許的。計算Signature的具體流程為：

1. 建立一個 UTF-8 編碼的 policy。
2. 將 policy 進行 base64 編碼，其值即為 policy 表單域該填入的值，將該值作為將要簽名的字元串。

3. 使用 AccessKeySecret 對要簽名的字元串進行簽名，簽名方法與Head中籤名的計算方法相同（將要簽名的字元串替換為 policy 即可），請參見在Header中包含簽名。

樣本 Demo

Web 端表單直傳 OSS 樣本 Demo，請參見*JavaScript*客戶端簽名直傳。

7.1.10 Callback

用戶只需要在發送給OSS的請求中攜帶相應的Callback參數，即能實現回調。

現在支援CallBack的API 介面有：PutObject、PostObject、CompleteMultipartUpload。

構造CallBack參數

CallBack參數是由一段經過base64編碼的Json字串，用戶關鍵需要指定請求回調的伺服器URL（callbackUrl）以及回調的內容（callbackBody）。詳細的Json欄位如下：

欄位	含義	選項
callbackUrl	· 檔案上傳成功後OSS向此url發送回調請求，要求方法為POST，body為callbackBody指定的內容。正常情況下，該url需要響應“HTTP/1.1 200 OK”，body必須為JSON格式，回應標頭Content-Length必須為合法的值，且不超過3MB。 · 支援同時配置最多5個url，以”;”分割。OSS會依次發送請求直到第一個返回成功為止。 · 如果沒有配置或者值為空則認為沒有配置callback。 · 支援HTTPS地址。 · 為了保證正確處理中文等情況，callbackUrl需做url編碼處理，比如`http://example.com/中文.php?key=value&中文名稱=中文值` 需要編碼成 `http://example.com/%E4%B8%AD%E6%96%87.php?key=value&%E4%B8%AD%E6%96%87%E5%90%8D%E7%A7%B0=%E4%B8%AD%E6%96%87%E5%80%BC`	必選項
callbackHost	· 發起回調請求時Host頭的值，只有在設定了callbackUrl時才有效。 · 如果沒有配置 callbckHost，則會解析callbackUrl中的url並將解析出的host填充到callbackHost中	可選項

欄位	含義	選項
callbackBody	· 發起回調時請求body的值，例如：key=$(key)&etag=$(etag)&my_var=$(x:my_var)。 · 支援OSS系統變數、自訂變數和常量，支援的系統變數如下表所示 。自訂變數的支援方式在PutObject和CompleteMultipart中是通過callback-var來傳遞，在PostObject中則是將各個變數通過表單域來傳遞。	必選項
callbackBodyType	· 發起回調請求的Content-Type，支援application/x-www-form-urlencoded和application/json，預設為前者。 · 如果為application/x-www-form-urlencoded，則callbackBody中的變數將會被經過url編碼的值替換掉，如果為application/json，則會按照json格式替換其中的變數。	可選項

樣本json串如下

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

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

其中callbackBody中可以設定的系統變數有，其中imageInfo針對於圖片格式，如果為非圖片格式都為空：

系統變數	含義
bucket	bucket
object	object
etag	檔案的etag，即返回給用戶的etag欄位
size	object大小，CompleteMultipartUpload時為整個object的大小
mimeType	資源類型，如jpeg圖片的資源類型為image/jpeg
imageInfo.height	圖片高度
imageInfo.width	圖片寬度
imageInfo.format	圖片格式，如jpg、png等

自訂參數

用戶可以通過callback-var參數來配置自訂參數。

自訂參數是一個Key-Value的Map，用戶可以配置自己需要的參數到這個Map。在OSS發起POST回調請求的時候，會將這些參數和上一節所述的系統參數一起放在POST請求的body中以方便接收回調方獲取。

構造自訂參數的方法和callback參數的方法是一樣的，也是以json格式來傳遞。該json字元串就是一個包含所有自訂參數的Key-Value的Map。

说明:

用戶自訂參數的Key一定要以x:開頭，且必須為小寫。否則OSS會返回錯誤。

假定用戶需要設定兩個自訂的參數分別為x:var1和x:var2，對應的值分別為value1和value2，那麼構造出來的json格式如下：

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

構造回調請求

構造完成上述的callback和callback-var兩個參數之後，一共有三種方式傳給OSS。其中callback為必填參數，callback-var為選擇性參數，如果沒有自訂參數的話可以不用添加callback-var欄位。這三種方式如下：

- 在URL中攜帶參數。

- 在Header中攜帶參數。
- 在POST請求的body中使用表單域來攜帶參數。

> 说明:
> 在使用POST請求上傳Object的時候只能使用這種方式來指定回調參數。

這三種方式只能同時使用其中一種，否則OSS會返回InvalidArgument錯誤。

要將參數附加到OSS的請求中，首先要將上文構造的json字元串使用base64編碼，然後按照如下的方法附加到OSS的請求中：

- 如果在URL中攜帶參數。把`callback=[CallBack]`或者`callback-var=[CallBackVar]`作為一個url參數帶入請求發送。計算簽名CanonicalizedResource時，將callback或者callback-var當做一個sub-resource計算在內
- 如果在Header中攜帶參數。把`x-oss-callback=[CallBack]`或者`x-oss-callback-var=[CallBackVar]`作為一個head帶入請求發送。在計算簽名CanonicalizedOSSHeaders時，將x-oss-callback-var和x-oss-callback計算在內。如下樣本：

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

- 如果需要在POST上傳Object的時候附帶回調參數會稍微複雜一點，callback參數要使用獨立的表單域來附加，如下面的樣本：

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

```
ImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVy
bGVuY29kZWQifQ==
```

如果擁有自訂參數的話，不能直接將callback-var參數直接附加到表單域中，每個自訂的參數都需要使用獨立的表單域來附加，舉個例子，如果用戶的自訂參數的json為

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

那麼POST請求的表單域應該如下：

```
--9431149156168
Content-Disposition: form-data; name="callback"
eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5w
aHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tC
b2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIs
ImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVy
bGVuY29kZWQifQ==
--9431149156168
Content-Disposition: form-data; name="x:var1"
value1
--9431149156168
Content-Disposition: form-data; name="x:var2"
value2
```

同時可以在policy中添加callback條件（如果不添加callback，則不對該參數做上傳驗證）如：

```
{ "expiration": "2014-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith" },
    {"callback": "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYW
xsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2
FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJjYWxsYmFja0
JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkIn0=",
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

發起回調請求

如果檔案上傳成功，OSS會根據用戶的請求中的callback參數和自訂參數（callback-var參數），將特定內容以POST方式發送給應用伺服器。

```
POST /index.html HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 181
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
```

```
bucket=callback-test&object=test.txt&etag=D8E8FCA2DC0F896FD7CB
4CB0031BA249&size=5&mimeType=text%2Fplain&imageInfo.height=&imageInfo.
width=&imageInfo.format=&x:var1=for-callback-test
```

返回回調結果

比如應用伺服器端返回的回應請求為：

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

返回上傳結果

再給客戶端的內容為：

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

需要注意的是，如果類似CompleteMultipartUpload這樣的請求，在返回請求本身body中存在內容（如XML格式的資訊），使用上傳回調功能後會覆蓋原有的body的內容如`{"a":"b"}`，希望對此處做好判斷處理。

回調簽名

使用者佈建callback參數後，OSS將按照使用者佈建的callbackUrl發送POST回調請求給用戶的應用伺服器。應用伺服器收到回調請求之後，如果希望驗證回調請求確實是由OSS發起的話，那麼可以通過在回調中帶上簽名來驗證OSS的身份。

- **生成簽名**

  簽名在OSS端發生，採用RSA非對稱方式簽名，私密金鑰加密的過程為：

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

  

  说明:

其中private_key為私密金鑰，只有oss知曉，path為回調請求的資源路徑，query_string為查詢字元串，body為回調的消息體，所以簽名過程由以下幾步組成：

- **獲取待簽名字元串：資源路徑經過url解碼後，加上原始的查詢字元串，加上一個回車符，加上回調消息體**
- **RSA簽名：使用秘鑰對待簽名字元串進行簽名，簽名的hash函數為md5**
- **將簽名後的結果做base64編碼，得到最終的簽名，簽名放在回調請求的authorization頭中**

如下例：

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

path為`/index.php`**，query_string為**`?id=1&index=2`**，body為**`bucket=yonghu-test`**，最終簽名結果為**`kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==`

- **驗證簽名**

  驗證簽名的過程即為簽名的逆過程，由應用伺服器驗證，過程如下：

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

  欄位的含義與簽名過程中描述相同，其中public_key為公開金鑰， authorization為回調頭中的簽名，整個驗證簽名的過程分為以下幾步：

  1. **回調請求的x-oss-pub-key-url頭保存的是公開金鑰的url地址的base64編碼，因此需要對其做base64解碼後獲取到公開金鑰，即**

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

     這裡需要注意，用戶需要校驗`x-oss-pub-key-url`**頭的值必須以**`http://gosspublic.alicdn.com/`**或者**`https://gosspublic.alicdn.com/`**開頭，目的是為了保證這個publickey是由OSS頒發的。**

  2. **獲取base64解碼後的簽名**

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

  3. **獲取待簽名字元串，方法與簽名一致**

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

  4. **驗證簽名**

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

  以上例為例：

  1. **獲取到公開金鑰的url地址，即**`aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==`**經過base64解碼後得到**`http://gosspublic.alicdn.com/callback_pub_key_v1.pem`
  2. **簽名頭**`kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==`**做base64解碼（由於為非列印字元，無法顯示出解碼後的結果）**
  3. **獲取待簽名字元串，即url_decode(“index.php”)+“?id=1&index=2”+“\n”+“bucket=yonghu-test”，並做md5**
  4. **驗證簽名**

- **應用伺服器樣本**

  以下為一段python樣本，示範了一個簡單的應用伺服器，主要是說明驗證簽名的方法，此樣本需要安裝M2Crypto庫

```
import httplib
import base64
import md5
import urllib2
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
from M2Crypto import RSA
from M2Crypto import BIO
def get_local_ip():
    try:
        csock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        csock.connect(('8.8.8.8', 80))
        (addr, port) = csock.getsockname()
        csock.close()
        return addr
    except socket.error:
        return ""
class MyHTTPRequestHandler(BaseHTTPRequestHandler):
    '''
    def log_message(self, format, *args):
        return
    '''
    def do_POST(self):
        #get public key
        pub_key_url = ''
        try:
            pub_key_url_base64 = self.headers['x-oss-pub-key-url']
            pub_key_url = pub_key_url_base64.decode('base64')
            if not pub_key_url.startswith("http://gosspublic.alicdn.
com/") and not pub_key_url.startswith("https://gosspublic.alicdn.com
/"):
                self.send_response(400)
                self.end_headers()
                return
            url_reader = urllib2.urlopen(pub_key_url)
            #you can cache it
            pub_key = url_reader.read()
        except:
            print 'pub_key_url : ' + pub_key_url
            print 'Get pub key failed!'
            self.send_response(400)
            self.end_headers()
            return
        #get authorization
        authorization_base64 = self.headers['authorization']
        authorization = authorization_base64.decode('base64')
        #get callback body
        content_length = self.headers['content-length']
        callback_body = self.rfile.read(int(content_length))
        #compose authorization string
        auth_str = ''
        pos = self.path.find('?')
        if -1 == pos:
            auth_str = urllib2.unquote(self.path) + '\n' +
callback_body
        else:
            auth_str = urllib2.unquote(self.path[0:pos]) + self.path
[pos:] + '\n' + callback_body
```

```
        print auth_str
        #verify authorization
        auth_md5 = md5.new(auth_str).digest()
        bio = BIO.MemoryBuffer(pub_key)
        rsa_pub = RSA.load_pub_key_bio(bio)
        try:
            result = rsa_pub.verify(auth_md5, authorization, 'md5')
        except:
            result = False
        if not result:
            print 'Authorization verify failed!'
            print 'Public key : %s' % (pub_key)
            print 'Auth string : %s' % (auth_str)
            self.send_response(400)
            self.end_headers()
            return
        #do something accoding to callback_body
        #response to OSS
        resp_body = '{"Status":"OK"}'
        self.send_response(200)
        self.send_header('Content-Type', 'application/json')
        self.send_header('Content-Length', str(len(resp_body)))
        self.end_headers()
        self.wfile.write(resp_body)
class MyHTTPServer(HTTPServer):
    def __init__(self, host, port):
        HTTPServer.__init__(self, (host, port), MyHTTPRequestHandler
)
if '__main__' == __name__:
    server_ip = get_local_ip()
server_port = 23451
server = MyHTTPServer(server_ip, server_port)
```

```
server.serve_forever()
```

其它語言實現的應用伺服器如下：

Java版本：

- 下載地址：點擊這裡
- 運行方法：解壓包運行`java -jar oss-callback-server-demo.jar 9000`（9000就啟動並執行通信埠，可以自己指定）

PHP版本：

- 下載地址：點擊這裡
- 運行方法：部署到Apache環境下，因為PHP本身語言的特點，取一些數據頭部會依賴於環境。所以可以參考例子根據所在環境修改。

Python版本：

- 下載地址：點擊這裡
- 運行方法：解壓包直接運行`python callback_app_server.py`，運行該程式需要安裝rsa的依賴。

C#版本：

- 下載地址：點擊這裡
- 運行方法：解壓後參看`README.md`。

Go版本：

- 下載地址：點擊這裡
- 運行方法：解壓後參看`README.md`。

Go版本：

- 下載地址：點擊這裡
- 運行方法：解壓後參看`README.md`。

Ruby版本：

- 下載地址：點擊這裡
- 運行方法： ruby aliyun_oss_callback_server.rb

特別須知

- 如果傳入的callback或者callback-var不合法，則會返回400錯誤，錯誤碼為”InvalidArgument”，不合法的情況包括以下幾類：
  - PutObject()和CompleteMultipartUpload()介面中url和header同時傳入callback(x-oss-callback)或者callback-var(x-oss-callback-var)參數
  - callback或者callback-var(PostObject()由於沒有callback-var參數，因此沒有此限制，下同)參數過長（超過5KB）
  - callback或者callback-var沒有經過base64編碼
  - callback或者callback-var經過base64解碼後不是合法的json格式
  - callback參數解析後callbackUrl欄位包含的url超過限制（5個），或者url中傳入的port不合法，比如

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

  - callback參數解析後callbackBody欄位為空
  - callback參數解析後callbackBodyType欄位的值不是”application/x-www-form-urlencoded”或者”application/json”
  - callback參數解析後callbackBody欄位中變數的格式不合法，合法的格式為${var}
  - callback-var參數解析後不是預期的json格式，預期的格式應該為`{"x:var1":"value1","x:var2":"value2"...}`
- 如果回調失敗，則返回203，錯誤碼為”CallbackFailed”，回調失敗只是表示OSS沒有收到預期的回調響應，不代表應用伺服器沒有收到回調請求（比如應用伺服器返回的內容不是json格式），另外，此時檔案已經成功上傳到了OSS
- 應用伺服器返回OSS的響應必須帶有Content-Length的Header，Body大小不要超過1MB。

Callback支援的地域

Callback目前支援的地域如下：華北 2（北京）、華東 1（杭州）、華北 1（青島）、華東 2（上海）、上海金融雲、華南 1（深圳）、香港、華北 5（呼和浩特）、華北 3（張家口）、中東東部 1（迪拜）、亞太東北 1（日本）、歐洲中部 1（法蘭克福）、亞太東南 1（新加坡）、美國東部 1（維吉尼亞）、美國西部 1（矽谷）、亞太東南 2 （悉尼）以及亞太東南 3（吉隆坡）。

7.1.11 RestoreObject

`RestoreObject`介面用於服務端執行解凍任務。

只針對歸檔類型的Object讀取，需要調用RestoreObject介面讓服務端執行解凍任務。如果一個Object是標準或者低頻訪問類型，不要調用該介面。

歸檔類型Object在執行Restore前後的狀態變換過程如下：

1. 一個歸檔類型的Object初始時處於冷凍狀態。
2. 提交一次Restore操作後，Object將處於解凍中的狀態，服務端執行解凍。
3. 待服務端執行完成解凍任務後，Object進入解凍狀態，此時用戶可以讀取Object。
4. 解凍狀態預設持續1天，24小時內再次調用RestoreObject介面則解凍狀態會自動延長24小時，最多可延長7天，之後，Object又回到初始時的冷凍狀態。

狀態變換過程中產生的相關費用如下：

- 對一個處於冷凍狀態的Object執行Restore操作，會產生數據取回費用。
- 解凍狀態最多延長7天。在此期間內不再重複收取數據取回費用。
- 解凍狀態結束後，Object又回到冷凍狀態，再次解凍的首次讀取數據會收取數據取回費用。

請求文法

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

細節分析

- 如果是針對該Object第一次調用RestoreObject介面，則返回202。
- 如果已經成功調用過RestoreObject介面，且服務端仍處於解凍中，再次調用時返回409，錯誤碼為：RestoreAlreadyInProgress。服務端返回該錯誤，代表格服務端正在執行restore操作，用戶只需要等待作業完成，最長等待時間4小時。
- 如果已經成功調用過RestoreObject介面，且服務端解凍已經完成，再次調用時返回200，且會將object的可下載時間延長一天，最多延長7天。
- 如果object不存在，則返回404。
- 如果針對非歸檔類型的Object提交restore，則返回400，錯誤碼為：OperationNotSupported。

樣本

首次提交restore的請求樣本：

```
POST /oss.jpg?restore HTTP/1.1
Host: oss-archive-example.oss-cn-hangzhou.aliyuncs.com
Date: Sat, 15 Apr 2017 07:45:28 GMT
Authorization: OSS e1Unnbm1rgdnpI:y4eyu+4yje5ioRCr5PB=
```

返回樣本

```
HTTP/1.1 202 Accepted
Date: Sat, 15 Apr 2017 07:45:28 GMT
```

```
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
```

再次調用，且restore沒有完成時：

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

返回樣本

```
HTTP/1.1 409 Conflict
Date: Sat, 15 Apr 2017 07:45:29 GMT
Content-Length: 556
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>RestoreAlreadyInProgress</Code>
  <Message>The restore operation is in progress.</Message>
  <RequestId>58EAF141461FB42C2B000008</RequestId>
  <HostId>10.101.200.203</HostId>
</Error>
```

再次調用，且restore已經完成時：

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

返回樣本

```
HTTP/1.1 200 Ok
Date: Sat, 15 Apr 2017 07:45:30 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
```

7.1.12 SelectObject（公測）

功能介紹

對象儲存（Object Storage Service，簡稱OSS）是基於阿里雲飛天分布式系統的海量、安全和高可靠的雲端儲存體服務，是一種面向互連網的大規模、低成本、通用儲存，提供RESTful API，具備容量和處理的彈性擴充能力。OSS不僅非常適合儲存海量的媒體檔案，也適合作為資料倉儲儲存海量的資料檔案。目前Hadoop 3.0已經支援OSS，在EMR上運行Spark/Hive/Presto等服務以及阿里自研的MaxCompute、HybridDB以及新上線的Data Lake Analytics都支援從OSS直接處理數據。

然而，目前OSS提供的GetObject介面決定了大數據平台只能把OSS數據全部下載到本地然後進行分析過濾，在很多查詢場景下浪費了大量頻寬和客戶端資源。

SelectObject介面是對上述問題的解決方案。其核心思想是大數據平台將條件、Projection下推到OSS層，讓OSS做基本的過濾，從而只返回有用的數據。客戶端一方面可以減少網路頻寬，另一方面也減少了數據的處理量，從而節省了CPU和記憶體用來做其他更多的事情。這使得基於OSS的資料倉儲、資料分析成為一種更有吸引力的選擇。

SelectObject現在處於公測階段，提供了Java、Python 的SDK。目前支援RFC 4180標準的CSV（包括TSV等類CSV檔案，檔案的行資料行分隔符號以及Quote字元都可自訂），且檔案編碼為UTF-8。支援標準儲存類型和低頻訪問儲存類型的檔案。支援加密檔案（OSS完全託管、KMS加密-預設KMS主要金鑰）。

支援的SQL文法如下：

- SQL 陳述式： Select From Where
- 資料類型：String, Int(64bit), float(64bit), Timestamp, Boolean
- 操作： 邏輯條件（AND,OR,NOT)， 算術運算式（+-*/%)， 比較操作(>,=, <, >=, <=, !=), String 操作 (LIKE, ||)

和GetObject提供了基於Byte的分區下載類似，SelectObject也提供了分區查詢的機制，包括兩種分區方式：按行分區和按Split分區。按行分區是常用的分區方式，然而對於稀疏數據來說，按行分區可能會導致分區時負載不均衡。Split是OSS用於分區的一個概念，一個Split包含多行數據，每個Split的數據大小大致相等，相對按行來，按Spit是更加高效的分區方式。尤其是對於CSV數據來說，基於Byte的分區可能會將數據破壞，因此按Spit分區更加合適。

關於資料類型，OSS中的CSV數據預設都是String類型，用戶可以使用CAST函數實現數據轉換，比如下面的SQL查詢將_1和_2轉換為int後進行比較。

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

同時，對於SelectObject支援在Where條件中進行隱式的轉換，比如下面的語句中第一列和第二列將被轉換成int：

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

RESTful API使用說明

對目標CSV檔案執行SQL語句，返回執行結果。同時該命令會自動儲存CSV檔案的metadata資訊，比如總的行數和列數等。

正確執行時，該API返回206。如果SQL語句不正確，或者和CSV檔案不匹配，則會返回400錯誤。

請求文法

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

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

名稱	類型	描述
SelectRequest	**容器**	**保存Select請求的容器** **子節點：Expression, InputSerialization, OutputSerialization** **父節點：None**
Expression	**字元串**	**以Base 64 編碼的SQL語句** **子節點：None** **父節點：SelectRequest**

名稱	類型	描述
InputSerialization	容器	輸入序列化參數（可選） 子節點:CompressionType, CSV 父節點：SelectRequest
OutputSerialization	容器	輸出序列化參數（可選） 子節點：CSV，OutputRawData 父節點：SelectRequest
CSV（InputSerialization）	容器	輸入CSV的格式參數（可選） 子節點：FileHeaderInfo,RecordDelimiter,FieldDelimiter,QuoteCharacter, CommentCharacter, Range 父節點：InputSerialization
CSV(OutputSerialization)	容器	輸出CSV的格式參數（可選） 子節點： RecordDelimiter, FieldDelimiter,KeepAllColumns 父節點：OutputSerialization

名稱	類型	描述
OutputRawData	bool，預設false	指定輸出數據為純數據（不是下面提到的基於Frame格式）（可選） 子節點：None 父節點：OutputSerialization
CompressionType	枚舉	指定檔案壓縮類型。目前不支援任何壓縮，故只能為None 子節點: None 父節點：InputSerialization
FileHeaderInfo	枚舉	指定CSV檔案頭資訊（可選） 取值： · Use：該CSV檔案有頭資訊，可以用CSV列名作為Select裡的列名。 · Ignore：該CSV檔案有頭資訊，但不可用CSV列名作為Select裡的列名。 · None：該檔案沒有頭資訊，為預設值。 子節點：None 父節點：CSV （輸入）

名稱	類型	描述
RecordDelimiter	字元串	指定CSV分行符號，以Base64編碼。預設值為\n（可選）。 未編碼前的值最多為兩個字元，以字元的ANSI值表示，比如在Java裡用\n表示換行。 子節點：None 父節點：CSV （輸入、輸出）
FieldDelimiter	字元串	指定CSV資料行分隔符號，以Base64編碼。預設值為，（可選） 未編碼前的值必須為一個字元，以字元的ANSI值表示，比如Java裡用，表示逗號。 子節點：None 父節點：CSV （輸入，輸出）
QuoteCharacter	字元串	指定CSV的引號字元，以Base64編碼。預設值為\”（可選）。在CSV中引號內的分行符號，資料行分隔符號將被視作一般字元。為編碼前的值必須為一個字元，以字元的ANSI值表示，比如Java裡用\”表示引號。 子節點：None 父節點：CSV （輸入）
CommentCharacter	字元串	指定CSV的注釋符，以Base464編碼。預設值為#（可選）

名稱	類型	描述
Range	字元串	指定查詢檔案的範圍（可選）。支援兩種格式： · 按行查詢：line-range=start-end · 按Split查詢：split-range=start-end 其中start和end均為inclusive。其格式和range get中的range參數一致。 子節點：None 父節點：CSV （輸入）
KeepAllColumns	bool	指定返回結果中包含CSV所有列的位置（可選，預設值為false）。但僅僅在select 語句裡出現的列會有值，不出現的列則為空，返回結果中每一行的數據按照CSV列的順序從低到高排列。比如下面語句： `select _5, _1 from ossobject.` 如果KeepAllColumn = true，假設一共有6列數據，則返回的數據如下： Value of 1st column,,,, Value of 5th column,\n 子節點：None 父節點：CSV（輸出）

返回結果

請求結果以一個個Frame形式返回。每個Frame的格式如下,其中checksum均為CRC32：

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

<---4 bytes--><---4 bytes----------><-------4 bytes-------><variable><----4bytes---------->

一共有三種不同的Frame Type， 列舉如下：

名稱	Frame-Type值	Payload格式	描述
Data Frame	version | 8388609 <--1 byte><--3 bytes ->	scanned size ｜ data <-8 bytes----------><---variable-> 其中scanned size為目前已掃描過的數據大小，data為查詢返回的數據。	Data Frame用以返回查詢數據，並同時可以彙報當前的進展。
Continuous Frame	version ｜8388612 <--1 byte><--3 bytes ->	scanned size <----8 bytes-->	Continuous Frame用以彙報當前進展以及維持http串連。如果該查詢在5s內未返回數據則會返回一個Continuous Frame。

名稱	Frame-Type值	Payload格式	描述
End Frame	version | 8388613	Offset | total scanned bytes | http status code | error message <--8bytes-><--8bytes----------><----4 bytes--------><-variable------> 其中offset為掃描後最終的位置位移，total scanned bytes為最終掃描過的數據大小。http status code為最終的處理結果，error message為錯誤資訊。	這裡返回status code的原因在於SelectObject為串流，因而在發送Response Header的時候僅僅處理了第一個Block。如果第一個Block數據和SQL是匹配的，則在Response Header中的Status為206，但如果下面的數據非法，我們已無法更改Header中的Status，只能在End Frame裡包含最終的Status及其出錯資訊。因此客戶端應該視其為最終狀態。

樣例請求

```
POST /oss-select/bigcsv_normal.csv?x-oss-process=csv%2Fselect HTTP/1.1
Date: Fri, 25 May 2018 22:11:39 GMT
Content-Type:
Authorization: OSS LTAIJPXxMLocA0fD:FC/9JRbBGRw4o2QqdaL246Pxuvk=
User-Agent: aliyun-sdk-dotnet/2.8.0.0(windows 16.7/16.7.0.0/x86;4.0.
30319.42000)
Content-Length: 748
Expect: 100-continue
Connection: keep-alive
Host: host name

<?xml version="1.0"?>
<SelectRequest>
 <Expression>c2VsZWN0IGNvdW50KCopIGZyb20gb3Nzb2JqZWN0IHdoZXJlIF
80ID4gNDU=
 </Expression>
 <InputSerialization>
  <Compression>None</Compression>
  <CSV>
   <FileHeaderInfo>Ignore</FileHeaderInfo>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
   <Comments>Iw==</Comments>
  </CSV>
 </InputSerialization>
```

```
 <OutputSerialization>
  <CSV>
   <RecordDelimiter>Cg==</RecordDelimiter>
   <FieldDelimiter>LA==</FieldDelimiter>
   <QuoteCharacter>Ig==</QuoteCharacter>
   <KeepAllColumns>false</KeepAllColumns>
 </CSV>
 <OutputRawData>false</OutputRawData>
 </OutputSerialization>
</SelectRequest>
```

SQL語句Regex

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

其中SELECT, OSSOBJECT以及 WHERE為關鍵字不得更改。

```
select_list: column name

| column index (比如_1, _2)

| function(column index | column name)

| select_list AS alias
```

支援的function為AVG,SUM,MAX,MIN,COUNT, CAST(類型轉換函式)。其中COUNT後只能用*。

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

op：包括 > < >= <= != =, LIKE， +-*/%以及字元串串連||。

cast: 對於同一個column，只能cast成一種類型。

```
limit_opt:
```

| limit 整數

彙總和Limit的混用

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

對於上面的語句，其含義是指在前100行中計算第一列的AVG值。這個行為和MY SQL不同，原因是在 SelectObject中彙總永遠只返回一行數據，因而對彙總來說限制其輸出規模是多餘的。因此SelectObject裡limit 將先於彙總函式執行。

SQL語句限制

- 目前僅僅支援UTF-8編碼的文字檔。GZIP壓縮過的文本將在以後版本中支援，目前只能處理未壓縮檔。
- 僅支援單檔案查詢，不支援join, order by, group by, having
- Where語句裡不能包含彙總條件(e.g. where max(cast(age as int)) > 100這個是不允許的)。
- 支援的最大的列數是1000，SQL中最大的列名稱為1024。
- 在LIKE語句中，支援最多5個%萬用字元。*和%是等價的，表示0或多個任一字元。
- 在IN語句中，最多支援1024個常量項。
- Select後的Projection可以是列名，列索引(_1, _2等)，或者是彙總函式，或者是CAST函數；不支援其他運算式。 比如select _1 + _2 from ossobject是不允許的。
- 支援的最大行及最大列長度是都是256K。

CREATE SELECT OBJECT META

Create Select Object Meta API作用獲得目標CSV檔案的總的行數，總的列個數，以及Splits個數。如果該資訊不存在，則會掃描整個檔案分析並記錄下CSV檔案的上述資訊。如果該API執行正確，返回200。否則如果目標CSV檔案為非法、或者指定的分隔符號和目標CSV不匹配，則返回400。

請求元素

名稱	類型	描述
CsvMetaRequest	容器	保存建立Select Meta請求的容器。 子節點：Expression, InputSerialization, OutputSerialization 父節點：None

名稱	類型	描述
InputSerialization	容器	輸入序列化參數（可選） 子節點:CompressionType, CSV 父節點：CsvMetaRequest
OverwriteIfExists	bool	重新計算SelectMeta，覆蓋已有數據。（可選，預設是false，即如果Select Meta已存在則直接返回） 子節點: None 父節點：CsvMetaRequest
CompressionType	枚舉	指定檔案壓縮類型。目前不支援任何壓縮，故只能為None 子節點: None 父節點：InputSerialization
RecordDelimiter	字元串	指定CSV分行符號，以Base64編碼。預設值為’\n’（可選）。未編碼前的值最多為兩個字元，以字元的ANSI值表示，比如在Java裡用‘\n’表示換行。 子節點：None 父節點：CSV

名稱	類型	描述
FieldDelimiter	字元串	指定CSV資料行分隔符號，以Base64編碼。預設值為，（可選） 未編碼前的值必須為一個字元，以字元的ANSI值表示，比如Java裡用，表示逗號。 子節點：None 父節點：CSV（輸入，輸出）
QuoteCharacter	字元串	指定CSV的引號字元，以Base64編碼。預設值為\”（可選）。在CSV中引號內的分行符號，資料行分隔符號將被視作一般字元。為編碼前的值必須為一個字元，以字元的ANSI值表示，比如Java裡用\”表示引號。 子節點：None 父節點：CSV（輸入）
CSV	容器	指定CSV輸入格式 子節點：RecordDelimiter，FieldDelimiter，QuoteCharacter 父節點：InputSerialization

Response Body：空

Response Header：

- x-oss-select-csv-lines： 總行數
- x-oss-select-csv-columns： 總列數
- x-oss-select-csv-splits： 總Splits數

- **content-length：檔案內容length**

> **说明:**
> **x-oss-select-csv-columns是指第一行的列數，假設用戶第一行的數據是正確的。**

樣例請求

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

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

返迴響應

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 25 May 2018 23:06:42 GMT
Content-Type: application/vnd.ms-excel
Content-Length: 0
Connection: close
x-oss-request-id: 5B089702461FB4C07B000C75
x-oss-location: oss-cn-hangzhou-a
x-oss-access-id: LTAIJPXxMLocA0fD
x-oss-sign-type: NormalSign
x-oss-object-name: bigcsv_normal.csv
Accept-Ranges: bytes
ETag: "3E1372A912B4BC86E8A51234AEC0CA0C-400"
Last-Modified: Wed, 09 May 2018 00:22:32 GMT
x-oss-object-type: Multipart
x-oss-bucket-storage-type: standard
x-oss-hash-crc64ecma: 741622077104416154
x-oss-storage-class: Standard
**x-oss-select-csv-rows: 54000049**
**x-oss-select-csv-columns: 4**
**x-oss-select-csv-splits: 960**
```

Python SDK 樣例

```
import os
import oss2
```

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

# 首先初始化AccessKeyId、AccessKeySecret、Endpoint等資訊。
# 通過環境變數獲取，或者把諸如“<yourAccessKeyId>”替換成真實的AccessKeyId等。
#
# 以杭州區域為例，Endpoint可以是：
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com

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

# 建立儲存空間實例，所有檔案相關的方法都需要通過儲存空間實例來調用。
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
key =  'python_select.csv'
content =  'Tom Hanks,USA,45\r\n'*1024
filename =  'python_select.csv'
# 上傳檔案
bucket.put_object(key, content)
csv_meta_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n'}
select_csv_params = {'CsvHeaderInfo': 'None',
'RecordDelimiter': '\r\n',
'LineRange': (500, 1000)}

csv_header = bucket.create_select_object_meta(key, csv_meta_params)
print(csv_header.csv_rows)
print(csv_header.csv_splits)
result = bucket.select_object(key, "select * from ossobject where _3
 > 44 limit 100000", select_call_back, select_csv_params)
content_got =  b''
for chunk in result:
 content_got += chunk
print(content_got)

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

bucket.delete_object(key)
```

Java SDK 樣例

```
package samples;

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

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

/**
```

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

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

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

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

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

```
}
```

最佳實踐

當一個檔案很大時，要有效實現分區查詢，推薦的流程如下：

1. **調用Create Select Object Meta API獲得該檔案的總的Split數。理想情況下如果該檔案需要用SelectObject，則該API最好在查詢前進行非同步呼叫，這樣可以節省掃描時間。**
2. **根據客戶端資源情況選擇合適的並發度n，用總的Split數除以並發度n得到每個分區查詢應該包含的Split個數。**
3. **在請求Body中用諸如split-range=1-20的形式進行分區查詢。**
4. **如果需要最後可以合并結果。**

SelectObject和Normal類型檔案配合性能更佳。Multipart 以及Appendable類型的檔案由於其內部結構差異導致性能較差。

7.2 分區上傳（MulitipartUpload）

7.2.1 InitiateMultipartUpload

使用Multipart Upload模式傳輸數據前，必須先調用該介面來通知OSS初始化一個Multipart Upload事件。

該介面會返回一個OSS伺服器建立的全域唯一的Upload ID，用於標識本次Multipart Upload事件。用戶可以根據這個ID來發起相關的操作，如中止Multipart Upload、查詢Multipart Upload等。

請求文法

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

請求參數(Request Parameters)

Initiate Multipart Upload時，可以通過encoding-type對返回結果中的Key進行編碼。

名稱	類型	描述
encoding-type	字元串	指定對返回的Key進行編碼，目前支援url編碼。Key使用UTF-8字元，但xml 1.0標準不支援解析一些控制字元，比如ascii值從0到10的字元。對於Key中包含xml 1.0標準不支援的控制字元，可以通過指定encoding-type對返回的Key進行編碼。 預設值：無 可選值：url

請求Header

名稱	類型	描述
Cache-Control	字元串	指定該Object被下載時的網頁的緩存行為；更詳細描述請參照*RFC2616*。 預設值：無
Content-Disposition	字元串	指定該Object被下載時的名稱；更詳細描述請參照*RFC2616*。 預設值：無
Content-Encoding	字元串	指定該Object被下載時的內容編碼格式；更詳細描述請參照*RFC2616*。 預設值：無
Expires	整數	過期時間（milliseconds）；更詳細描述請參照*RFC2616*。 預設值：無

名稱	類型	描述
x-oss-server-side-encryption	字元串	指定上傳該Object每個part時使用的伺服器端加密編碼演算法，OSS會對上傳的每個part採用伺服器端加密編碼進行儲存。 合法值：AES256 或 KMS 注意：用戶需要在控制台開通KMS（金鑰管理服務），才可使用KMS密碼編譯演算法，否則會報KmsServiceNotEnabled錯誤碼。

響應元素(Response Elements)

名稱	類型	描述
Bucket	字元串	初始化一個Multipart Upload事件的Bucket名稱。 父節點：InitiateMultipartUploadResult
InitiateMultipartUploadResult	容器	保存Initiate Multipart Upload請求結果的容器。 子節點：Bucket, Key, UploadId 父節點：None
Key	字元串	初始化一個Multipart Upload事件的Object名稱。 父節點：InitiateMultipartUploadResult
UploadId	字元串	唯一標示此次Multipart Upload事件的ID。 父節點：InitiateMultipartUploadResult

名稱	類型	描述
EncodingType	字元串	指明返回結果中編碼使用的類型。如果請求的參數中指定了encoding-type，那返回的結果會對Key進行編碼。 父節點：容器

細節分析

- 該操作計算認證簽名的時候，需要加“?uploads”到CanonicalizedResource中。
- 初始化Multipart Upload請求，支援如下標準的HTTP要求標頭：Cache-Control，Content- Disposition，Content-Encoding，Content-Type，Expires，以及以“x-oss-meta-”開頭的用戶自訂Headers。具體含義請參見PUT Object介面。
- 初始化Multipart Upload請求，並不會影響已經存在的同名object。
- 伺服器收到初始化Multipart Upload請求後，會返回一個XML格式的請求體。該請求體內有三個元素：Bucket，Key和UploadID。請記錄下其中的UploadID，以用於後續的Multipart相關操作。
- 初始化Multipart Upload請求時，若設定了x-oss-server-side-encryption Header，則在回應標頭中會返回該Header，並且在上傳的每個part時，服務端會自動對每個part進行熵編碼加密儲存，目前OSS伺服器端只支援AES256和KMS加密，指定其他值會返回400和相應的錯誤提示：InvalidEncryptionAlgorithmError；在上傳每個part時不必再添加x-oss-server-side-encryption 要求標頭，若指定該要求標頭則OSS會返回400和相應的錯誤提示：InvalidArgument。

樣本

請求樣本：

```
POST /multipart.data?uploads HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:/cluRFtRwMTZpC2hTj4F6
7AGdM4=
```

返回樣本：

```
HTTP/1.1 200 OK
Content-Length: 230
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 42c25703-7503-fbd8-670a-bda01eaec618
Date: Wed, 22 Feb 2012 08:32:21 GMT
Content-Type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
```

```
<InitiateMultipartUploadResult xmlns=”http://doc.oss-cn-hangzhou.
aliyuncs.com”>
    <Bucket> multipart_upload</Bucket>
    <Key>multipart.data</Key>
    <UploadId>0004B9894A22E5B1888A1E29F8236E2D</UploadId>
</InitiateMultipartUploadResult>
```

7.2.2 UploadPart

初始化一個Multipart Upload之後，可以根據指定的Object名和Upload ID來分塊（Part）上傳數據。每一個上傳的Part都有一個標識它的號碼（part number，範圍是1~10,000）。

對於同一個Upload ID，該號碼不但唯一標識這一塊數據，也標識了這塊數據在整個檔案內的相對位置。如果你用同一個part號碼，上傳了新的數據，那麼OSS上已有的這個號碼的Part數據將被覆蓋。除了最後一塊Part以外，其他的part最小為100KB；最後一塊Part沒有大小限制。

請求文法

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

細節分析

- **調用該介面上傳Part數據前，必須調用Initiate Multipart Upload介面，獲取一個OSS伺服器頒發的Upload ID。**
- **Multipart Upload要求除最後一個Part以外，其他的Part大小都要大於100KB。但是Upload Part介面並不會立即校驗上傳Part的大小（因為不知道是否為最後一塊）；只有當Complete Multipart Upload的時候才會校驗。**
- **OSS會將伺服器端收到Part數據的MD5值放在ETag頭內返回給用戶。**
- **Part號碼的範圍是1~10000。如果超出這個範圍，OSS將返回InvalidArgument的錯誤碼。**
- **若調用Initiate Multipart Upload介面時，指定了x-oss-server-side-encryption要求標頭，則會對上傳的Part進行加密編碼，並在Upload Part回應標頭中返回x-oss-server-side-encryption頭，其值表明該Part的伺服器端密碼編譯演算法，具體見Initiate Multipart Upload介面。**
- **為了保證數據在網路傳輸過程中不出現錯誤，用戶發送請求時攜帶Content-MD5，OSS會計算上傳數據的MD5與用戶上傳的MD5值比較，如果不一致返回InvalidDigest錯誤碼。**

樣本

請求樣本:

```
PUT /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E36  HTTP/
1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
```

```
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMm
UmWjI=
[6291456 bytes data]
```

返回樣本:

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

7.2.3 UploadPartCopy

UploadPartCopy**通過從一個已存在的Object中拷貝數據來上傳一個Part**。

通過在Upload Part請求的基礎上增加一個Header:x-oss-copy-source來調用該介面。當拷貝一個大於1GB的檔案時，必須使用Upload Part Copy的方式進行拷貝。Upload Part Copy 的源Bucket地址和目標Bucket地址必須是同一個Region。如果想通過單個操作拷貝小於1GB的檔案，可以參考Copy Object。

請求文法

```
PUT /ObjectName? partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Content-Length: Size
Authorization: SignatureValue
x-oss-copy-source: /SourceBucketName/SourceObjectName
x-oss-copy-source-range:bytes=first-last
```

請求Header

除了通用的請求Header，Upload Part Copy請求中通過下述Header指定拷貝的源Object地址和拷貝的範圍。

名稱	類型	描述
x-oss-copy-source	字元串	複製源地址（必須有可讀許可權） 預設值：無

名稱	類型	描述
x-oss-copy-source-range	整型	源Object的拷貝範圍。如，設定 bytes=0-9，表示傳送第0到第9這10個字元。 當拷貝整個源Object時不需要該請求Header。 預設值：無

下述請求Header作用於x-oss-copy-source指定的源Object。

名稱	類型	描述
x-oss-copy-source-if-match	字元串	如果源Object的ETAG值和用戶提供的ETAG相等，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-none-match	字元串	如果源Object自從用戶指定的時間以後就沒有被修改過，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-unmodified-since	字元串	如果傳入參數中的時間等於或者晚於檔案實際修改時間，則正常傳輸檔案，並返回200 OK；否則返回412 precondition failed錯誤。 預設值：無
x-oss-copy-source-if-modified-since	字元串	如果源Object自從用戶指定的時間以後被修改過，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無

響應元素(Response Elements)

名稱	類型	描述
x-oss-copy-source-if-match	字元串	如果源Object的ETAG值和用戶提供的ETAG相等，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-none-match	字元串	如果源Object自從用戶指定的時間以後就沒有被修改過，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無
x-oss-copy-source-if-unmodified-since	字元串	如果傳入參數中的時間等於或者晚於檔案實際修改時間，則正常傳輸檔案，並返回200 OK；否則返回412 precondition failed錯誤。 預設值：無
x-oss-copy-source-if-modified-since	字元串	如果源Object自從用戶指定的時間以後被修改過，則執行拷貝操作；否則返回412 HTTP錯誤碼（預先處理失敗）。 預設值：無

細節分析

- 調用該介面上傳Part數據前，必須調用Initiate Multipart Upload介面，獲取一個OSS伺服器頒發的Upload ID。
- Multipart Upload要求除最後一個Part以外，其他的Part大小都要大於100KB。但是Upload Part介面並不會立即校驗上傳Part的大小（因為不知道是否為最後一塊）；只有當Complete Multipart Upload的時候才會校驗。
- 不指定x-oss-copy-source-range要求標頭時，表示拷貝整個源Object。當指定該要求標頭時，則返回消息中會包含整個檔案的長度和此次拷貝的範圍，例如：Content-Range: bytes 0-9/44，表示整個檔案長度為44，此次拷貝的範圍為0-9。當指定的範圍不符合範圍規範時，則拷貝整個檔案，並且不在結果中提及Content-Range。

- 若調用Initiate Multipart Upload介面時，指定了x-oss-server-side-encryption要求標頭，則會對上傳的Part進行加密編碼，並在Upload Part回應標頭中返回x-oss-server-side-encryption頭，其值表明該Part的伺服器端密碼編譯演算法，具體見Initiate Multipart Upload介面。
- 該操作不能拷貝通過Append追加上傳方式產生的object。
- 如果Bucket的類型為Archive，則不能調用該介面，否則返回400錯誤，錯誤碼為OperationNotSupported。

樣本

請求樣本:

```
PUT /multipart.data?partNumber=1&uploadId=0004B9895DBBB6EC98E36  HTTP/
1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMm
UmWjI=
x-oss-copy-source: /oss-example/ src-object
x-oss-copy-source-range:bytes=100-6291756
```

返回樣本:

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 3e6aba62-1eae-d246-6118-8ff42cd0c21a
Date: Thu, 17 Jul 2014 06:27:54 GMT'
<?xml version="1.0" encoding="UTF-8"?>
<CopyPartResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs.com”>
    <LastModified>2014-07-17T06:27:54.000Z </LastModified>
    <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE"</ETag>
</CopyPartResult>
```

7.2.4 CompleteMultipartUpload

在將所有數據Part都上傳完成後，必須調用Complete Multipart Upload API來完成整個檔案的Multipart Upload。

在執行該操作時，用戶必須提供所有有效數據Part的列表（包括part號碼和ETAG）。OSS收到用戶提交的Part列表後，會逐一驗證每個數據Part的有效性。當所有的數據Part驗證通過後，OSS將把這些數據part組合成一個完整的Object。

請求文法

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

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

請求參數(Request Parameters)

Complete Multipart Upload時，可以通過encoding-type對返回結果中的Key進行編碼。

名稱	類型	描述
encoding-type	字元串	指定對返回的Key進行編碼，目前支援url編碼。Key使用UTF-8字元，但xml 1.0標準不支援解析一些控制字元，比如ascii值從0到10的字元。對於Key中包含xml 1.0標準不支援的控制字元，可以通過指定encoding-type對返回的Key進行編碼。 預設值：無 可選值：url

請求元素(Request Elements)

名稱	類型	描述
CompleteMultipartUpload	容器	保存保存Complete Multipart Upload請求內容的容器。 子節點：一個或多個Part元素 父節點：無
ETag	字元串	Part成功上傳後，OSS返回的ETag值。 父節點：Part

名稱	類型	描述
Part	容器	保存已經上傳Part資訊的容器。 子節點：ETag, PartNumber 父節點：InitiateMultipartUploadResult
PartNumber	整數	Part數目。 父節點：Part

響應元素(Response Elements)

名稱	類型	描述
Bucket	字元串	Bucket名稱。 父節點：CompleteMultipartUploadResult
CompleteMultipartUploadResult	容器	保存Complete Multipart Upload請求結果的容器。 子節點：Bucket, Key, ETag, Location 父節點：None
ETag	字元串	ETag (entity tag) 在每個Object生成的時候被建立，用於標示一個Object的內容。Complete Multipart Upload請求建立的Object，ETag值是其內容的UUID。ETag值可以用於檢查Object內容是否發生變化。 父節點：CompleteMultipartUploadResult
Location	字元串	新建立Object的URL。 父節點：CompleteMultipartUploadResult

名稱	類型	描述
Key	字元串	新建立Object的名字。 父節點：CompleteMultipartUploadResult
EncodingType	字元串	指明返回結果中編碼使用的類型。如果請求的參數中指定了encoding-type，那返回的結果會對Key進行編碼。 父節點：容器

細節分析

- Complete Multipart Upload時，會確認除最後一塊以外所有塊的大小都大於100KB，並檢查用戶提交的Partlist中的每一個Part號碼和Etag。所以在上傳Part時，客戶端除了需要記錄Part號碼外，還需要記錄每次上傳Part成功後，伺服器返回的ETag值。
- OSS處理Complete Multipart Upload請求時，會持續一定的時間。在這段時間內，如果客戶端和OSS之間的連結斷掉，OSS仍會繼續將請求做完。
- 用戶提交的Part List中，Part號碼可以是不連續的。例如第一塊的Part號碼是1，第二塊的Part號碼是5。
- OSS處理Complete Multipart Upload請求成功後，該Upload ID就會變成無效。
- 同一個Object可以同時擁有不同的Upload Id，當Complete一個Upload ID後，該Object的其他Upload ID不受影響。
- 若調用Initiate Multipart Upload介面時，指定了x-oss-server-side-encryption要求標頭，則在Complete Multipart Upload的回應標頭中返回x-oss-server-side-encryption，其值表明該Object的伺服器端密碼編譯演算法。
- 如果用戶上傳了Content-MD5要求標頭，OSS會計算body的Content-MD5並檢查一致性。如果不一致，將返回InvalidDigest錯誤碼。

樣本

請求樣本：

```
POST /multipart.data? uploadId=0004B9B2D2F7815C432C9057C03134D4  HTTP/
1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 1056
Date: Fri, 24 Feb 2012 10:19:18 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:8VwFhFUWmVecK6jQlHlXMK/
zMT0=

<CompleteMultipartUpload>
```

```
    <Part>
        <PartNumber>1</PartNumber>
        <ETag>"3349DC700140D7F86A078484278075A9"</ETag>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <ETag>"8EFDA8BE206636A695359836FE0A0E0A"</ETag>
    </Part>
    <Part>
        <PartNumber>8</PartNumber>
        <ETag>"8C3150651671324441774 11FDA149B92"</ETag>
    </Part>
</CompleteMultipartUpload>
```

返回樣本：

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

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns=”http://doc.oss-cn-hangzhou.
aliyuncs.com”>
    <Location>http://oss-example.oss-cn-hangzhou.aliyuncs.com /
multipart.data</Location>
    <Bucket>oss-example</Bucket>
    <Key>multipart.data</Key>
    <ETag>B864DB6A936D376F9F8D3ED3BBE540DD-3</ETag>
</CompleteMultipartUploadResult>
```

7.2.5 AbortMultipartUpload

AbortMultipartUpload**介面可以根據用戶提供的Upload ID中止其對應的Multipart Upload事件。**

當一個Multipart Upload事件被中止後，就不能再使用這個Upload ID做任何操作，已經上傳的Part數據也會被刪除。

請求文法

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

細節分析

- **中止一個Multipart Upload事件時，如果其所屬的某些Part仍然在上傳，那麼這次中止操作將無法刪除這些Part。所以如果存在並發訪問的情況，為了徹底釋放OSS上的空間，需要調用幾次Abort Multipart Upload介面。**
- **如果輸入的Upload Id不存在，OSS會返回404錯誤，錯誤碼為：NoSuchUpload。**

樣本

請求樣本:

```
Delete /multipart.data?&uploadId=0004B9895DBBB6EC98E  HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:J/lICfXEvPmmSW86bBAfMm
UmWjI=
```

返回樣本:

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

7.2.6 ListMultipartUploads

ListMultipartUploads**可以羅列出所有執行中的Multipart Upload事件，即已經被初始化的Multipart Upload但是未被Complete或者Abort的Multipart Upload事件。**

OSS返回的羅列結果中最多會包含1000個Multipart Upload資訊。如果想指定OSS返回羅列結果內Multipart Upload資訊的數目，可以在請求中添加max-uploads參數。另外，OSS返回羅列結果中的IsTruncated元素標明是否還有其他的Multipart Upload。

請求文法

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

請求參數(Request Parameters)

名稱	類型	描述
delimiter	字元串	是一個用於對Object名字進行分組的字元。所有名字包含指定的首碼且第一次出現delimiter字元之間的object作為一組元素——CommonPrefixes。
max-uploads	字元串	限定此次返回Multipart Uploads事件的最大數目，如果不設定，預設為1000，max-uploads取值不能大於1000。

名稱	類型	描述
key-marker	字元串	與upload-id-marker參數一同使用來指定返回結果的起始位置。 · 如果upload-id-marker參數未設定，查詢結果中包含：所有Object名字的字典序大於key-marker參數值的Multipart事件。 · 如果upload-id-marker參數被設定，查詢結果中包含：所有Object名字的字典序大於key-marker參數值的Multipart事件和Object名字等於key-marker參數值，但是Upload ID比upload-id-marker參數值大的Multipart Uploads事件。
prefix	字元串	限定返回的object key必須以prefix作為首碼。注意使用prefix查詢時，返回的key中仍會包含prefix。
upload-id-marker	字元串	與key-marker參數一同使用來指定返回結果的起始位置。 · 如果key-marker參數未設定，則OSS忽略upload-id-marker參數。 · 如果key-marker參數被設定，查詢結果中包含：所有Object名字的字典序大於key-marker參數值的Multipart事件和Object名字等於key-marker參數值，但是Upload ID比upload-id-marker參數值大的Multipart Uploads事件。

名稱	類型	描述
encoding-type	字元串	指定對返回的內容進行編碼，指定編碼的類型。Delimiter、KeyMarker、Prefix、N和Key使用UTF-8字元，但xml 1.0標準不支援解析一些控制字元，比如ascii值從0到10的字元。對於包含xml 1.0標準不支援的控制字元，可以通過指定encoding-type對返回的Delimiter、KeyMarker、Prefix、NextK和Key進行編碼。 預設值：無

響應元素(Response Elements)

名稱	類型	描述
ListMultipartUploadsResult	容器	保存List Multipart Upload請求結果的容器。 子節點：Bucket, KeyMarker, UploadIdMarker, NextKeyMarker, NextUploadIdMarker, MasUploads, Delimiter, Prefix, CommonPrefixes, IsTruncated, Upload 父節點：None
Bucket	字元串	Bucket名稱。 父節點：ListMultipartUploadsResult

名稱	類型	描述
EncodingType	字元串	指明返回結果中編碼使用的類型。如果請求的參數中指定了encoding-type，那返回的結果會對Delimiter、KeyMarker、Prefix、NextK和Key這些元素進行編碼。 父節點：ListMultipartUploadsResult
KeyMarker	字元串	列表的起始Object位置。 父節點：ListMultipartUploadsResult
UploadIdMarker	字元串	列表的起始UploadID位置。 父節點：ListMultipartUploadsResult
NextKeyMarker	字元串	如果本次沒有返回全部結果，響應請求中將包含NextKeyMarker元素，用於標明接下來請求的KeyMarker值。 父節點：ListMultipartUploadsResult
NextUploadMarker	字元串	如果本次沒有返回全部結果，響應請求中將包含NextUploadMarker元素，用於標明接下來請求的UploadMarker值。 父節點：ListMultipartUploadsResult
MaxUploads	整數	返回的最大Upload數目。 父節點：ListMultipartUploadsResult

名稱	類型	描述
IsTruncated	枚舉字元串	標明是否本次返回的Multipart Upload結果清單被截斷。“true”表示本次沒有返回全部結果；“false”表示本次已經返回了全部結果。 有效值：false、true 預設值：false 父節點：ListMultipartUploadsResult
Upload	容器	保存Multipart Upload事件資訊的容器。 子節點：Key, UploadId, Initiated 父節點：ListMultipartUploadsResult
Key	字元串	初始化Multipart Upload事件的Object名字。 父節點：Upload
UploadId	字元串	Multipart Upload事件的ID。 父節點：Upload
Initiated	日期	Multipart Upload事件初始化的時間。 父節點：Upload

細節分析

- max-uploads參數最大值為1000。
- 在OSS的返回結果首先按照Object名字字典序升序排列；對於同一個Object，則按照時間序，升序排列。
- 可以靈活地使用prefix參數對bucket內的object進行分組管理（類似與檔案夾的功能）。

- **List Multipart Uploads請求支援5種請求參數：** prefix、marker、delimiter、upload-id-marker**和**max-uploads**。通過這些參數的組合，可以設定查詢Multipart Uploads事件的規則，獲得期望的查詢結果。**

樣本

請求樣本：

```
Get /?uploads  HTTP/1.1
Host:oss-example. oss-cn-hangzhou.aliyuncs.com
Date: Thu, 23 Feb 2012 06:14:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:JX75CtQqsmBBz+dcivn7kwBM
vOY=
```

返回樣本：

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

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns=”http://doc.oss-cn-hangzhou.aliyuncs
.com”>
    <Bucket>oss-example</Bucket>
    <KeyMarker></KeyMarker>
    <UploadIdMarker></UploadIdMarker>
    <NextKeyMarker>oss.avi</NextKeyMarker>
    <NextUploadIdMarker>0004B99B8E707874FC2D692FA5D77D3F</NextUpload
IdMarker>
    <Delimiter></Delimiter>
    <Prefix></Prefix>
    <MaxUploads>1000</MaxUploads>
    <IsTruncated>false</IsTruncated>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0004B999EF518A1FE585B0C9360DC4C8</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0004B999EF5A239BB9138C6227D69F95</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>oss.avi</Key>
        <UploadId>0004B99B8E707874FC2D692FA5D77D3F</UploadId>
        <Initiated>2012-02-23T06:14:27.000Z</Initiated>
    </Upload>
</ListMultipartUploadsResult>
```

7.2.7 ListParts

ListParts**介面用於列舉指定Upload ID所屬的所有已經上傳成功Part。**

請求文法

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

請求參數(Request Parameters)

名稱	類型	描述
uploadId	字元串	Multipart Upload事件的ID。 預設值：無
max-parts	整數	規定在OSS響應中的最大Part數目。 預設值：1,000
part-number-marker	整數	指定List的起始位置，只有Part Number數目大於該參數的Part會被列出。 預設值：無
encoding-type	字元串	指定對返回的內容進行編碼，指定編碼的類型。Key使用UTF-8字元，但xml 1.0標準不支援解析一些控制字元，比如ascii值從0到10的字元。對於Key中包含xml 1.0標準不支援的控制字元，可以通過指定encoding-type對返回的Key進行編碼。 預設值：無 可選值：url

響應元素(Response Elements)

名稱	類型	描述
ListPartsResult	容器	保存List Part請求結果的容器。 子節點：Bucket, Key, UploadId, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part 父節點：無
Bucket	字元串	Bucket名稱。 父節點：ListPartsResult
EncodingType	字元串	指明對返回結果進行編碼使用的類型。如果請求的參數中指定了encoding-type，那會對返回結果中的Key進行編碼。 父節點：ListPartsResult
Key	字元串	Object名稱。 父節點：ListPartsResult
UploadId	字元串	Upload事件ID。 父節點：ListPartsResult
PartNumberMarker	整數	本次List結果的Part Number起始位置。 父節點：ListPartsResult
NextPartNumberMarker	整數	如果本次沒有返回全部結果，響應請求中將包含NextPartNumberMarker元素，用於標明接下來請求的PartNumberMarker值。 父節點：ListPartsResult

名稱	類型	描述
MaxParts	整數	返回請求中最大的Part數目。 父節點：ListPartsResult
IsTruncated	枚舉字元串	標明是否本次返回的List Part結果清單被截斷。“true”表示本次沒有返回全部結果；“false”表示本次已經返回了全部結果。 有效值：true、false 父節點：ListPartsResult
Part	字元串	保存Part資訊的容器 子節點：PartNumber, LastModified, ETag, Size 父節點：ListPartsResult
PartNumber	整數	標示Part的數字。 父節點：ListPartsResult.Part
LastModified	日期	Part上傳的時間。 父節點：ListPartsResult.part
ETag	字元串	已上傳Part內容的ETag。 父節點：ListPartsResult.Part
Size	整數	已上傳Part大小。 父節點：ListPartsResult.Part

細節分析

- List Parts支援max-parts和part-number-marker兩種請求參數。
- max-parts參數最大值為1000；預設值也為1000。
- 在OSS的返回結果按照Part號碼升序排列。

- **由於網路傳輸可能出錯，所以不推薦用List Part出來的結果（Part Number和ETag值）來生成最後Complete Multipart的Part列表。**

樣本

請求樣本：

```
Get  /multipart.data?uploadId=0004B999EF5A239BB9138C6227D69F95  HTTP/1
.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 23 Feb 2012 07:13:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:4qOnUMc9UQWqkz8wDqD3
lIsa9P8=
```

返回樣本：

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

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

7.3 許可權控制（ACL)

7.3.1 PutObjectACL

PutObjectACL介面用於修改Object的存取權限。

目前Object有四種存取權限：default，private, public-read, public-read-write。Put Object ACL操作通過Put請求中的“x-oss-object-acl”頭來設定，這個操作只有Bucket Owner有許可權執行。如果操作成功，則返回200；否則返回相應的錯誤碼和提示資訊。

請求文法

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

Object ACL釋義

名稱	描述
private	該ACL表明某個Object是私有資源，即只有該Object的Owner擁有該Object的讀寫權限，其他的用戶沒有許可權操作該Object
public-read	該ACL表明某個Object是公共讀資源，即非Object Owner只有該Object的讀許可權，而Object Owner擁有該Object的讀寫權限
public-read-write	該ACL表明某個Object是公共讀寫資源，即所有用戶擁有對該Object的讀寫權限
default	該ACL表明某個Object是遵循Bucket讀寫權限的資源，即Bucket是什麼許可權，Object就是什麼許可權

細節分析

- Object的讀操作包括：GetObject，HeadObject，CopyObject和UploadPartCopy中的對source object的讀；Object的寫操作包括：PutObject，PostObject，AppendObject，DeleteObject，DeleteMultipleObjects，CompleteMultipartUpload以及CopyObject對新的Object的寫。
- x-oss-object-acl中許可權的值必須在上述4種許可權中。如果有不屬於上述4種的許可權，OSS返回400 Bad Request消息，錯誤碼：InvalidArgument。
- 用戶不僅可以通過PutObjectACL介面來設定Object ACL，還可以在Object的寫操作時，在要求標頭中帶上x-oss-object-acl來設定Object ACL，效果與PutObjectA ACL等同。例如PutObject時在要求標頭中帶上x-oss-object-acl可以在上傳一個Object的同時設定某個Object的ACL。

- **對某個Object沒有讀許可權的用戶讀取某個Object時，OSS返回 403 Forbidden消息，錯誤碼：AccessDenied，提示：You do not have read permission on this object。**
- **對某個Object沒有寫入權限的用戶寫某個Object時，OSS返回 403 Forbidden消息，錯誤碼：AccessDenied，提示：You do not have write permission on this object。**
- **只有Bucket Owner才有許可權調用PutObjectACL來修改該Bucket下某個Object的ACL。非Bucket Owner調用PutObjectACL時，OSS返回 403 Forbidden消息，錯誤碼：AccessDenied，提示：You do not have write acl permission on this object。**
- **Object ACL優先順序高於Bucket ACL。例如Bucket ACL是private的，而Object ACL是public-read-write的，則訪問這個Object時，先判斷Object的ACL，所以所有用戶都擁有這個Object的存取權限，即使這個Bucket是private bucket。如果某個Object從來沒設定過ACL，則存取權限遵循Bucket ACL。**

樣本

請求樣本：

```
PUT /test-object?acl HTTP/1.1
x-oss-object-acl: public-read
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3J
xrTZHiA=
```

返回樣本：

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

7.3.2 GetObjectACL

GetObjectACL**用來獲取某個Bucket下的某個Object的存取權限。**

請求文法

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

```
Authorization: SignatureValue
```

響應元素(Response Elements)

名稱	類型	描述
AccessControlList	容器	儲存ACL資訊的容器 父節點：AccessControlPolicy
AccessControlPolicy	容器	保存Get Object ACL結果的容器 父節點：None
DisplayName	字元串	Bucket擁有者的名稱.。(目前和ID一致) 父節點：AccessControlPolicy.Owner
Grant	枚舉字元串	Object的ACL許可權 有效值：private，public-read，public-read-write 父節點：AccessControlPolicy.AccessControlList
ID	字元串	Bucket擁有者的用戶ID 父節點：AccessControlPolicy.Owner
Owner	容器	保存Bucket擁有者資訊的容器。 父節點：AccessControlPolicy

細節分析

- 只有Bucket的擁有者才能使用GetObjectACL這個介面來獲取該Bucket下某個Object的ACL，非Bucket Owner調用該介面時，返回403 Forbidden消息。錯誤碼：AccessDenied，提示You do not have read acl permission on this object。

- **如果從來沒有對某個Object設定過ACL，則調用GetObjectACL時，OSS返回的ObjectACL會是default，表明該Object ACL遵循Bucket ACL。即：如果Bucket是private的，則該object也是private的；如果該object是public-read-write的，則該object也是public-read-write的。**

樣本

請求樣本：

```
GET /test-object?acl HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0Fmg
brQ0=
```

返回樣本：

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

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

7.4 軟連結（Symlink）

7.4.1 PutSymlink

PutSymlink**可用於針對OSS上的TargetObject建立符號連結，用戶可以通過該符號連結訪問TargetObject。**

請求文法

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

```
x-oss-symlink-target: TargetObjectName
```

請求Header

名稱	類型	描述
x-oss-symlink-target	字元串	符號連結指向的目標檔案。 合法值：命名規範同Object。

細節分析

- TargetObjectName同ObjectName一樣，需要URL encode。
- 符號連結的目標檔案類型不能為符號連結。
- 建立符號連結時，
  - 不檢查目標檔案是否存在
  - 不檢查目標檔案類型是否合法
  - 不檢查目標檔案是否有許可權訪問

  以上檢查，都延遲到GetObject等需要訪問目標檔案的API。
- 如果試圖添加的檔案已經存在，並且有存取權限。新添加的檔案將覆蓋原來的檔案，成功返回200 OK。
- 如果在PutSymlink的時候，攜帶以x-oss-meta-為首碼的參數，則視為user meta，比如x-oss-meta-location。一個Object可以有多個類似的參數，但所有的user meta總大小不能超過8k。
- 如果Bucket的類型為Archive，則不能調用該介面，否則返回400錯誤，錯誤碼為OperationNotSupported。

樣本

請求樣本：

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

返回樣本：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 08 Nov 2016 02:00:25 GMT
Content-Length: 0
```

```
Connection: keep-alive
x-oss-request-id: 582131B9109F4EE66CDE56A5
ETag: "0A477B89B4602AA8DECB8E19BFD447B6"
```

7.4.2 GetSymlink

GetSymlink**用於獲取符號連結，此操作要求用戶對該符號連結有讀許可權**。

請求文法

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

響應Header

名稱	類型	描述
x-oss-symlink-target	字元串	符號連結指向的目標檔案。

細節分析

如果符號連結不存在返回404 Not Found錯誤。錯誤碼：NoSuchKey

樣本

請求樣本：

```
GET /link-to-oss.jpg?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 06:38:30 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:UNQDb7GapEgJCZkcde6O
hZ9Jfe8=
```

返回樣本：

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5650BD72207FB30443962F9A
x-oss-symlink-target: oss.jpg
ETag: "A797938C31D59EDD08D86188F6D5B872"
```

7.5 標籤（Tagging）

7.6 PutObjectTagging

您可以通過PutObjectTagging介面設定或更新對象的標籤（Object Tagging）。

請求文法

```
PUT /objectname?tagging
Content-Length: 114
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
<Tagging>
  <TagSet>
    <Tag>
      <Key>Key</Key>
      <Value>Value</Value>
    </Tag>
  </TagSet>
</Tagging>
```

請求元素

名稱	類型	是否必需	描述
Tagging	容器	是	子節點：TagSet
TagSet	容器	是	父節點：Tagging 子節點：Tag
Tag	容器	否	父節點：TagSet 子節點：Key，Value
Key	字串	否	父節點：Tag 子節點：無
Value	字串	否	父節點：Tag 子節點：無

細節分析

- **要求者需要有PutObjectTagging許可權。**
- **更改Tagging#會更新Object Last-Modified時間。**
- **標籤合法字元集包括大小寫字母、數字、空格和以下符號：**

  +-=._:/

示#

- **請求樣本**

```
PUT /objectname?tagging
Content-Length: 114
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Mon, 18 Mar 2019 08:25:17 GMT
Authorization: OSS ***********:**********************
<Tagging>
  <TagSet>
    <Tag>
      <Key>a</Key>
      <Value>1</Value>
    </Tag>
    <Tag>
      <Key>b</Key>
      <Value>2</Value>
    </Tag>
  </TagSet>
</Tagging>
```

- **返回樣本**

```
200 (OK)
content-length: 0
server: AliyunOSS
x-oss-request-id: 5C8F55ED461FB4A64C000004
date: Mon, 18 Mar 2019 08:25:17 GMT
```

7.7 GetObjectTagging

您可以通過GetObjectTagging介面擷取對象的標籤資訊。

請求文法

```
GET /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

響應元素

名稱	類型	描述
Tagging	**容器**	**子節點：TagSet**
TagSet	**容器**	**父節點：Tagging** **子節點：Tag**
Tag	**容器**	**父節點：TagSet** **子節點：Key，Value**

名稱	類型	描述
Key	字串	父節點：Tag 子節點：無
Value	字串	父節點：Tag 子節點：無

樣本

- 請求樣本

```
GET /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 20 Mar 2019 02:02:36 GMT
Authorization: OSS ************:********************
```

- 響應樣本

```
200 (OK)
content-length: 209
server: AliyunOSS
x-oss-request-id: 5C919F38461FB42826000002
date: Wed, 20 Mar 2019 02:02:32 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<Tagging>
  <TagSet>
    <Tag>
      <Key>a</Key>
      <Value>1</Value>
    </Tag>
    <Tag>
      <Key>b</Key>
      <Value>2</Value>
    </Tag>
  </TagSet>
</Tagging>
```

7.8 DeleteObjectTagging

您可以通過DeleteObjectTagging刪除指定對象的標籤。

請求文法

```
DELETE /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
```

```
Authorization: SignatureValue
```

樣本

- **請求樣本**

```
DELETE /objectname?tagging
Host: bucketname.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 09 Apr 2019 03:00:33 GMT
Authorization: OSS LTAIbsTkySSptaz****/Zr0o6BKgAl7iiBtHN2JMC****
```

- **響應樣本**

```
204 (No Content)
content-length: 0
server: AliyunOSS
x-oss-request-id: 5CAC0AD16D0232E2051B****
date: Tue, 09 Apr 2019 03:00:33 GMT
```