Object Storage Service

API リファレンス

API リファレンス

OSS API ドキュメントの概要

Object Storage Service (OSS) は、非常に大きな容量、高度なセキュリティ、低いコスト、高い信頼性を誇る、Alibaba Cloud のクラウドストレージサービスです。このドキュメントで説明するシンプルな REST インターフェイスを使用して、いつでも、どこでも、あらゆるインターネットデバイスからデータをアップロード/ダウンロードできます。OSS を使用すると、各種のマルチメディア共有 Web サイト、オンラインストレージ、個人および企業のデータバックアップなど、多種多様なデータベースのサービスを開発できます。

ここで取り上げるインターフェイスを使用する前に、OSS プロダクトについての指示、利用規約、課金方法を十分に理解しておく必要があります。

API の概要

API の概要

基本概念

本ドキュメントの用語は OSS の概要 を参照してください。

サービス操作

API	説明
GetService	このアカウントに属するすべてのバケットを取得

バケット操作

API	説明

Put Bucket	バケットを作成
Put Bucket ACL	バケットのアクセス権限を設定
Put Bucket Logging	バケットのログ機能を有効化
Put Bucket Website	バケットを静的 Web サイトホスティングモードに設定
Put Bucket Referer	バケットの Anti-leech 保護ルールを設定
Put Bucket Lifecycle	バケット内のオブジェクトのライフサイクルルールを設定
Get Bucket Acl	バケットのアクセス権限を取得
Get Bucket Location	データセンター内のバケットの場所に関する情報を取得
Get Bucket Logging	バケットのアクセスログ設定を表示
Get Bucket Website	バケットの静的 Web サイトホスティングステータスを表示
Get Bucket Referer	バケットの Anti-leech 保護ルールを表示
Get Bucket Lifecycle	バケット内のオブジェクトのライフサイクルルールを表示
Delete Bucket	バケットを削除
Delete Bucket Logging	バケットのログ機能を無効化
Delete Bucket Website	バケットの静的 Web サイトホスティングモードを無効化
Delete Bucket Lifecycle	バケット内のオブジェクトのライフサイクルルールを削除
Get Bucket(List Object)	バケット内のすべてのオブジェクトに関する情報を取得

オブジェクトの操作

API	説明
Put Object	オブジェクトをアップロード
Copy Object	オブジェクトを別のオブジェクトとしてコピー
Get Object	オブジェクトを取得
Delete Object	オブジェクトを削除
Delete Multiple Objects	複数のオブジェクトを削除
Head Object	オブジェクトのメタ情報を取得
Post Object	post リクエストを使用してオブジェクトをアップロード

Append Object	アップロードデータをオブジェクトの最後に追加
Put Object ACL	オブジェクトの ACL を設定
Get Object ACL	オブジェクトの ACL の情報を取得

マルチパートアップロード操作

API	説明
Initiate Multipart Upload	MultipartUpload イベントを初期化
Upload Part	オブジェクトをパート単位でアップロード
Upload Part Copy	ファイルのパートをコピーしてオブジェクトをアップロード
Complete Multipart Upload	ファイル全体のマルチパートアップロードを完了
Abort Multipart Upload	マルチパートアップロードイベントを中止
List Multipart Uploads	実行中のすべてのマルチパートアップロードイベントをリスト
List Parts	指定したアップロード ID に属する正常にアップロードされたすべてのパートをリスト

Cross-Origin Resource Sharing (CORS)

API	説明
Put Bucket cors	指定したバケットの CORS ルールを設定
Get Bucket cors	指定したバケットの現在の CORS ルールを取得
Delete Bucket cors	指定したバケットで CORS 機能を無効にして、そのバケットのすべての CORS ルールを削除
Option Object	クロスオリジンアクセスのプリフライトリクエスト

アクセス制御

ユーザー署名の認証

OSS は、Access Key ID/Access Key Secret の対称暗号化方式を使用してリクエストの送信者の ID を検証します。AccessKey ID はユーザーを識別します。AccessKey Secret は、ユーザーによる署名文字列の暗号化と、OSS による署名文字列の Access Key の検証に使用されます。AccessKey Secret は、ユーザーと OSS 以外には知られないようにする必要があります。AccessKey は、アカウントタイプに基づいて次のタイプに分類できます。

- Alibaba Cloud アカウントの AccessKey: 各 Alibaba Cloud アカウントによって提供される AccessKey には、そのリソースに対する完全な権限が付与されます。
- RAM アカウントの AccessKey: RAM アカウントは、Alibaba Cloud アカウントの権限付与に基づいて生成されます。RAM アカウントの AccessKey には、特定のリソースに対する操作権限が付与されます。
- STS の一時的なアクセス資格情報: 一時的な資格情報は、Alibaba Cloud アカウントまたは RAM アカウントによって生成されます。一時的な資格情報の AccessKey には、特定のリソースに対する操作権限が期限付きで付与されます。有効期限を過ぎると権限が取り消されます。

詳細については、OSS プロダクトのドキュメントの「アクセス ID の検証」を参照してください。

個々の ID として OSS にリクエストを送信する前に、OSS の規定の形式に従ってリクエストの署名文字列を生成する必要があります。さらに、AccessKey Secret を使用してその署名文字列を暗号化して、検証コードを生成します。OSS は、リクエストを受信すると、Access Key ID に基づいて対応する Access Key Secret を見つけて、署名文字列と検証コードを同じ方法で取得します。取得した検証コードと提供された検証コードが同じだった場合、そのリクエストは有効と見なされます。同じでなかった場合は、そのリクエストは拒否されて、HTTP 403 エラーが返されます。

ヘッダーへの署名の追加

Authorization ヘッダーを追加して、HTTP リクエストに署名情報を含めることができます。これにより、そのメッセージが権限付与済みであることが示されます。

Authorization フィールドの計算

```
Authorization = "OSS " + AccessKeyId + ":" + Signature

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

- AccessKeySecret は、署名に必要なキーを示します。
- VERB は、HTTP リクエストメソッド (PUT、GET、POST、HEAD、DELETE) を示します。
- \nは改行です。
- Content-MD5 は、リクエストコンテンツデータの MD5 の値を示します。メッセージの内容 (ヘッダーを除く) は、128ビットの数値であるMD5値を取得するために計算されます。 この番号はBase64でContent-MD5値にエンコードされます。 要求ヘッダーを使用して、メッセージの有効性をチェックすることができます。つまり、メッセージの内容が “eB5eJF1ptWaXm4bijSPyxw==”などの送信されたコンテンツと一致するかどうかを確認できます。詳細については、『RFC2616』を参照してください。
- Content-Type は、リクエストコンテンツのタイプを示します。” application/octet-stream” のようなものである。 それは空であるかもしれません。
- DATE は、操作の日時を示します。HTTP1.1 でサポートされている GMT 形式である必要があります。
- CanonicalizedOSSHeaders は、プレフィックスが “x-oss- ” である HTTP ヘッダーのアセンブリを示します。
- CanonicalizedResource は、ユーザーがアクセスしようとしている OSS リソースを示します。

これらのうち、DATE と CanonicalizedResource は、値を空白にすることができません。リクエストの DATE の値と OSS サーバーの時刻の差が 15 分を超えていると、OSS サーバーはサービスを拒否して HTTP 403 エラーを返します。

CanonicalizedOSSHeaders の作成

プレフィックスが “x-oss- ” であるすべての HTTP ヘッダーを CanonicalizedOSSHeaders と呼びます。CanonicalizedOSSHeaders を作成する方法を以下に示します。

1. プレフィックスが **x-oss-** であるすべての HTTP リクエストヘッダーの名前を**小文字**に変換します。たとえば、X-OSS-Meta-Name: TaoBaoを x-oss-meta-name: TaoBao に変換します。
2. 前の手順で得られたすべての HTTP リクエストヘッダーを辞書式順序の昇順で並べ替えます。
3. 『RFC2616』の第 4.2 章に従って、同じ名前のリクエストヘッダーを結合します (2 つの値をコンマ (,) で区切ります)。たとえば、” x-oss-meta-name” というリクエストヘッダーが 2 つあり、値が “TaoBao” と “Alipay” である場合、値を結合すると x-oss-meta-name:TaoBao,Alipay になります。
4. リクエストのヘッダーとコンテンツの間にある区切り記号の両側のスペースを削除します。たとえば、x-oss-meta-name: TaoBao,Alipay を x-oss-meta-name:TaoBao,Alipay に変換します。
5. すべてのヘッダーとコンテンツのペアを、区切り記号に “\n” を使用して区切ります。これで、CanonicalizedOSSHeaders が完成します。

注意

- CanonicalizedOSSHeaders が null の場合、末尾に \n を追加する必要はありません。
- ヘッダーが1つだけの場合、x-oss-meta-a \n のようにする必要があります。 最後に \n に注意してください。
- 複数のヘッダーがある場合、x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n のように

なります。 最後に\nに注意してください。

CanonicalizedResource の作成

ユーザーが送信するリクエストに指定されているターゲット OSS リソースを CanonicalizedResource と呼びます。CanonicalizedResource を作成する方法を以下に示します。

1. CanonicalizedResource を空白の文字列 (“”) として設定します。
2. アクセスするOSSリソースを ‘/BucketName/ObjectName’ の形式で追加します。 (**ObjectNameが存在しない場合**、CanonicalizedResourceは /BucketName/です。 **BucketNameが存在しない場合**、CanonicalizedResourceは “/ “です。
3. リクエストするリソースにサブリソースが含まれている場合は、すべてのサブリソースを辞書式順序の昇順で並べ替えて、区切り記号に & を使用して区切り、サブリソース文字列を生成します。 そのサブリソース文字列を、” ?” と共に CanonicalizedResource 文字列の末尾に追加します。 この場合、CanonicalizedResource は、たとえば /BucketName/ObjectName?acl &uploadId=UploadId のようになります。
4. ユーザーリクエストでクエリ文字列 (QueryString、HTTPリクエストパラメータとも呼ばれます) が指定されている場合は、これらのクエリ文字列をソートし、**辞書順の**昇順で値を要求し、セパレータ ‘&’ を使用してクエリ文字列と要求値を分離し、パラメータに基づいてそれらを CanonicalizedResourceに追加します。この場合、CanonicalizedResourceは次のようになります。/BucketName/ObjectName ? acl＆response-content-type=ContentType&uploadId=UploadIdのようになります。

注意:

- 現在、OSSでサポートされているサブリソースには、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 および response-content-encoding。
- サブリソースには次の3種類があります。
  - リソース識別子 (acl、append、uploadId、およびsymlink サブリソースなど) 。 詳細については、パケット関連の操作 とオブジェクト関連の操作を参照してください) 。
  - response - *** のようなレスポンスヘッダフィールドを指定します。 詳細は、Get Objectを参照してください。
  - オブジェクト処理メソッド x-oss-processなど、これは、イメージサービスのアクセスルールなどのオブジェクト処理メソッドとして使用されます。

署名ヘッダーの計算のルール

1. 署名に使用する文字列は UTF-8 形式である必要があります。漢字を含む署名文字列に対しては、UTF-8 エンコーディングを実行する必要があります。その後、Access Key Secretと共に使用して最終的な署名を計算できます。
2. 署名方法には、RFC 2104 で定義されている HMAC-SHA1 方式が採用されています (Key は Access Key Secretです)。
3. リクエストの Content-Type と Content-MD5 は省略可能です。署名の検証が必要な場合は null 値を改行 “\n” に置き換えます。
4. すべての非標準 HTTP ヘッダーのうち、署名文字列が必要なのは、” x-oss- “ で始まるヘッダーだけです。その他の非標準 HTTP ヘッダー (上の例の x-oss-magic など) は、OSS では無視されます。
5. “x-oss- “ で始まるヘッダーを署名の検証に使用するには、次の仕様に準拠している必要があります。
   - ヘッダー名が小文字に変更されている。
   - ヘッダーが辞書式順序の昇順で並べ替えられている。
   - ヘッダーの名前と値を区切るコロンの前後にスペースがない。
   - 各ヘッダーの後に改行 “\n” がある。ヘッダーがない場合、CanonicalizedOSSHeaders は null に設定されます。

署名の例

AccessId が “44CF9590006BF252F707” で、 Access Key Secret が “OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV” であるとすると、署名を追加する方法は次のようになります。

リクエスト	署名文字列計算式	署名文字列
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 eB5eJF1ptWaXm4bijSPyxw==\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/nelson”

署名の計算方法は次のとおりです。

Python コードの例

```
import base64
import hmac
import sha
h = hmac.new("OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
"PUT\neB5eJF1ptWaXm4bijSPyxw==\ntext/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-magic:abracadabra\nx-
```

```
oss-meta-author:foo@bar.com\n/oss-example/nelson", sha)
base64.encodestring(h.digest()).strip()
print("Signature: %s" % Signature)
```

計算される署名は 26NBxoKdsyly4EDv6inkoDft/yA= です。式Authorization = “OSS” + AccessKeyID + “:” + Signatureによれば、許可の値はOSS 44CF9590006BF252F707：26NBxoKdsyly4EDv6inkoDft/yA= です。この値は、送信されるメッセージを形成するために認証ヘッダーとともに追加されます。

```
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
```

詳細分析

1. AccessID が受信されない場合や、受信した AccessID がアクティブでない場合は、エラー 403 Forbidden が返されます。エラーコードは InvalidAccessKeyId です。
2. ユーザーリクエストヘッダーの Authorization の値が無効である場合は、エラー 400 Bad Request が返されます。エラーコードは InvalidArgument です。
3. OSS のすべてのリクエストで、HTTP 1.1 プロトコルによって規定されている GMT 時刻形式を使用する必要があります。次の 3 つの日付形式を使用できます。date1 = 2DIGIT SP month SP 4DIGIT; day month year (例: 02 Jun 1982)上の 3 つの日付形式では、” day” が “2 DIGIT” になっています。したがって、” Jun 2”、” 2 Jun 1982”、” 2-Jun-82” はすべて無効な日付形式になります。
4. Date がヘッダーに入力されていない場合や、署名の検証で形式が正しくないと判断された場合は、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
5. リクエストは、OSS サーバーの現在の時刻に基づいて 15 分以内に入力する必要があります。そうしないと、エラー 403 Forbidden が返されます。エラーコードは RequestTimeTooSkewed です。
6. AccessID はアクティブだが、ユーザーリクエストの署名が正しくないと判断された場合は、エラー 403 Forbidden が返され、検証と暗号化のための正しい署名文字列がユーザーに応答メッセージで返されます。この OSS の応答に基づいて、署名文字列が正しいかどうかを確認できます。戻り値の例:

   ```
   <?xml version="1.0" ?>
   <Error>
   <Code>
   SignatureDoesNotMatch
   </Code>
   <Message>
   ```

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

注意:

- OSS SDKが署名を実装しました。 OSS SDKの使用中に署名の問題について心配する必要はありません。 特定の言語の署名実装の詳細については、OSS SDKのコードを参照してください。 OSS SDKの署名を実装するためのファイルは、次の表のとおりです。

SDK	署名の実装
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

よくある質問

コンテンツMD5の計算方法

```
Content-MD5 calculation
The message content "123456789" is used as an example. The Content-MD5 value of the string

is calculated as follows:
The algorithm defined in related standards can be simplified to the following:
1. Calculate the MD5-encrypted 128-bit binary array.
2. Encode the binary array (instead of the 32-bit string code) with Base64.

Python is used as an example.
The correct calculation code is:
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='

Note:
The correct code is: hash.digest(), used to calculate a 128-bit binary array
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'

A common error is encoding the calculated 32-bit string code with Base64.
An incorrect example: hash.hexdigest(), and a visible 32-bit string will be calculated.
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
Result of encoding the incorrect MD5 value with Base64:
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

URL への署名の追加

URL への署名の追加

権限付与済みアクセスを実装するには、Authorization ヘッダーを使用するほかに、URL に署名情報を追加してサードパーティに転送することもできます。

実装方法

署名を含む URL の例:

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

URL の署名には、少なくとも、Signature、Expires、OSSAccessKeyId の 3 つのパラメーターが含まれている必要があります。

- Expires: URL のタイムアウト時間を示します。このパラメーターの値は UNIX 時間です (1970 年 1 月 1 日午前 0 時 0 分 0 秒 (UTC) からの経過秒数。詳細については Wikipedia を参照)。OSS が URL リクエストを受信した時刻が、署名に含まれている Expires パラメーターの値より遅かった場合は、リクエストがタイムアウトしたことを示すエラーコードが返されます。たとえば、現在の時刻が 1141889060 である場合に、60 秒後に自動的に無効になる URL を作成するには、Expires の値を 1141889120 に設定します。
- OSSAccessKeyId: Access Key ID を示します。
- Signature: 署名情報を示します。OSS では、さまざまなリクエストとヘッダーパラメーターがサポートされていますが、URL に署名を追加するためのアルゴリズムは、ヘッダーへの署名の追加のアルゴリズムと基本的に同じです。

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

違いは次のとおりです。

1. 署名が URL に追加されると、Date パラメーターが Expires パラメーターに置き換えられます。
2. 署名を URL とヘッダーに同時に含めることはできません。
3. 複数の受信Signature、Expires、またはAccessKeyId値が使用可能な場合は、最初の値が使用されます。
4. リクエストの時刻が Expires の時刻より遅いかどうかは署名の前に検証されます。
5. 署名文字列をURLに挿入するときは、URLのUrlEncodeを必ず実行してください。

- 一時的なユーザーのURLに署名を追加する場合は、「security-token」を指定する必要があります。 形式は次のとおりです。

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

サンプルコード

URL に署名を追加する Python コードの例:

```
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())
```

注意:

- 上記はPythonのサンプルコードです。
- OSS SDKは、URLに署名を追加する方法を提供します。 使用方法については、SDKファイルの「アクセス許可」を参照してください。
- OSS SDKのURLに署名を追加する実装については、下記の表を参照してください。

SDK	URL署名方式	実装ファイル
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

詳細分析

1. URL に署名を追加する方法を使用する場合、権限付与済みのデータは、有効期限が切れるとインターネットで公開されます。この方法を使用する際には注意してください。
2. URL に署名を追加する方法は、PUT と GET の両方のリクエストでサポートされています。
3. 署名が URL に追加されると、Signature、Expires、OSSAccessKeyId の順序が変更される場合があります。Signature、Expires、OSSAccessKeyId の各パラメーターがすべて揃っていないと、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
4. 現在のアクセス時刻がリクエストの Expires の設定時刻より遅い場合は、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
5. Expires の時刻の形式が正しくない場合は、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
6. URL に Signature、Expires、OSSAccessKeyId の各パラメーターが 1 つ以上含まれている場合に、ヘッダーに署名情報が含まれていると、エラー 400 Bad Request が返されます。エラーコード

は InvalidArgument です。
7. 署名文字列が生成されると、Date パラメーターが Expires パラメーターに置き換えられますが、前のセクションで説明した content-type や content-md5 などのヘッダーはそのまま含まれます (Date リクエストヘッダーはまだリクエストに存在するため、署名文字列に Date ヘッダーを追加する必要はありません)。

一時的な権限を使ったアクセス

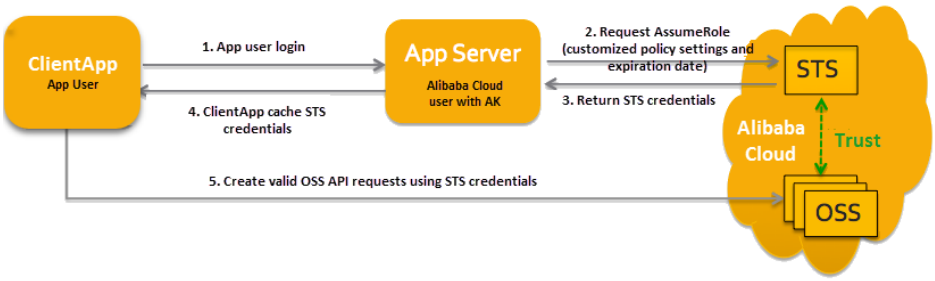
一時的なアクセス資格情報

STS の概要

OSS では、Alibaba Cloud Security Token Service (STS) を使用して、アクセスに一時的な権限を付与することができます。Alibaba Cloud STS (Security Token Service) とは、クラウドコンピューティングユーザーに一時的なアクセストークンを提供する Web サービスです。STS を使用すると、サードパーティ製のアプリケーションや関連するユーザーにアクセス資格情報を付与できます。権限と有効期間をカスタマイズすることもできます (ユーザー ID を管理できます)。サードパーティ製のアプリケーションや関連するユーザーは、これらのアクセス資格情報を使用して、Alibaba Cloud プロダクトの API を直接呼び出したり、Alibaba Cloud プロダクトの SDK を使用してクラウドプロダクトの API にアクセスしたりできます。

- サードパーティ製のアプリケーションに長期キー (AccessKey) を公開する必要はありません。アクセストークンを生成してサードパーティ製アプリケーションに送信するだけで済みます。このトークンのアクセス権限と有効期間はカスタマイズできます。
- 権限の失効について気にかける必要もありません。アクセス資格情報は、有効期限が切れると自動的に無効になります。

アプリを例として、やり取りのプロセスを以下に示します。



このソリューションの詳細を以下に示します。

1. アプリユーザーとしてログインします。アプリユーザーの ID はお客様が管理します。ID 管理シス

テムをカスタマイズすることも、外部の Web アカウントや OpenID を使用することもできます。AppServer で、有効なアプリユーザーごとに最小限のアクセス権限を細かく定義できます。
2. AppServer が STS にセキュリティトークン (SecurityToken) をリクエストします。STS を呼び出す前に、アプリユーザーの最小限のアクセス権限 (ポリシー構文で記述) と権限付与の有効期限を決定する必要があります。決定したら、STS の AssumeRole インターフェイスを呼び出してセキュリティトークンを取得します。ロールの管理と使用方法の詳細については、『RAM ユーザーガイド』の役割の管理を参照してください。
3. STS から AppServer に有効なアクセス資格情報が返されます。このアクセス資格情報には、セキュリティトークン、一時的なアクセスキー (AccessKeyId と AccessKeySecret)、有効期限が含まれます。
4. AppServer から ClientApp にアクセス資格情報が返されます。ClientApp はこの資格情報をキャッシュします。資格情報が無効になったら、AppServer に新しい有効なアクセス資格情報をリクエストする必要があります。たとえば、アクセス資格情報の有効期間が 1 時間である場合、AppServer にアクセス資格情報の更新を 30 分ごとにリクエストできます。
5. ClientApp は、ローカルにキャッシュしたアクセス資格情報を使用して Alibaba Cloud サービスの API をリクエストします。ECS は、STS のアクセス資格情報を認識し、STS を使用して資格情報を検証して、ユーザーリクエストに適切に応答します。

STS のセキュリティトークンの詳細については、『RAM ユーザーガイド』の役割の管理を参照してください。ここでのポイントは、STS の AssumeRole インターフェイスを呼び出すだけで有効なアクセス資格情報を取得できることです。このメソッドは、STS DSK を使用して呼び出すこともできます。詳細はこちらをクリックしてください。

STS の資格情報を使用して署名付きリクエストを作成する

ユーザーのクライアントは、STS の一時的な資格情報を取得すると、その資格情報のセキュリティトークン (SecurityToken) と一時的なアクセスキー (AccessKeyId と AccessKeySecret) を使用して署名を作成します。権限付与済みアクセスの署名を作成する方法は、ルートアカウントを使用する AccessKey を使ってヘッダーに署名を追加する方法と基本的に同じです。次の 2 つのポイントに注意してください。

- ユーザーが使用する署名キーは、STS によって提供される一時的なアクセスキー (AccessKeyId と AccessKeySecret) です。
- セキュリティトークン (Security Token) をリクエストヘッダーに含めるか、リクエストパラメーターとして URI に含める必要があります。この 2 つの方法を同時に使用することはできません。同時に使用すると、InvalidArgument エラーが返されます。
    - ヘッダーに “x-oss-security-token: SecurityToken” を含めます。x-oss-security-token は、署名の CanonicalizedOSSHeaders が計算されるときに考慮されます。
    - URL にパラメーター security-token=SecurityToken を含めます。security-token は、署名の CanonicalizedResource が計算されるときに考慮され、サブリソースと見なされます。

バケット権限の制御

OSSは、バケット・レベルのアクセス制御のためのアクセス制御リスト（ACL）を提供します。現在、バケットには、パブリックリード/ライト、パブリックリード、プライベートの3つのアクセス権があります。

許可	訪問者のアクセス制限
パブリック読み取り/書き込み	- 誰でも（匿名アクセスを含む）バケット内のオブジェクトの読み込み、書き込み、および削除が可能です。 - そのような操作によって発生する料金は、バケットの所有者が負担します。この許可は慎重に使用してください。
パブリック読み取り	- バケットの所有者と許可されたユーザーだけがバケット内のオブジェクトに対して書き込み操作と削除操作を実行できます。 - 誰でも（匿名アクセスを含む）バケット内のオブジェクトを読み取ることができます。
プライベート	- バケットの所有者と許可されたユーザーだけが、バケット内のオブジェクトに対する読み込み、書き込み、および削除操作を実行できます。 - 他のユーザーは、バケット内のオブジェクトにアクセスすることはできません。

注意:

- 権限が指定されていない新しいバケットが作成されると、OSSは自動的にバケットのプライベート許可を設定します。
- 既存のバケットの場合、バケットの作成者のみが、OSSが提供するPut Bucket Aclインターフェイスを使用してそのパーミッションを変更できます。

共通 HTTP ヘッダーの定義

共通リクエストヘッダー

OSS RESTful インターフェイスでは、いくつかの共通リクエストヘッダーを使用します。これらのリクエス

トヘッダーはすべての OSS リクエストで使用できます。これらのリクエストヘッダーの具体的な定義を次の表に示します。

名前	説明
Authorization	リクエストの有効性の検証に使用する検証情報を示します。 型: string デフォルト値: なし アプリケーションシナリオ: 非匿名リクエスト
Content-Length	『RFC2616』で定義されている HTTP リクエストコンテンツの長さを示します。 型: string デフォルト値: なし アプリケーションシナリオ: データを OSS に送信する必要があるリクエスト
Content-Type	『RFC2616』で定義されている HTTP リクエストコンテンツのタイプを示します。 型: string デフォルト値: なし アプリケーションシナリオ: データを OSS に送信する必要があるリクエスト
Date	HTTP 1.1 プロトコルで規定されている GMT 時刻を示します (例: Wed, 05 Sep. 2012 23:00:00 GMT)。 型: string デフォルト値: なし
Host	アクセス先のホストを示します。値の形式は <バケット名>.oss-cn-hangzhou.aliyuncs.com です。 型: string デフォルト値: なし

共通応答ヘッダー

OSS RESTful インターフェイスでは、いくつかの共通応答ヘッダーを使用します。これらの応答ヘッダーはすべての OSS リクエストで使用できます。これらの応答ヘッダーの具体的な定義を次の表に示します。

名前	説明
Content-Length	『RFC2616』で定義されている HTTP リクエストコンテンツの長さを示します。 型: string デフォルト値: なし アプリケーションシナリオ: データを OSS に送信する必要があるリクエスト
Connection	クライアントと OSS サーバーの間の接続ステータスを示します。 型: enumerative 有効な値: open と close デフォルト値: なし
Date	HTTP 1.1 プロトコルで規定されている GMT 時

	刻を示します (例: Wed, 05 Sep. 2012 23:00:00 GMT)。 型: string デフォルト値: なし
ETag	ETag (エンティティタグ) は、オブジェクトの生成時に作成され、オブジェクトのコンテンツを示すために使用されます。Put Object リクエストを使用して作成されたオブジェクトの場合、ETag の値はオブジェクトのコンテンツの MD5 値です。その他の方法で作成されたオブジェクトの場合、ETag の値はオブジェクトのコンテンツの UUID です。この ETag の値を使用して、オブジェクトのコンテンツが変更されているかどうかを確認できます。 型: string デフォルト値: なし
Server	応答を生成するサーバーを示します。 型: string デフォルト値: AliyunOSS
x-oss-request-id	Alibaba Cloud OSS によって作成される、応答の UUID を示します。OSS サービスの使用時に問題が発生した場合は、問題を迅速に特定できるように、このフィールドを使用して OSS サポート担当者にお問い合わせください。 型: string デフォルト値: なし

サービス操作

GetService(ListBucket)

GetService (ListBuckets)

サーバーに Get リクエストを送信すると、リクエスト送信者が所有するすべてのバケットを取得できます。”/“はルートディレクトリを表します。

リクエスト構文

```
GET / HTTP/1.1
```

```
Host: oss.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

リクエストパラメーター

GetService(ListBuckets) を使用するときに、prefix、marker、max-uploads の各パラメーターで条件を指定することで、結果を絞ることができます。

名前	説明
prefix	返されるバケット名をフィルターしたい場合はプレフィックスを設定します。prefix を設定しない場合はプレフィックス情報はフィルター処理されません。 データ型: 文字列 デフォルト値: なし
marker	設定した marker の後のエントリーをアルファベット順で結果を返します。marker を設定しない場合は結果が先頭から返されます。 データ型: 文字列 デフォルト値: なし
max-uploads	1 件のリクエストに対して返されるバケットの最大数を制限します。設定されていない場合のデフォルト値は 100 です。max-uploads の上限は 1000 です。 データ型: 文字列 デフォルト値: 100

応答の要素

名前	説明
ListAllMyBucketsResult	Get Service リクエストの結果の保存に使用されるコンテナー。 型: コンテナー サブノード: Owner、Buckets 親ノード: なし
Prefix	このクエリの結果のプレフィックス。このノードは、返されていないバケットがある場合にのみ表示されます。 型: 文字列 親ノード: ListAllMyBucketsResult
Marker	この GetService(ListBuckets) リクエストの開始位置を示します。このノードは、返されていないバケットがある場合にのみ表示されます。 型: 文字列 親ノード: ListAllMyBucketsResult
MaxKeys	リクエストに応じて返される結果の最大数。このノードは、返されていないバケットがある場合に

	のみ表示されます。 型: 文字列 親ノード: ListAllMyBucketsResult
IsTruncated	すべての結果が返されたかどうかを示します。”true” は返されなかった結果があることを示し、”false” はすべての結果が返されたことを示します。このノードは、返されていないバケットがある場合にのみ表示されます。 型: 列挙文字列 有効な値: true と false 親ノード: ListAllMyBucketsResult
NextMarker	これは、返されなかった結果を次の GetService(ListBuckets) リクエストで取得するためのマーカーと見なすことができます。このノードは、返されていないバケットがある場合にのみ表示されます。 型: 文字列 親ノード: ListAllMyBucketsResult
Owner	バケットオーナーに関する情報の保存に使用されるコンテナー。 型: コンテナー 親ノード: ListAllMyBucketsResult
ID	バケットオーナーのユーザー ID。 型: 文字列 親ノード: ListAllMyBucketsResult.Owner
DisplayName	バケットオーナーの名前 (現在は ID と同じです)。 型: 文字列 親ノード: ListAllMyBucketsResult.Owner
Buckets	複数のバケットに関する情報の保存に使用されるコンテナー。 型: コンテナー サブノード: Bucket 親ノード: ListAllMyBucketsResult
Bucket	バケット情報の保存に使用されるコンテナー。 型: コンテナー サブノード: Name、CreationDate、Location 親ノード: ListAllMyBucketsResult.Buckets
Name	バケット名。 型: 文字列 親ノード: ListAllMyBucketsResult.Buckets.Bucket
CreateDate	バケットの作成日時。 型: 日時 (形式: yyyy-mm-ddThh:mm:ss.timezone、例: 2011-12-01T12:27:13.000Z) 親ノード: ListAllMyBucketsResult.Buckets.Bucket
Location	バケットが配置されているデータセンター。 型: 文字列 親ノード:

	ListAllMyBucketsResult.Buckets.Bucket
ExtranetEndpoint	バケットがアクセスするインターネットドメイン名 型: 文字列 親ノード: ListAllMyBucketsResult.Buckets.Bucket
IntranetEndpoint	同一リージョンのECSからバケットにアクセスするイントラネットドメイン名 型: 文字列 親ノード: ListAllMyBucketsResult.Buckets.Bucket
StorageClass	バケットのストレージタイプ。 “Standard”, “IA”, and “Archive” が利用可能です。 (“Archive” は現状同一リージョンでのみ利用可能です。 Type: 文字列 Parent node: ListAllMyBucketsResult.Buckets.Bucket

詳細分析

1. GetService API は、認証されたユーザーに対してのみ有効です。
2. リクエストでユーザー認証の情報が指定されていない場合、つまり匿名アクセスの場合は、403 Forbidden が返されます。エラーコードは AccessDenied です。
3. すべてのバケットが返されている場合、返される xml にノード Prefix、Marker、MaxKeys、IsTruncated、および NextMarker は含まれません。まだ返されていない結果がある場合は、上記のノードが追加されます。NextMarker は、後続のクエリのマーカーを割り当てるために使用されます。

例

リクエストの例 I

```
GET / HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: oss-cn-hangzhou.aliyuncs.com
Authorization: OSS nxj7dtl1c24jwhcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv2qUl5jI=
```

戻り値の例 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>
```

リクエストの例 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 nxj7dtl1c24jwhcyl5hpvnhi:COS3OQkfQPnKmYZTEHYv2qUl5jI=
```

戻り値の例 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>
```

バケット操作

PutBucket

バケット作成の有効にするために PutBucket を利用します (匿名アクセスはサポートされていません) 。バケットの作成リージョンに従い、バケット内にすべてのオブジェクトはそのリージョンに所属されます。詳細については、OSS アクセスドメイン名を参照してください。

リクエスト構文

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

<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration>
<StorageClass>Standard</StorageClass>
</CreateBucketConfiguration>
```

詳細分析

1. x-oss-aclヘッダーの使用により、バケットのアクセス権限を有効になります。現在バケットアク

セス権限は、公開読み書き、公開読み取り、非公開の3つがあります。
2. リクエストされたバケットすでに存在する場合は、409 Conflictが表示されます。エラーコード: BucketAlreadyExists。
3. 作成するバケットが命名規則に従っていない場合は、400 Bad Request が表示されます。エラーコード: InvalidBucketName。
4. PUT Bucket リクエストの開始時にユーザー認証の情報が導入されていない場合は、403 Forbidden が表示されます。エラーコード: AccessDenied 。
5. PutBucket リクエストを開始したときに作成できるバケットの最大数(**デフォルトで30**) が超えられた場合、400 Bad Request が表示されます。エラーコード：TooManyBuckets。
6. 作成されたバケットにアクセス許可が指定されていない場合、デフォルトでPrivateアクセス権が適用されます。
7. 新しいバケットの格納タイプを指定できます。Standard、IA、および Archive が使用可能です。

サンプル

リクエストの例:

```
PUT / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2017 03:15:40 GMT
x-oss-acl: private
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:77Dvh5wQgIjWjwO/KyRt8dOPfo8=

<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration>
<StorageClass>Standard</StorageClass>
</CreateBucketConfiguration>
```

レスポンスの例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2017 03:15:40 GMT
Location: /oss-example
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

PutBucketACL

Put Bucket Acl

Put Bucket ACL インターフェイスを使用すると、バケットのアクセス権限を変更できます。現在バケットで使用できるアクセス権限は、公開読み書き、公開読み取り、非公開の 3 つです。Put Bucket ACL 操作を設定するには、Put リクエストで “x-oss-acl” ヘッダーを使用します。この操作を実行できるのはバケットの作成者だけです。操作が成功すると、200 が返されます。失敗すると、対応するエラーコードとプロンプトメッセージが返されます。

リクエスト構文

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

詳細分析

1. バケットが既に存在しており、リクエスト送信者が所有しているが、リクエストの権限が既存の権限と異なっている場合は、バケットのコンテンツは変更されずに権限が更新されます。
2. Put Bucket リクエストの開始時にユーザー認証の情報が導入されていない場合は、403 Forbidden というメッセージが返されます。エラーコードは AccessDenied です。
3. リクエストに有効な “x-oss-acl” ヘッダーが含まれておらず、バケットが既に存在していてリクエスト送信者に属している場合、元のバケットの権限は変更されません。

例

リクエストの例:

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

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

この設定の権限が存在しない場合は、400 Bad Request というメッセージが表示されます。

エラーの戻り値の例:

```
HTTP/1.1 400 Bad Request
x-oss-request-id: 56594298207FB304438516F9
Date: Fri, 24 Feb 2012 03:55:00 GMT
Content-Length: 309
Content-Type: text/xml; charset=UTF-8
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" ?>
<Error>
<Code>InvalidArgument</Code>
<Message>no such bucket access control exists</Message>
<RequestId>56594298207FB304438516F9</RequestId>
<HostId>leo.oss-test.aliyun-inc.com</HostId>
<ArgumentName>x-oss-acl</ArgumentName>
<ArgumentValue>error-acl</ArgumentValue>
</Error>
```

PutBucketLogging

バケットオーナーは、自分のバケットでアクセスログ機能を有効にすることができます。この機能を有効にすると、そのバケットへのリクエストに関する詳細情報が自動的に記録されて、指定したルールに従ってアクセスログが書き込まれます。アクセスログは、指定したバケットに 1 時間ごとにオブジェクトとして書き込まれます。バケットオーナーは、OSS が提供するバケットアクセスログを使用して、バケットのアクセス状況を簡単に把握および分析できます。OSS が提供するバケットアクセスログは、すべてのアクセスレコードが記録されていることを保証するものではありません。

> バケットオーナーは、OSS が提供するバケットアクセスログを使用して、バケットのアクセス状況を簡単に把握および分析できます。OSS が提供するバケットアクセスログは、すべてのアクセスレコードが記録されていることを保証するものではありません。

リクエスト構文

```
PUT /?logging HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Authorization: SignatureValue
```

```
Host: BucketName.oss-cn-hangzhou.aliyuncs.com

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
<LoggingEnabled>
<TargetBucket>TargetBucket</TargetBucket>
<TargetPrefix>TargetPrefix</TargetPrefix>
</LoggingEnabled>
</BucketLoggingStatus>
```

リクエストの要素

名前	説明	必須
BucketLoggingStatus	ログステータス情報のコンテナー。 型: コンテナー 子: LoggingEnabled 親: なし	はい
LoggingEnabled	ログ情報のコンテナー。この要素は、ログを有効にする場合にのみ存在します (ログを無効にする場合は存在しません)。 型: コンテナー 子: TargetBucket、TargetPrefix 親: BucketLoggingStatus	いいえ
TargetBucket	サーバーアクセスログを格納するバケットを指定します。 型: 文字列 子: なし 親: BucketLoggingStatus.LoggingEnabled	この要素は、サーバーアクセスログに有効化が必要です。
TargetPrefix	この要素を使用すると、ログファイルが格納されるオブジェクトのプレフィックスを指定できます。 型: 文字列 子: なし 親: BucketLoggingStatus.LoggingEnabled	いいえ

アクセスログを格納するオブジェクトの命名規則

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

この命名規則では、TargetPrefix はユーザーが指定します。YYYY、mm、DD、HH、MM、SS には、作成

日時の年、月、日、時、分、秒がアラビア数字で設定されます (桁数に注意してください)。UniqueString は、OSS システムによって生成される文字列です。OSS アクセスログの格納に使用されるオブジェクトの名前の実例を以下に示します。

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

上の例では、” MyLog-“ はユーザーが指定したオブジェクトプレフィックス、” oss-example” はオリジンバケットの名前、” 2012-09-10-04-00-00” はオブジェクトの作成日時 (北京現地時間)、” 0000” は OSS システムによって生成された文字列です。

ログファイルの形式

名前	例	説明
Remote IP	119.140.142.11	リクエストが開始された IP アドレス (このフィールドは、プロキシやユーザーファイアウォールによってブロックされる場合があります)
Reserved	-	予約フィールド
Reserved	-	予約フィールド
Time	[02/May/2012:00:00:04 +0800]	OSS がリクエストを受信した日時
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	ユーザーがリクエストした URI (クエリ文字列を含む)
HTTP Status	200	OSS から返された HTTP ステータスコード
SentBytes	5576	ユーザーが OSS からダウンロードしたトラフィック
RequestTime (ms)	71	このリクエストが完了するまでにかかった時間 (単位はミリ秒)
Referer	http://www.alicloud.com/product/oss	リクエストの HTTP リファラー
User-Agent	curl/7.15.5	HTTP User-Agent ヘッダー
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	アクセスリクエストのドメイン名
Request ID	505B01695037C2AF032593A4	このリクエストを一意に識別するために使用される UUID
LoggingFlag	true	アクセスログ機能が有効かどうか
Requester Alibaba Cloud ID	1657136103983691	リクエスト送信者の Alibaba Cloud ID (匿名アクセスの場合は “-“)

Operation	GetObject	リクエストタイプ
Bucket	oss-example	アクセスをリクエストされたバケットの名前
Key	/aliyun-logo.png	ユーザーリクエストのキー
ObjectSize	5576	オブジェクトサイズ
Server Cost Time (ms)	17	OSS サーバーがこのリクエストの処理に費やした時間 (単位はミリ秒)
Error Code	NoSuchBucket	OSS から返されたエラーコード
Request Length	302	ユーザーリクエストの長さ (バイト)
UserID	1657136103983691	バケットオーナーの ID
Delta DataSize	280	バケットサイズの変動 (変動がない場合は “- ”)
Sync Request	-	CNDからのバックツーソースリクエストかどうか、(そうでない場合は “- ”)
Reserved	-	予約フィールド

詳細分析

- ソースバケットとターゲットバケットは同じユーザーに属している必要があります。
- 上のリクエスト構文で、” BucketName” はアクセスログを有効にするバケット、” TargetBucket” はアクセスログの保存先のバケット、” TargetPrefix” はアクセスログを格納するオブジェクトの名前のプレフィックス (null にすることもできます) を指します。
- ソースバケットとターゲットバケットは、同じバケットでも別のバケットでもかまいません。複数のソースバケットのログを同じターゲットバケットに保存できます (TargetPrefix を別の値に設定することをお勧めします)。
- バケットのアクセスログ機能を無効にするには、BucketLoggingStatus が空のリクエストを送信するだけです。詳細については、以下のリクエストの例を参照してください。
- すべての PUT Bucket Logging リクエストに署名が必要です。したがって、匿名アクセスはサポートされていません。
- PUT Bucket Logging リクエストを開始したユーザーがソースバケット (リクエストの例の BucketName) のオーナーでない場合、エラーコード 403 が返されます。
- ソースバケットが存在しない場合は、エラーコード NoSuchBucket が返されます。
- PUT Bucket Logging リクエストを開始したユーザーがターゲットバケット (リクエストの例の TargetBucket) のオーナーでない場合、エラーコード 403 が返されます。ターゲットバケットが存在しない場合は、エラーコード InvalidTargetBucketForLogging が返されます。
- ソースバケットとターゲットバケットは同じデータセンターに属している必要があります。属していない場合は、エラー 400 が返されます。エラーコードは InvalidTargetBucketForLogging です

。
- PUT Bucket Logging リクエストの XML が無効である場合は、エラーコード MalformedXML が返されます。
- ソースバケットとターゲットバケットは同じバケットでもかまいません。複数の異なるソースバケットのログを同じターゲットバケットに保存できます (TargetPrefix を別の値に設定する必要があります)。
- ソースバケットが削除されると、対応するログルールも削除されます。
- バケットアクセスログファイルは 1 時間ごとに生成されますが、 その時間内のすべてのリクエストがその時間のログファイルに記録されるとは限りません。前のログファイルや次のログファイルに記録される場合もあります。
- OSS によって生成されるログファイルの命名規則の “UniqueString” は、OSS がオブジェクトを一意に識別するために生成する UUID です。
- バケットアクセスログファイルが生成されるたびに、PUT 操作がカウントされ、占有スペースが記録されますが、生成されたトラフィックは記録されません。生成されたログファイルは、通常のオブジェクトとして操作できます。
- “x-“ というプレフィックスが付いたクエリ文字列パラメーターは、OSS ではすべて無視されますが、アクセスログには記録されます。大量のアクセスログで特定のリクエストにマークを付けるには、” x-“ というプレフィックスが付いたクエリ文字列パラメーターを URL に追加できます。例:
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png?x-user=admin
- 上の 2 つのリクエストは、OSS の処理結果は同じですが、 アクセスログで “x-user=admin” を検索すると、マークを付けたリクエストをすばやく見つけることができます。
- OSS ログのフィールドの値が “-“ になっている場合があります。これは、データが不明であるか、そのフィールドが現在のリクエストに対して無効であることを示します。
- 将来的に、OSS ログファイルの末尾に必要に応じてフィールドが追加されます。ログ処理ツールを開発する際には、互換性の問題を考慮に入れることをお勧めします。
- Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

リクエストの例 (バケットアクセスログを有効にする場合):

```
PUT /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
<LoggingEnabled>
<TargetBucket>doc-log</TargetBucket>
<TargetPrefix>MyLog-</TargetPrefix>
```

```
</LoggingEnabled>
</BucketLoggingStatus>
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

リクエストの例 (バケットアクセスログを無効にする場合):

```
PUT /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Type: application/xml
Content-Length: 86
Date: Fri, 04 May 2012 04:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus>
</BucketLoggingStatus>
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 04:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

PutBucketWebsite

Put Bucket Website

Put Bucket Website 操作を使用すると、バケットを静的 Web サイトホスティングモードに設定できます。

リクエスト構文

```
PUT /?website HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>errorDocument.html</Key>
</ErrorDocument>
</WebsiteConfiguration>
```

リクエストの要素

名前	説明	必須
ErrorDocument	子要素Keyの親要素。 型: コンテナー 親要素: WebsiteConfiguration	いいえ
IndexDocument	子要素Suffixの親要素。 型: コンテナー 親要素: WebsiteConfiguration	はい
Key	4XX クラスエラーが発生した場合に使用するファイル名。 型: 文字列 親要素: WebsiteConfiguration.ErrorDocument 条件: ErrorDocument が指定されている場合は必須	条件による
Suffix	ドキュメントルートを設定するとき、空またはスラッシュ (/) が含まれない。たとえば、インデックスファイルが index.html の場合、oss-cn-hangzhou.aliyuncs.com/mybucket/mydir/ にアクセスすると、デフォルトではoss-cn-hangzhou.aliyuncs.com/mybucket/index.html と同等アクセスであること。 型: 文字列 親要素: WebsiteConfiguration.IndexDocument	はい
WebsiteConfiguration	リクエストのコンテナー。 型: コンテナー 親要素: なし	はい

詳細分析

1. 静的 Web サイトとは、すべての Web ページが静的コンテンツ (クライアントで実行される JavaScript などのスクリプトを含む) で構成されている Web サイトです。OSS では、サーバーで処理する必要があるコンテンツ (PHP、JSP、APS.NET など) はサポートされていません。
2. 自分のドメイン名を使用してバケットベースの静的 Web サイトにアクセスする場合は、CNAME ドメイン名を使用できます。具体的な設定方法については、セクション 3.4 の「カスタマイズドメイン名のバインド」を参照してください。
3. バケットを静的 Web サイトホスティングモードに設定する際には、インデックスページを指定する必要があります。エラーページはオプションです。
4. バケットを静的 Web サイトホスティングモードに設定すると、指定したインデックスページとエラーページがバケット内のオブジェクトになります。
5. バケットを静的 Web サイトホスティングモードに設定すると、静的 Web サイトのルートドメイン名への匿名アクセスに対してはインデックスページが返され、静的 Web サイトのルートドメイン名への署名付きアクセスに対しては Get Bucket の結果が返されるようになります。
6. Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

リクエストの例:

```
PUT /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 209
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration>
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
</WebsiteConfiguration>
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
```

```
Server: AliyunOSS
```

PutBucketReferer

Put Bucket Referer

Put Bucket Referer 操作を使用すると、バケットのリファラーアクセスホワイトリストを設定したり、リファラーフィールドが null であるアクセスリクエストを許可するかどうかを指定したりできます。Bucket Referer Anti-leech Protection の詳細については、「OSS Anti-leech 保護」を参照してください。

リクエスト構文

```
PUT /?referer HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss.aliyuncs.com
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
<RefererList>
<Referer> http://www.aliyun.com</Referer>
<Referer> https://www.aliyun.com</Referer>
<Referer> http://www.*.com</Referer>
<Referer> https://www.?.aliyuncs.com</Referer>
</RefererList>
</RefererConfiguration>
```

リクエストの要素

名前	説明	必須
RefererConfiguration	リファラー設定のコンテンツの保存に使用されるコンテナー。 型: コンテナー サブノード: AllowEmptyReferer ノード、RefererList ノード 親ノード: なし	はい
AllowEmptyReferer	リファラーフィールドが null	はい

	であるアクセスリクエストを許可するかどうかを指定します。 型: 列挙文字列 有効な値: true と false デフォルト値: true 親ノード: RefererConfiguration	
RefererList	リファラーアクセスホワイトリストの保存に使用されるコンテナー。 型: コンテナー 親ノード: RefererConfiguration サブノード: Referer	はい
RefererList	リファラーアクセスホワイトリストを指定します。 型: 文字列 親ノード: RefererList	オプション

詳細分析

1. Put Bucket Referer リクエストを開始できるのはバケットオーナーだけです。他のユーザーが開始しようとすると、403 Forbidden というメッセージが返されます。エラーコードは AccessDenied です。
2. AllowEmptyReferer で指定した設定は、前の AllowEmptyReferer の設定を置き換えます。このフィールドは必須です。デフォルトでは、システムの AllowEmptyReferer の設定は true です。
3. この操作を実行すると、以前に設定したホワイトリストが RefererList のホワイトリストで上書きされます。アップロードした RefererList が空である (Referer リクエスト要素が含まれていない) 場合も、設定済みのホワイトリストが上書きされます (以前に設定した RefererList が削除されます)。
4. Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

リクエストの例:

リクエストの例 (Referer が含まれていない場合):

```
PUT /?referer HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

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

```
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

リクエストの例 (Referer が含まれている場合):

```
PUT /?referer HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList>
<Referer> http://www.aliyun.com</Referer>
<Referer> https://www.aliyun.com</Referer>
<Referer> http://www.*.com</Referer>
<Referer> https://www.?.aliyuncs.com</Referer>
</ RefererList>
</RefererConfiguration>
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

PutBucketLifecycle

Put Bucket Lifecycle

バケットオーナーは、Put Bucket Lifecycle リクエストを使用してバケットのライフサイクルを設定できます。ライフサイクルを有効にすると、ライフサイクルルールに一致するオブジェクトが定期的に削除されます。

リクエスト構文

```
PUT /?lifecycle HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Authorization: SignatureValue
Host: BucketName.oss.aliyuncs.com

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
<Rule>
<ID>RuleID</ID>
<Prefix>Prefix</Prefix>
<Status>Status</Status>
<Expiration>
<Days>Days</Days>
</Expiration>
<AbortMultipartUpload>
<Days>Days</Days>
</AbortMultipartUpload>
</Rule>
</LifecycleConfiguration>
```

リクエストの要素

名前	説明	必須
CreatedBeforeDate	ルールがいつ有効になるかを指定します。日付は ISO8601 形式に準拠している必要があります。時刻は常に午前 0 時 (UTC) です。 型: 文字列 親ノード: Expiration	Days または CreatedBeforeDate
Days	オブジェクトが最後に変更されてから何日後にルールが有効になるかを指定します。 型: 正の整数 親ノード: Expiration	Days または CreatedBeforeDate
Expiration	ルールの有効期限のプロパティを指定します。 型: コンテナー サブノード: Days または CreatedBeforeDate 親ノード: Rule	はい
AbortMultipartUpload	満たされていないパートルールの有効期限属性を指定します。 型: コンテナー サブノード: Days または CreatedBeforeDate 親ノード: Rule	いいえ
ID	ルールの一意の ID。長さの上限は 255 バイトです。この値	いいえ

	を指定しなかった場合や、値が null の場合は、自動的に一意の値が生成されます。 型: 文字列 サブノード: なし 親ノード: Rule	
LifecycleConfiguration	ライフサイクル設定の格納に使用されるコンテナー。最大 1000 個のルールを保持できます。 型: コンテナー サブノード: Rule 親ノード: なし	はい
Prefix	ルールが適用されるプレフィックスを指定します。一致するプレフィックスを持つオブジェクトにのみルールが適用されます。 型: 文字列 サブノード: なし 親ノード: Rule	はい
Rule	ルールを記述します。 型: コンテナー サブノード: ID、Prefix、Status、Expiration 親ノード: LifecycleConfiguration	はい
Status	この値が Enabled である場合、このルールは定期的に実行されます。この値が Disabled である場合、このルールは無視されます。 型: 文字列 親ノード: Rule 有効な値: Enabled、Disabled	はい

詳細分析

1. Put Bucket Lifecycle リクエストを開始できるのはバケットオーナーだけです。他のユーザーが開始しようとすると、403 Forbidden というメッセージが返されます。エラーコードは AccessDenied です。
2. 以前にライフサイクルが設定されていない場合は、新しいライフサイクル設定が作成されます。設定されている場合は、前の設定が上書きされます。
3. オブジェクトまたはパートの有効期限を設定することもできます。 ここでパートはマルチパートアップロードのための未提出部分を参照できます。

例

リクエストの例:

```
PUT /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Content-Length: 443
Date: Mon, 14 Apr 2014 01:08:38 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
<Rule>
<ID>delete objects and parts after one day</ID>
<Prefix>logs/</Prefix>
<Status>Enabled</Status>
<Expiration>
<Days>1</Days>
</Expiration>
<AbortMultipartUpload>
<Days>1</Days>
</AbortMultipartUpload>
</Rule>

<Rule>
<ID>delete created before date</ID>
<Prefix>backup/</Prefix>
<Status>Enabled</Status>
<Expiration>
<CreatedBeforeDate>2014-10-11T00:00:00.000Z</CreatedBeforeDate>
</Expiration>
<AbortMultipartUpload>
<CreatedBeforeDate>2014-10-11T00:00:00.000Z</CreatedBeforeDate>
</AbortMultipartUpload>
</Rule>
</LifecycleConfiguration>
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Mon, 14 Apr 2014 01:17:10 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

GetBucket (List Object)

Get Bucket 操作を使用すると、バケット内のオブジェクトの情報をすべて表示できます。

リクエスト構文

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

リクエストパラメーター

GetBucket (ListObject) リクエストを開始するときに、prefix、marker、delimiter、max-keys の各パラメーターを使用して制限を指定すると、部分的な結果を取得することができます。

名前	説明
delimiter	オブジェクト名をグループ化するために使用する文字。指定したプレフィックスから最初の delimiter までの名前を持つオブジェクトはすべて要素のグループ (CommonPrefixes) として機能します。 データ型: 文字列 デフォルト値: なし
marker	アルファベット順で marker 後の最初のエントリから結果を返すように設定します。 データ型: 文字列 デフォルト値: なし
max-keys	1 件のリクエストに対して返されるオブジェクトの最大数を制限します。指定されていない場合のデフォルト値は 100 です。max-keys の値は 1000 以下にする必要があります。 データ型: 文字列 デフォルト値: 100
prefix	返されるオブジェクトキーに指定に応じたプレフィックスが付与されるという制限を適用します。プレフィックスを使用するクエリから返されるキーにはそのプレフィックスも含まれることに注意してください。 データ型: 文字列 デフォルト値: なし
encoding-type	返されるコンテンツのエンコーディングとエンコーディングのタイプを指定します。オブジェクトキーでは UTF-8 文字が使用されますが、XML 1.0 標準では、0 ~ 10 の ASCII 値を含む文字などの特定の制御文字の解析はサポートされていません。XML 1.0 標準でサポートされていない制御文字がオブジェクトキーに含まれている場合は、encoding-type を指定して返されるオブジェクトキーをエンコードできます。 データ型: 文字列 デフォルト値: なし

応答の要素

名前	説明
Contents	返されるすべてのオブジェクトのメタの保存に使用されるコンテナー。 型: コンテナー 親ノード: ListBucketResult
CommonPrefixes	リクエストに delimiter パラメーターを指定した場合、OSS から返される応答に CommonPrefixes 要素が含まれます。この要素は、末尾が delimiter で、共通のプレフィックスを持つ、オブジェクトのセットを示します。 型: 文字列 親ノード: ListBucketResult
Delimiter	オブジェクト名をグループ化するために使用する文字。指定したプレフィックスから最初の delimiter までの名前を持つオブジェクトはすべて要素のグループ (CommonPrefixes) として機能します。 型: 文字列 親ノード: ListBucketResult
encoding-type	返される結果におけるエンコーディングのタイプを指定します。リクエストで encoding-type を指定した場合は、返される結果において Delimiter、Marker、Prefix、NextMarker、Key などの要素がエンコードされます。 型: 文字列 親ノード: ListBucketResult
DisplayName	オブジェクトオーナーの名前。 型: 文字列 親ノード: ListBucketResult.Contents.Owner
ETag	ETag (エンティティタグ) は、オブジェクトの生成時に作成され、オブジェクトのコンテンツを示すために使用されます。Put Object リクエストを使用して作成されたオブジェクトの場合、ETag の値はオブジェクトのコンテンツの MD5 値です。その他の方法で作成されたオブジェクトの場合、ETag の値はオブジェクトのコンテンツの UUID です。この ETag の値を使用して、オブジェクトのコンテンツが変更されているかどうかを確認できます。 型: 文字列 親ノード: ListBucketResult.Contents
ID	バケットオーナーのユーザー ID。 型: 文字列 親ノード: ListBucketResult.Contents.Owner
IsTruncated	すべての結果が返されたかどうかを示します。”true” の場合は、今回返されなかった結果があります。”false” の場合は、すべての結果が返されています。 型: 列挙文字列

	有効な値: true と false 親ノード: ListBucketResult
Key	オブジェクトのキー。 型: 文字列 親ノード: ListBucketResult.Contents
LastModified	オブジェクトが最後に変更された日時。 型: 日時 親ノード: ListBucketResult.Contents
ListBucketResult	Get Bucket リクエストの結果の保存に使用されるコンテナー。 型: コンテナー サブノード: Name、Prefix、Marker、MaxKeys、Delimiter、IsTruncated、Nextmarker、Contents 親ノード: なし
Marker	現在の Get Bucket (List Object) リクエストの開始位置を示します。 型: 文字列 親ノード: ListBucketResult
MaxKeys	リクエストに応じて返される結果の最大数。 型: 文字列 親ノード: ListBucketResult
Name	バケット名。 型: 文字列 親ノード: ListBucketResult
Owner	バケットオーナーに関する情報の保存に使用されるコンテナー。 型: コンテナー サブノード: DisplayName、ID 親ノード: ListBucketResult
Prefix	クエリの現在の結果の開始プレフィックス。 型: 文字列 親ノード: ListBucketResult
Size	オブジェクトのバイト数。 型: 文字列 親ノード: ListBucketResult.Contents
StorageClass	オブジェクトのストレージタイプ。現在利用できるのは “Standard” ,“IA” と “Archive” タイプです。 (Archiveタイプは現在同一リージョンでのみ利用可能です。) 型: 文字列 親ノード: ListBucketResult.Contents

詳細分析

- GetBucket リクエストでは、オブジェクトのユーザー定義メタは返されません。
- アクセス先のバケットが存在しない場合や、標準の命名規則に従っていないために作成できない場合は、エラー 404 Not Found が返されます。エラーコードは NoSuchBucket です。

- バケットにアクセスする権限がない場合は、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
- max-keys の設定のために結果を一度に表示できない場合は、返される結果に <NextMarker> が追加されます。NextMarker は、続きがあることを示すマーカーと見なすことができます。NextMarker の値も結果に含まれます。
- 条件クエリでは、実際にはリストに marker が含まれていなくても、返される結果は、アルファベット順で marker の次に相当するエントリから表示されます。max-keys の値が 0 より小さいか 1000 より大きい場合は、エラー 400 Bad Request が返されます。エラーコードは InvalidArgument です。
- prefix、marker、delimiter のいずれかのパラメーターが長さの要件を満たしていない場合は、400 Bad Request が返されます。エラーコードは InvalidArgument です。
- パラメーターの prefix と marker を使用すると、ページ分割を実現できます。パラメーターの長さは 1024 バイト未満である必要があります。
- フォルダーの名前をプレフィックスとして設定すると、そのプレフィックスで始まるファイルが列挙されて、そのフォルダーのすべてのファイルとサブフォルダーが再帰的に返されます。さらに delimiter を “/ “ に設定すると、返される値にはフォルダーのファイルのみが含まれ、サブフォルダーは CommonPrefixes セクションに返されます。サブフォルダー内のファイルとフォルダーは再帰的に表示されません。たとえば、バケットに次の 3 つのオブジェクトがあったとします。fun/test.jpg、fun/movie/001.avi、fun/movie/007.aviprefix を “fun/ “ に設定すると、3 つのオブジェクトが返されます。さらに delimiter を “/ “ に設定すると、ファイル “fun/test.jpg” とプレフィックス “fun/movie/ “ が返されます。つまり、フォルダーロジックが実現されます。

シナリオ例

バケット “my_oss” に 4 つのオブジェクトがあり、それぞれ次のような名前が付けられています。

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

例

リクエストの例:

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

戻り値の例:

```
HTTP/1.1 200 OK
```

```
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1866
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<Name>oss-example</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter></Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>fun/movie/001.avi</Key>
<LastModified>2012-02-24T08:43:07.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/movie/007.avi</Key>
<LastModified>2012-02-24T08:43:27.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/test.jpg</Key>
<LastModified>2012-02-24T08:42:32.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>oss.jpg</Key>
<LastModified>2012-02-24T06:07:48.000Z</LastModified>
<ETag>&quot;5B3C1A2E053D763E1B002CC607C5A0FE&quot;</ETag>
<Type>Normal</Type>
```

```
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user-example</DisplayName>
</Owner>
</Contents>
</ListBucketResult>
```

リクエストの例 (prefix を含む場合):

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

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 1464
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<Name>oss-example</Name>
<Prefix>fun</Prefix>
<Marker></Marker>
<MaxKeys>100</MaxKeys>
<Delimiter></Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>fun/movie/001.avi</Key>
<LastModified>2012-02-24T08:43:07.000Z</LastModified>
<ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/movie/007.avi</Key>
<LastModified>2012-02-24T08:43:27.000Z</LastModified>
<ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
```

```
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>fun/test.jpg</Key>
<LastModified>2012-02-24T08:42:32.000Z</LastModified>
<ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
</Contents>
</ListBucketResult>
```

リクエストの例 (prefix と delimiter を含む場合):

```
GET /?prefix=fun/&delimiter=/ HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrnx7xHk3sgysx7I8U9I9IY1vY=
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<Name>oss-example</Name>
<Prefix>fun/</Prefix>
<Marker> </Marker>
<MaxKeys>100</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>fun/test.jpg</Key>
<LastModified>2012-02-24T08:42:32.000Z</LastModified>
<ETag>&amp;quot;5B3C1A2E053D763E1B002CC607C5A0FE&amp;quot;</ETag>
<Type>Normal</Type>
<Size>344606</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>00220120222</ID>
<DisplayName>user_example</DisplayName>
</Owner>
```

```
</Contents>
<CommonPrefixes>
<Prefix>fun/movie/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

GetBucketAcl

Get Bucket ACL

Get Bucket ACL を使用すると、バケットのアクセス権限を取得できます。

リクエスト構文

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

応答の要素

名前	説明
AccessControlList	ACL 情報の格納に使用されるコンテナー。 型: コンテナー 親ノード: AccessControlPolicy
AccessControlPolicy	Get Bucket ACL の結果の保存に使用されるコンテナー。 型: コンテナー 親ノード: なし
DisplayName	バケットオーナーの名前 (現在、この名前はバケットオーナー ID と同じです)。 型: 文字列 親ノード: AccessControlPolicy.Owner
Grant	バケットの ACL 権限。 型: 列挙文字列 有効な値: private、public-read、public-read-write 親ノード: AccessControlPolicy.AccessControlList

ID	バケットオーナーのユーザー ID。 型: 文字列 親ノード: AccessControlPolicy.Owner
Owner	バケットオーナーに関する情報の保存に使用されるコンテナー。 型: コンテナー 親ノード: AccessControlPolicy

詳細分析

1. Get Bucket ACL インターフェイスを使用できるのはバケットオーナーだけです。

例

リクエストの例:

```
GET /?acl HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 04:11:23 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0FmgbrQ0=
```

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 04:11:23 GMT
Content-Length: 253
Content-Tupe: application/xml
Connection: keep-alive
Server: AliyunOSS

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

GetBucketLocation

GetBucketLocation を使用すると、バケットが属するデータセンターの場所の情報を表示できます。

リクエスト構文

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

応答の要素

名前	説明
LocationConstraint	バケットがあるリージョンを指定します。 型: 文字列 有効な 値: oss-cn-hangzhou, oss-cn-qingdao, oss-cn-beijing, oss-cn-hongkong, oss-cn-shenzhen, and oss-cn-shanghai

詳細分析

- バケットの場所の情報を表示できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
- 現在、LocationConstraint の有効な値は、oss-cn-hangzhou、oss-cn-qingdao、oss-cn-beijing、oss-cn-hongkong、oss-cn-shenzhen、oss-cn-shanghai、oss-us-west-1、oss-us-east-1、oss-ap-southeast-1 です。これらの値は、それぞれ、杭州 (中国東部 1) 、青島 (中国北部 1) 、北京 (中国北部 2) 、香港、深セン (中国南部 1) 、上海 (中国東部 2) 、シリコンバレー (米国西部 1) 、バージニア (米国東部 1) 、シンガポールの各リージョンに相当します。

例

リクエストの例:

```
Get /?location HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 04 May 2012 05:31:04 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

リターンの例 (ログルールが既に設定されている場合):

```
HTTP/1.1 200
```

```
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 15 Mar 2013 05:31:04 GMT
Connection: keep-alive
Content-Length: 90
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">oss-cn-hangzhou</LocationConstraint >
```

GetBucketInfo

GetBucketInfo操作は、バケット情報を表示するために使用されます。 バケット情報には次のものが含まれます。

- 作成時間
- インターネットアクセスエンドポイント
- イントラネットアクセスエンドポイント
- バケット所有者情報
- バケットACL (AccessControlList)

リクエスト構文

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

レスポンス 要素

名前	説明
BucketInfo	バケット情報の内容を保存するコンテナ 型：container サブノード：Bucket node 親ノード：なし
Bucket	バケット固有情報を保存するコンテナ 型：container 親ノード：BucketInfoノード
CreationDate	バケットの作成時間。時間形式：2013-07-31T10:56:21.000Z 型：time

	親ノード：BucketInfo.Bucket	
ExtranetEndpoint	バケットにアクセスするインターネットドメイン名 型：string 親ノード：BucketInfo.Bucket	
IntranetEndpoint	同じリージョン内のECSからバケットにアクセスするためのイントラネットドメイン名 型：string 親ノード：BucketInfo.Bucket	
Location	バケットが配置されているデータセンターのリージョン 型：string 親ノード：BucketInfo.Bucket	
Name	バケット名 型：string 親ノード：BucketInfo.Bucket	
Owner	バケット所有者に関する情報を保存するコンテナ 型：container 親ノード：BucketInfo.Bucket	
ID	バケット所有者のユーザーID 型：string 親ノード ：BucketInfo.Bucket.Owner	
DisplayName	バケット所有者の名前（現在はIDと同じ） 型：string 親ノード ：BucketInfo.Bucket.Owner	
AccessControlList	ACL 情報を格納するコンテナ Type：container 親ノード：BucketInfo.Bucket	
Grant	バケットのACL権限 型：列挙型string 有効な値：private、public-read、public-read-write 親ノード ：BucketInfo.Bucket.AccessControlList	

詳細分析

1. バケットが存在しない場合、エラー 404 が返されます。 エラーコード：NoSuchBucket。
2. バケットの所有者だけがバケットの情報を見ることができます。 他のユーザーがバケット情報へアクセスしようとするとエラー 403 が返されます。 エラーコード：AccessDenied。
3. リクエストは任意のOSSエンドポイントから開始できます。

例

リクエストの例：

```
Get /?bucketInfo HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Sat, 12 Sep 2015 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39zXw=
```

リターンの例：(バケット情報が正常に取得される場合)

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 531
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<BucketInfo>
<Bucket>
<CreationDate>2013-07-31T10:56:21.000Z</CreationDate>
<ExtranetEndpoint>oss-cn-hangzhou.aliyuncs.com</ExtranetEndpoint>
<IntranetEndpoint>oss-cn-hangzhou-internal.aliyuncs.com</IntranetEndpoint>
<Location>oss-cn-hangzhou</Location>
<Name>oss-example</Name>
<Owner>
<DisplayName>username</DisplayName>
<ID>271834739143143</ID>
</Owner>
<AccessControlList>
<Grant>private</Grant>
</AccessControlList>
</Bucket>
</BucketInfo>
```

リターンの例：(バケット情報は存在しない場合)

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906009B
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error>
<Code>NoSuchBucket</Code>
<Message>The specified bucket does not exist.</Message>
```

```
<RequestId>568D547F31243C673BA14274</RequestId>
<HostId>nosuchbucket.oss.aliyuncs.com</HostId>
<BucketName>nosuchbucket</BucketName>
</Error>
```

リターンの例：(バケット情報へのアクセス権限がない場合)

```
HTTP/1.1 403
x-oss-request-id: 534B371674E88A4D8906008C
Date: Sat, 12 Sep 2015 07:51:28 GMT
Connection: keep-alive
Content-Length: 209
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error>
<Code>AccessDenied</Code>
<Message>AccessDenied</Message>
<RequestId>568D5566F2D0F89F5C0EB66E</RequestId>
<HostId>test.oss.aliyuncs.com</HostId>
</Error>
```

GetBucketLogging

Get Bucket Logging

Get Bucket Logging を使用すると、バケットのアクセスログ設定を表示できます。

リクエスト構文

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

応答の要素

名前	説明
BucketLoggingStatus	応答のコンテナー。 型: コンテナー 親要素: なし

LoggingEnabled	ログ情報のコンテナー。この要素とその子要素は、ログが有効になっている場合にのみ存在します。有効になっていない場合は存在しません。 型: コンテナー 親要素: BucketLoggingStatus
TargetBucket	この要素は、サーバーアクセスログが配信されるバケットを指定します。 型: 文字列 親要素: BucketLoggingStatus.LoggingEnabled
TargetPrefix	ログファイルが格納されるキーのプレフィックスを指定します。 型: 文字列 親要素: BucketLoggingStatus.LoggingEnabled

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. バケットのアクセスログ設定を表示できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
3. ソースバケットにログルールが設定されていない場合、XML メッセージの本文は返されますが、BucketLoggingStatus 要素が null になります。

例

リクエストの例:

```
Get /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 04 May 2012 05:31:04 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

戻り値の例 (ログルールが既に設定されている場合):

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 210
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<LoggingEnabled>
<TargetBucket>mybucketlogs</TargetBucket>
<TargetPrefix>mybucket-access_log/</TargetPrefix>
</LoggingEnabled>
```

```
</BucketLoggingStatus>
```

戻り値の例 (LOG ルールが設定されていない場合):

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 04 May 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 110
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
</BucketLoggingStatus>
```

GetBucketWebsite

Get Bucket Website

Get Bucket Website 操作を使用すると、バケットの静的 Web サイトホスティングステータスを表示できます。

リクエスト構文

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

応答の要素

名前	説明
ErrorDocument	Key 要素のコンテナー。 型: コンテナー 親: WebsiteConfiguration
IndexDocument	Suffix 要素のコンテナー。 型: コンテナー 親: WebsiteConfiguration
Key	4XX クラスエラーが発生した場合に使用するオブ

	ジェクトのキー名。 型: 文字列 親: WebsiteConfiguration.ErrorDocument 条件: ErrorDocument が指定されている場合は必須
Suffix	Web サイトエンドポイントのディレクトリに対するリクエストに追加されるサフィックス (たとえば、サフィックスが index.html である場合に、samplebucket/images/ に対するリクエストを送信すると、キー名が images/index.html であるオブジェクトのデータが返されます)。サフィックスを空にすることはできません。スラッシュを含めることもできません。 型: 文字列 親: WebsiteConfiguration.IndexDocument
WebsiteConfiguration	リクエストのコンテナー。 型: コンテナー 親: なし

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. バケットの静的 Web サイトホスティングステータスを表示できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
3. ソースバケットで静的 Web サイトホスティング機能が設定されていない場合は、エラー 404 が返されます。エラーコードは NoSuchWebsiteConfiguration です。

例

リクエストの例:

```
    Get /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39zXw=
```

戻り値の例 (ログルールが既に設定されている場合):

```
HTTP/1.1 200
x-oss-request-id: 50519080C4689A033D00235F
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: close
Content-Length: 218
Server: AliyunOSS

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

```
<WebsiteConfiguration xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<IndexDocument>
<Suffix>index.html</Suffix>
</IndexDocument>
<ErrorDocument>
<Key>error.html</Key>
</ErrorDocument>
</WebsiteConfiguration>
```

戻り値の例 (LOG ルールが設定されていない場合):

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:56:46 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<Error xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
<Code>NoSuchWebsiteConfiguration</Code>
<Message>The specified bucket does not have a website configuration.</Message>
<BucketName>oss-example</BucketName>
<RequestId>505191BEC4689A033D00236F</RequestId>
<HostId>oss-example.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
```

GetBucketReferer

Get Bucket Referer

Get Bucket Lifecycle 操作を使用すると、バケットのリファラー設定を表示できます。Bucket Referer Anti-leech Protection の詳細については、「OSS Anti-leech 保護」を参照してください。

リクエスト構文

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

応答の要素

名前	説明
RefererConfiguration	リファラー設定のコンテンツの保存に使用されるコンテナー。 型: コンテナー サブノード: AllowEmptyReferer ノード、RefererList ノード 親ノード: なし
AllowEmptyReferer	リファラーフィールドが null であるアクセスリクエストを許可するかどうかを指定します。 型: 列挙文字列 有効な値: true と false デフォルト値: true 親ノード: RefererConfiguration
RefererList	リファラーアクセスホワイトリストの保存に使用されるコンテナー。 型: コンテナー 親ノード: RefererConfiguration サブノード: Referer
RefererList	リファラーアクセスホワイトリストを指定します。 型: 文字列 親ノード: RefererList

詳細分析

1. バケットが存在しない場合は、エラー 404 が返されます。エラーコードは NoSuchBucket です。
2. バケットのリファラー設定を表示できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
3. バケットでリファラー設定が行われていない場合は、AllowEmptyReferer のデフォルト値と空の RefererList が返されます。

例

リクエストの例:

```
Get /?referer HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39zXw=
```

戻り値の例 (リファラールールが既に設定されている場合):

```
HTTP/1.1 200
```

```
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
Content-Length: 218
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
<RefererList>
<Referer> http://www.aliyun.com</Referer>
<Referer> https://www.aliyun.com</Referer>
<Referer> http://www.*.com</Referer>
<Referer> https://www.?.aliyuncs.com</Referer>
</RefererList>
</RefererConfiguration>
```

戻り値の例 (リファラールールが設定されていない場合):

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Thu, 13 Sep 2012 07:56:46 GMT
Connection: keep-alive
Content-Length: 308
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

GetBucketLifecycle

Get Bucket Lifecycle

Get Bucket Lifecycle を使用すると、バケットのライフサイクル設定を表示できます。

リクエスト構文

```
GET /?lifecycle HTTP/1.1
Host: BucketName.oss.aliyuncs.com
Date: GMT Date
```

```
Authorization: SignatureValue
```

詳細分析

1. バケットのライフサイクル設定を表示できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
2. バケットまたはライフサイクルが存在しない場合は、エラー 404 Not Found が返されます。エラーコードは NoSuchBucket または NoSuchLifecycle です。

例

リクエストの例:

```
Get /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:29 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

戻り値の例 (ライフサイクルが既に設定されている場合):

```
HTTP/1.1 200
x-oss-request-id: 534B371674E88A4D8906008B
Date: Mon, 14 Apr 2014 01:17:29 GMT
Connection: keep-alive
Content-Length: 255
Server: AliyunOSS

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
<Rule>
<ID>delete after one day</ID>
<Prefix>logs/</Prefix>
<Status>Enabled</Status>
<Expiration>
<Days>1</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

戻り値の例 (ライフサイクルが設定されていない場合):

```
HTTP/1.1 404
x-oss-request-id: 534B371674E88A4D8906008B
Date: Mon, 14 Apr 2014 01:17:29 GMT
Connection: keep-alive
Content-Length: 278
Server: AliyunOSS
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
<BucketName>oss-example</BucketName>
<Code>NoSuchLifecycle</Code>
<Message>No Row found in Lifecycle Table.</Message>
<RequestId>534B372974E88A4D89060099</RequestId>
<HostId> oss-example.oss.aliyuncs.com</HostId>
</Error>
```

DeleteBucket

Delete Bucket

Delete Bucket を使用すると、バケットを削除できます。

リクエスト構文

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

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. バケットを誤って削除することのないように、空でないバケットは削除できないようになっています。
3. 空でないバケットを削除しようとすると、エラー 409 Conflict が返されます。エラーコードは BucketNotEmpty です。
4. バケットを削除できるのはバケットオーナーだけです。権限のないバケットを削除しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。

例

リクエストの例:

```
DELETE / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:31:04 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:ceOEyZavKY4QcjoUWYSpYbJ3naA=
```

戻り値の例:

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:31:04 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

DeleteBucketLogging

Delete Bucket Logging

Delete Bucket Logging 操作を使用すると、バケットのアクセスログ機能を無効にすることができます。

リクエスト構文

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

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. バケットのアクセスログ機能を無効にできるのはバケットオーナーだけです。自分のものではないバケットを操作しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
3. ターゲットバケットでアクセスログ機能が有効になっていない場合は、HTTP ステータスコード 204 が返されます。

例

リクエストの例

```
DELETE /?logging HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:35:24 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/CnMZU=
```

戻り値の例

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:35:24 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

DeleteBucketWebsite

Delete Bucket Website

Delete Bucket Website 操作を使用すると、バケットの静的 Web サイトホスティングモードを無効にすることができます。

リクエスト構文

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

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. バケットの静的 Web サイトホスティングモードを無効にすることができるのはバケットオーナー

だけです。自分のものではないバケットを操作しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。

例

リクエストの例:

```
DELETE /?website HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWicOMEkVg=
```

戻り値の例:

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

DeleteBucketLifecycle

DeleteBucketLifecycle を使用すると、指定したバケットのライフサイクル設定を削除できます。

リクエスト構文

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

詳細分析

この操作では、指定したバケットのライフサイクルルールがすべて削除されます。それ以降、そのバケットのオブジェクトは自動的に削除されません。

バケットのライフサイクル設定を削除できるのはバケットオーナーだけです。自分のものではないバケットを操作しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。

例

リクエストの例:

```
DELETE /?lifecycle HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Date: Mon, 14 Apr 2014 01:17:35 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:6ZVHOehYzxoC1yxRydPQs/CnMZU=
```

戻り値の例:

```
HTTP/1.1 204 No Content
x-oss-request-id: 534B371674E88A4D8906008B
Date: Mon, 14 Apr 2014 01:17:35 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

オブジェクトの操作

PutObject

Put Object

Put Object は、ファイルをアップロードするために使用します。

リクエスト構文

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

```
Content-Type: ContentType
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Date: GMT Date
Authorization: SignatureValue
```

リクエストヘッダー

名前	説明
Cache-Control	オブジェクトがダウンロードされる際の Web ページのキャッシュ動作を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Disposition	オブジェクトがダウンロードされる際のオブジェクトの名前を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Encoding	オブジェクトがダウンロードされる際のコンテンツのエンコーディング形式を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-MD5	RFC 1864 で定義されているように、メッセージの内容 (ヘッダーを除く) が計算されて 128 ビットの数値の MD5 値が取得されます。その後、base64 を使用してこの数値が Content-MD5 値にエンコードされます。このリクエストヘッダーは、メッセージの有効性、つまりメッセージの内容が送信された内容と一致しているかどうかを確認するために使用できます。このリクエストヘッダーはオプションですが、OSS ではこのリクエストヘッダーをエンドツーエンドのチェックに使用することが推奨されます。 型: 文字列 デフォルト値: なし 制約: なし
Expires	有効期限をミリ秒単位で指定します。詳細については、『RFC2616』を参照してください。 型: 整数 デフォルト値: なし
x-oss-server-side-encryption	OSS によってオブジェクトが作成される際のサーバー側の暗号化アルゴリズムを指定します。 型: 文字列 有効な値: AES256
x-oss-object-acl	OSS によってオブジェクトが作成される際のアクセス権限を指定します。 型: 文字列 有効な値: public-read、private、public-read-write

詳細分析

- Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。
- リクエストヘッダーの Content-Length の値が実際のリクエスト本文で送信されるデータの長さを下回る場合でも、オブジェクトは作成されますが、オブジェクトサイズは Content-Length で定義されているサイズと同じになり、残りのデータは削除されます。
- 追加するオブジェクトのファイルが既に存在する場合にこのオブジェクトへのアクセス権限を持っていると、既存のファイルが新しく追加したファイルによって上書きされ、200 OK メッセージが返されます。
- PutObject リクエストに x-oss-meta- というプレフィックスが付いたパラメーターが含まれている場合、このパラメーターはユーザーメタ (x-oss-meta-location など) として扱われます。1 つのオブジェクトに対して同様のパラメーターを複数設定することはできますが、すべてのユーザーメタの合計サイズは 8 KB 以下にする必要があります。
- コンテンツの長さパラメーターがヘッダーに追加されていない場合は、411 Length Required メッセージが返されます。エラーコードは MissingContentLength です。
- 長さは設定されているがメッセージ本文が送信されない場合、または送信された本文のサイズが指定したサイズよりも小さい場合は、タイムアウト後に 400 Bad Request メッセージが返されます。エラーコードは RequestTimeout です。このとき、OSS にあるこのファイルの内容はアップロードしたデータになっています。
- 追加するオブジェクトのバケットが存在しない場合は、404 Not Found メッセージが返されます。エラーコードは NoSuchBucket です。
- 追加するオブジェクトのバケットにアクセスする権限がない場合は、403 Forbidden メッセージが返されます。エラーコードは AccessDenied です。
- 追加するファイルの長さが 5 GB を超える場合は、400 Bad Request メッセージが返されます。エラーコードは InvalidArgument です。
- 受け取ったオブジェクトキーの長さが 1023 ビットを超える場合は、400 Bad Request メッセージが返されます。エラーコードは InvalidObjectName です。
- オブジェクトをアップロードする際、OSS では、『RFC2616』で定義されている Cache-Control、Expires、Content-Encoding、および Content-Disposition の 4 つのヘッダーフィールドがサポートされます。オブジェクトのアップロード時にこれらのヘッダーを設定した場合は、次にこのオブジェクトがダウンロードされるときに、対応するヘッダー値がアップロードされた値に自動的に設定されます。
- オブジェクトのアップロード時に x-oss-server-side-encryption ヘッダーを指定する場合、このヘッダーの値は AES256 に設定する必要があります。それ以外に設定した場合は、400 メッセージとエラーコード InvalidEncryptionAlgorithmError が返されます。このヘッダーを指定すると、応答ヘッダーにもこのヘッダーが含まれ、アップロードされたオブジェクトの暗号化アルゴリズムが示されます。このオブジェクトのダウンロード時にも、応答ヘッダーには、値がこのオブジェクトの暗号化アルゴリズムに設定されている x-oss-server-side-encryption が含まれます。

FAQ

Content-MD5計算方法エラー

アップロードコンテンツ「0123456789」が例として使用されます。文字列のContent-MD5値が計算されます。

正しい計算方法

関連する標準で定義されているアルゴリズムは、以下のプロセスでContent-MD5を計算するために使用されます。

1. 値が128ビットのバイナリ配列であるアップロードContent-MD5を計算します。
2. バイナリ配列をBase64でエンコードします。

Pythonが例として使用されています。正しい計算コードは次のとおりです。

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

注意： hash.digest()の結果は、Base64エンコーディングの入力として使用されます。

```
>>> hash.digest()
'x\x1e^$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```

誤った計算方法

よくある間違いは、計算された32バイトの MD5文字列をBase64でエンコードすることです。不正な例：Base64で hash.hexdigest()をエンコードした場合：

```
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
>>>
>>> #次は不正確な計算結果です。
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

確認方法

次の方法を使用して確認できます。

1。最後に計算されたContent-MD5は、24バイトの表示可能な文字列です。

2。コンテンツとして “0123456789” を使用して、Content-MD5が eB5eJF1ptWaXm4bijSPyxw ==であるかどうかを確認します。

例

リクエストの例:

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

[344606 bytes of object data]
```

レスポンスの例:

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

CopyObject

Copy Object

Copy Object は、OSS の既存のオブジェクトを別のオブジェクトにコピーするために使用します。OSS に PUT リクエストを送信し、PUT リクエストヘッダーに要素 “x-oss-copy-source” を追加して、コピー元を指定できます。OSS ではこれが Copy Object 操作であると自動的に判別され、この操作がサーバー側で直接実行されます。Copy Object 操作が正常に完了すると、新しいオブジェクトに関する情報が返されます。この操作は 1 GB 未満のオブジェクトに適用されます。1 GB を超えるオブジェクトをコピーするには、マルチパートアップロード操作を使用する必要があります。この操作の詳細については、「Upload Part Copy」を参照してください。

注意：CopyObjectの操作では、ソースバケットとターゲットバケットが同じリージョンである必要が

あります 。

リクエスト構文

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

リクエストヘッダー

名前	説明
x-oss-copy-source	コピー元のソースアドレスを指定します (リクエストするには、ソースオブジェクトの読み取り権限が必要です)。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-match	ソースオブジェクトの ETAG 値とユーザーが提示した ETAG 値が一致する場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-none-match	ユーザーが指定した日時以降にソースオブジェクトが変更されていない場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-unmodified-since	受け取ったパラメーターによって指定された日時がファイルの変更日時と同じかそれ以降である場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-modified-since	ユーザーが指定した日時以降にソースオブジェクトが変更されている場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-metadata-directive	有効な値には COPY と REPLACE があります。このパラメーターが COPY に設定されている場合は、新しいオブジェクトのメタがソースオブジェクトからコピーされます。このパラメーターが REPLACE に設定されている場合は、ソースオブ

	ジェクトのすべてのメタ値が無視され、このリクエストで指定したメタ値が使用されます。このパラメーターが COPY および REPLACE 以外の値に設定されている場合は、400 Bad Request メッセージが返されます。このパラメーターが COPY に設定されている場合でも、ソースオブジェクトの x-oss-server-side-encryption のメタ値はコピーされません。 型: 文字列 デフォルト値: COPY 有効な値: COPY、REPLACE
x-oss-server-side-encryption	OSS によってターゲットオブジェクトが作成される際のサーバー側の暗号化アルゴリズムを指定します。 型: 文字列 有効な値: AES256またはKMS **注意**：KMS暗号化アルゴリズムを使用するには、コンソールでKMS（キー管理サービス）を有効にする必要があります。 それ以外の場合は、KmsServiceNotenabledエラーコードが報告されます。
x-oss-object-acl	OSS によってオブジェクトが作成される際のアクセス権限を指定します。 型: 文字列 有効な値: public-read、private、public-read-write

応答の要素

名前	説明
CopyObjectResult	Copy Object の結果。 型: 文字列 デフォルト値: なし
ETag	新しいオブジェクトの ETag 値。 型: 文字列 親要素: CopyObjectResult
LastModified	新しいオブジェクトの最終更新日時。 型: 文字列 親要素: CopyObjectResult

詳細分析

- Copy Object 操作を使用して、既存のオブジェクトのメタ情報を変更できます。
- Copy Object 操作でソースオブジェクトのアドレスがターゲットオブジェクトのアドレスと同じ場合は、x-oss-metadata-directive の値に関係なく、ソースオブジェクトのメタ情報が直接置き換えられます。
- Copy Object リクエストには、4 つの事前判断ヘッダーをいくつでも含めることができます。関連

するロジックの詳細については、Get Object の「詳細分析」を参照してください。
- Copy Object 操作を実行するには、ソースオブジェクトの読み取り権限が必要です。
- ソースオブジェクトとターゲットオブジェクトは同じデータセンターに属している必要があります。属していない場合は、403 AccessDenied メッセージと、” Target object does not reside in the same data center as source object” というエラー情報が返されます。
- Copy Object 操作の課金統計では、ソースオブジェクトのバケット内で Get リクエストの数が 1 ずつ増え、ターゲットオブジェクトのバケット内で Put リクエストの数が 1 ずつ増え、それに応じてストレージ容量が追加されます。
- Copy Object 操作では、関連するリクエストヘッダーはすべて x-oss- で始まるため、署名文字列に追加する必要があります。
- Copy Object リクエストで x-oss-server-side-encryption ヘッダーを指定した場合に、その値 (AES256) が有効であれば、サーバー側でソースオブジェクトが暗号化されているかどうかに関係なく、Copy Object 操作の実行後にサーバー側でターゲットオブジェクトが暗号化されます。また、Copy Object 応答ヘッダーには、値がターゲットオブジェクトの暗号化アルゴリズムに設定されている x-oss-server-side-encryption が含まれます。このターゲットオブジェクトのダウンロード時にも、応答ヘッダーには、値がこのターゲットオブジェクトの暗号化アルゴリズムに設定されている x-oss-server-side-encryption が含まれます。Copy Object 操作で x-oss-server-side-encryption リクエストヘッダーを指定しない場合は、サーバー側でソースオブジェクトが暗号化されているかどうかに関係なく、ターゲットオブジェクトはサーバー側で暗号化されないデータになります。
- Copy Object リクエストの x-oss-metadata-directive ヘッダーが COPY (デフォルト値) に設定されている場合でも、ソースオブジェクトの x-oss-server-side-encryption の値はコピーされません。つまり、ターゲットオブジェクトがサーバー側で暗号化されるのは、Copy Object リクエストで x-oss-server-side-encryption を指定した場合だけです。
- Copy Object リクエストで x-oss-server-side-encryption ヘッダーを指定したが、その値が AES256 でない場合は、400 メッセージとエラーコード InvalidEncryptionAlgorithmError が返されます。
- コピーするファイルのサイズが 1 GB を超える場合は、400 メッセージとエラーコード EntityTooLarge が返されます。
- この操作では、追加されたアップロードメソッドで生成されたオブジェクトをコピーできません。
- ファイルタイプが **シンボリックリンク** の場合、シンボリックリンクのみがコピーされます。

例

リクエストの例:

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

戻り値の例:

```
```

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
Content-Type: application/xml
Content-Length: 193
Connection: keep-alive
Date: Fri, 24 Feb 2012 07:18:48 GMT
Server: AliyunOSS

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

GetObject

Get Object は、オブジェクトを取得するために使用します。使用するには、このオブジェクトの読み取り権限が必要です。

リクエスト構文

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

リクエストパラメーター

OSSでは、GET要求を送信するときに、応答要求内のいくつかのヘッダーをカスタマイズできます。匿名ユーザは content-typeだけをカスタマイズすることができます。他のヘッダーの場合、GETリクエストは署名付きで送信する必要があります。これらのヘッダーは、以下となります。

名前	説明
response-content-type	OSS から返されるリクエストの content-type ヘッダーを指定します。匿名ユーザーは、コンテンツタイプヘッダーを指定することもできます。 型: 文字列 デフォルト値: なし
response-content-language	OSS から返されるリクエストの content-language ヘッダーを指定します。 型: 文字列 デフォルト値: なし

response-expires	OSS から返されるリクエストの expires ヘッダーを指定します。 型: 文字列 デフォルト値: なし
response-cache-control	OSS から返されるリクエストの cache-control ヘッダーを指定します。 型: 文字列 デフォルト値: なし
response-content-disposition	OSS から返されるリクエストの content-disposition ヘッダーを指定します。 型: 文字列 デフォルト値: なし
response-content-encoding	OSS から返されるリクエストの content-encoding ヘッダーを指定します。 型: 文字列 デフォルト値: なし

リクエストヘッダー

名前	説明
Range	ファイル転送の範囲を指定します。たとえば、範囲を bytes=0-9 に設定した場合は、バイト 0 ~ 9 が転送されます。 型: 文字列 デフォルト値: なし
If-Modified-Since	指定した日時が実際の変更日時より前である場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、304 Not Modified メッセージが返されます。 型: 文字列 デフォルト値: なし 時間形式: GMT, 例: Fri, 13 Nov 2015 14:47:53 GMT
If-Unmodified-Since	受け取ったパラメーターによって指定された日時がファイルの変更日時と同じかそれ以降である場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし 時間形式: GMT, 例: Fri, 13 Nov 2015 14:47:53 GMT
If-Match	受け取った ETag とオブジェクトの ETag が一致する場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
If-None-Match	受け取った ETag とオブジェクトの ETag が一致

	しない場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、304 Not Modified メッセージが返されます。 型: 文字列 デフォルト値: なし

詳細分析

1. Get Object リクエストの Range パラメーターを設定すると、ブレークポイントから再開可能なデータ転送をサポートできます。この機能はオブジェクトサイズが大きい場合に推奨されます。
2. リクエストヘッダーで Range パラメーターを指定した場合は、返されるメッセージにファイル全体の長さと今回返された範囲が示されます。たとえば、Content-Range: bytes 0-9/44 は、ファイル全体の長さが 44 で今回返された範囲が 0 ~ 9 であることを示しています。範囲の要件が満たされていない場合は、ファイル全体が転送され、結果に Content-Range は示されません。
3. If-Modified-Since によって指定された日時と実際の変更日時が一致しない場合は、ファイルが直接返されて 200 OK メッセージが返されます。
4. If-Modified-Since と If-Unmodified-Since は併用できます。If-Match と If-None-Match も併用できます。
5. リクエストに If-Unmodified-Since が含まれており、If-Unmodified-Since と実際の変更日時が一致しない場合、またはリクエストに If-Match が含まれており、If-Match とオブジェクトの ETag が一致しない場合は、412 Precondition Failed メッセージが返されます。
6. リクエストに If-Modified-Since が含まれており、If-Modified-Since と実際の変更日時が一致しない場合、またはリクエストに If-None-Match が含まれており、If-None-Match とオブジェクトの ETag が一致しない場合は、304 Not Modified メッセージが返されます。
7. ファイルが存在しない場合は、404 Not Found メッセージが返されます。エラーコードは NoSuchKey です。
8. OSS では、匿名アクセス中に GET Object リクエストのリクエストパラメーターを使用して、OSS から返されるリクエストのヘッダーをカスタマイズすることはできません。
9. OSS から返されるリクエストのヘッダーをカスタマイズしたときに、それらのヘッダーが GET Object リクエストのパラメーターで指定した値に設定されるのは、リクエストが正常に処理された場合、つまり 200 OK メッセージが返された場合だけです。
10. サーバー側でこのオブジェクトが暗号化されている場合は、GET Object リクエストの受信時に暗号化が解除されたオブジェクトが自動的に返されて、応答ヘッダーで x-oss-server-side-encryption が返されます。x-oss-server-side-encryption の値は、オブジェクトのサーバー側の暗号化アルゴリズムを示します。
11. 返されるコンテンツを GZIP で圧縮して転送する必要がある場合は、リクエストヘッダーの表示モードに Accept-Encoding:gzip を追加する必要があります。OSS では、ファイルのコンテンツタイプとサイズに基づいて、GZIP で圧縮されたデータを返すかどうかが判別されます。コンテンツが GZIP で圧縮されている場合、そのコンテンツには Etag が含まれません。現在、OSS では、HTML、Javascript、CSS、XML、RSS、および Json のコンテンツタイプに対する GZIP 圧縮がサポートされています。ファイルサイズは 1 KB 以上である必要があります。
12. ファイルタイプは **シンボリックリンク** の場合、ターゲットファイルのコンテンツが戻ってきます。また、レスポンスヘッダの Content-Length、ETag、Content-Md5 パラメータは、対象フ

ファイルのメタ情報となります。 Last-Modified パラメータはターゲットファイルとシンボリックリンクの最大値になります。 他のすべてのパラメータは、シンボリックリンクのメタ情報になります。

13. ファイルタイプは **シンボリックリンク** で、ターゲットファイルが存在しない場合、404 Not Found メッセージが返されます。エラーコードは SymlinkTargetNotExist です。
14. ファイルタイプは **シンボリックリンク** で、ターゲットファイルもシンボリックリンクの場合、400 Bad Request error メッセージが返されます。エラーコードは InvalidTargetType です。

アーカイブタイプは、オブジェクトがダウンロードする前に、リストア要求とリストア完了の設定が可能です。リストア操作の時のみダウンロードされるオブジェクトは、タイムアウトせずに完了します。

- 要求が設定されておらず、最後のリストア操作がタイムアウトした場合、403 エラーとなります。エラーコードは “InvalidObjectState” です。
- 要求が設定されており、リストア操作が完了しなかった場合、403 エラーとなります。エラーコードは “InvalidObjectState” です。
- リストア操作が完了し、タイムアウトが発生しなかった場合のみ、データは直接ダウンロードされます。

例

リクエストの例:

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

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 3a89276f-2e2d-7965-3ff9-51c875b99c41
x-oss-object-type: Normal
Date: Fri, 24 Feb 2012 06:38:30 GMT
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
ETag: "7DCA4FDCA3F27655940C866D52B04C39"
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS

[344606 bytes of object data]
```

Range を指定したリクエストの例:

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

```
Range: bytes=100-900
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:qZzjF3DUtd+yK16BdhGtFcCVknM=
```

戻り値の例:

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

[801 bytes of object data]
```

返されるメッセージヘッダーをカスタマイズしたリクエストの例:

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

戻り値の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 1144d124-055c-4052-2c65-a1e3439d41c1
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: "7DCA4FDCA3F27655940C866D52B04C39"
Content-Length: 344606
Connection: close
Content-disposition: attachment; filename:testing.txt
Content-language: Chinese
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]
```

シンボリックリンクのリクエストの例:

```
GET /link-to-oss.jpg HTTP/1.1
```

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

戻り値の例:

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

アーカイブ型オブジェクトのリストア操作におけるリクエストの例:

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

戻り値の例:

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

[354606 bytes of object data]
```

AppendObject

Append Object

Append Object は、ファイルを追加モードでアップロードするために使用します。Append Object 操作で作成されるタイプのオブジェクトは追加可能オブジェクトで、Put Object 操作でアップロードされるタイプのオブジェクトは標準オブジェクトです。

リクエスト構文

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

リクエストヘッダー

名前	説明
Cache-Control	オブジェクトがダウンロードされる際の Web ページのキャッシュ動作を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Disposition	オブジェクトがダウンロードされる際のオブジェクトの名前を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Encoding	オブジェクトがダウンロードされる際のコンテンツのエンコーディング形式を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-MD5	RFC 1864 で定義されているように、メッセージの内容 (ヘッダーを除く) が計算されて 128 ビットの数値の MD5 値が取得されます。その後、base64 を使用してこの数値が Content-MD5 値にエンコードされます。このリクエストヘッダーは、メッセージの有効性、つまりメッセージの内容が送信された内容と一致しているかどうかを確認するために使用できます。このリクエストヘッダーはオプションですが、OSS ではこのリクエストヘッダーをエンドツーエンドのチェックに使用することが推奨されます。 型: 文字列 デフォルト値: なし 制約: なし

Expires	有効期限をミリ秒単位で指定します。詳細については、『RFC2616』を参照してください。 型: 整数 デフォルト値: なし
x-oss-server-side-encryption	OSS によってオブジェクトが作成される際のサーバー側の暗号化アルゴリズムを指定します。 型: 文字列 有効な値: AES256
x-oss-object-acl	OSS によってオブジェクトが作成される際のアクセス権限を指定します。 型: 文字列 有効な値: public-read、private、public-read-write

応答ヘッダー

名前	説明
x-oss-next-append-position	次のリクエストで提示する必要がある位置を指定します。これは、実際には現在のオブジェクトの長さです。このヘッダーは、Append Object に対して成功メッセージが返された場合、または位置とオブジェクトの長さが一致しないために 409 エラーが発生した場合に含まれます。 型: 64 桁の整数
x-oss-hash-crc64ecma	オブジェクトの 64 ビットの CRC 値を指定します。この 64 ビットの CRC は、ECMA-182 に従って計算されます。 型: 64 桁の整数

他の操作との関連性

- Append Object は、追加可能オブジェクト以外には適用されません。たとえば、同じ名前の標準オブジェクトが既に存在する場合に Append Object 操作を実行すると、409 メッセージとエラーコード ObjectNotAppendable が返されます。
- 既存の追加可能オブジェクトに対して Put Object 操作を実行すると、この追加可能オブジェクトは新しいオブジェクトで上書きされ、このオブジェクトのタイプが標準オブジェクトに変更されます。
- Head Object 操作を実行すると、オブジェクトのタイプを示す x-oss-object-type が返されます。オブジェクトが追加可能オブジェクトである場合、x-oss-object-type の値は Appendable です。追加可能オブジェクトの場合は、Head Object 操作を実行すると、x-oss-next-append-position と x-oss-hash-crc64ecma も返されます。
- Get Bucket (List Object) リクエストの応答 XML では、追加可能オブジェクトのタイプは Appendable に設定されています。
- Copy Object を使用して追加可能オブジェクトをコピーすることも、このオブジェクトのサーバー側の暗号化属性を変更することもできません。ただし、Copy Object を使用してカスタマイズされ

たメタデータを変更することはできます。

詳細分析:

- 2 つの URL パラメーター append および position はどちらも CanonicalizedResource で、署名に含まれている必要があります。
- URL パラメーターには、操作が Append Object 操作であることを指定する append を含める必要があります。
- URL クエリパラメーターには、追加の開始位置を指定する position を含める必要があります。最初の Append Object 操作の position の値は 0 にする必要があり、その後の操作の position の値は現在のオブジェクトの長さです。たとえば、最初の Append Object リクエストで指定した position の値が 0 で、content-length の値が 65536 の場合、2 番目の Append Object リクエストで指定する position の値は 65536 に設定する必要があります。各操作が正常に完了すると、応答ヘッダーの x-oss-next-append-position にも次の Append Object リクエストの位置が示されます。
- position の値と現在のオブジェクトの長さが異なる場合は、409 メッセージとエラーコード PositionNotEqualToLength が返されます。このようなエラーが発生した場合は、次の Append Object リクエストの位置を応答ヘッダーの x-oss-next-append-position から取得して、Append Object リクエストを再送信することができます。
- position の値が 0 で同じ名前の追加可能オブジェクトが存在しない場合、または同じ名前の追加可能オブジェクトの長さが 0 の場合は、Append Object 操作が正常に実行されます。それ以外の場合は、位置とオブジェクトの長さが一致しないと見なされます。
- position の値が 0 で同じ名前のオブジェクトが存在しない場合は、Put Object リクエストと同様に、Append Object リクエストでヘッダー (x-oss-server-side-encryption など) を設定できます。これは Initiate Multipart Upload の場合と同じです。position の値が 0 で、正しい x-oss-server-side-encryption ヘッダーがリクエストに追加されている場合は、その後の Append Object リクエストに対する応答のヘッダーにも、暗号化アルゴリズムを示す x-oss-server-side-encryption が含まれます。後でメタの変更が必要になった場合は、Copy Object リクエストを使用できます。
- 並行処理が原因で、position の値を x-oss-next-append-position に設定した場合でも、このリクエストが PositionNotEqualToLength により失敗する可能性があります。
- Append Object によって生成されるオブジェクトの長さの制限は、Put Object によって生成されるオブジェクトの長さの制限と同じです。
- 各 Append Object 操作の実行後に、このオブジェクトの最終変更日時が更新されます。
- position の値が正しい場合、長さが 0 のコンテンツを既存の追加可能オブジェクトに追加する操作では、オブジェクトのステータスは変更されません。
- バケットタイプがアーカイブでこのインターフェイスを呼び出すことができないときは、システムはエラーコード 「400」 “OperationNotSupported” を返します。

CRC64 の計算方法

追加可能オブジェクトの CRC は、ECMA-182 に従って計算されます。その計算方法は XZ の計算方法と同

じです。CRC64 は、boost CRC モジュールを使用して次のように計算できます。

```
typedef boost::crc_optimal<64, 0x42F0E1EBA9EA3693ULL, 0xffffffffffffffffULL, 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();
}
```

また、Python crcmod を使用して次のように計算することもできます。

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

print do_crc64("123456789")
```

例

リクエストの例:

```
POST /oss.jpg?append&position=0 HTTP/1.1
Host: oss-example.oss.aliyuncs.com
Cache-control: no-cache
Expires: Wed, 08 Jul 2015 16:57:01 GMT
Content-Encoding: utf-8
Content-Disposition: attachment;filename=oss_download.jpg
Date: Wed, 08 Jul 2015 06:57:01 GMT
Content-Type: image/jpg
Content-Length: 1717
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=

[1717 bytes of object data]
```

応答例:

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

DeleteObject

DeleteObject はオブジェクトを削除するために使用します。

リクエスト構文

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

詳細分析

- DeleteObject でオブジェクトを削除するには、このオブジェクトの書き込み権限が必要です。
- 削除するオブジェクトが存在しない場合は、204 No Content メッセージが返されます。
- オブジェクトのバケットが存在しない場合は、404 Not Found メッセージが返されます。
- ファイルタイプは **シンボリックリンク** の場合、シンボリックリンクだけが削除されます。

例

リクエストの例:

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

応答の例:

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

DeleteMultipleObjects

DeleteMultipleObjects を使用すると、同じバケット内の複数のオブジェクトを 1 回の HTTP リクエストで削除できます。1 回の DeleteMultipleObjects 操作で削除できるオブジェクトの最大数は 1,000 で、詳細モードと出力抑制モードの 2 つの応答モードが用意されています。

- 詳細モード: OSS から返されるメッセージ本文には、各オブジェクトの削除結果が表示されます。
- 出力抑制モード: OSS から返されるメッセージ本文には、削除エラーが発生したオブジェクトの結果のみが表示されます。すべてのオブジェクトが正常に削除された場合、メッセージ本文は返されません。

リクエスト構文

```
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>
```

リクエストパラメータ

複数のオブジェクトの削除操作中に、エンコードタイプを使用して、返された結果にキーをエンコードすることができます。

名前	説明
encoding-type	返された結果に、キーのエンコーディングタイプを指定します。現在、URLエンコーディングがサポートされています。 KeyはUTF-8エンコーディングを採用します。 型: string デフォルト値: なし オプションの値: url

リクエストの要素

名前	説明

Delete	DeleteMultipleObjects リクエストを保存するコンテナーを指定します。 型: コンテナー サブノード: 1 つ以上の object 要素とオプションの quiet 要素 親ノード: なし
Key	削除するオブジェクトの名前を指定します。 型: string 親ノード: Object
Object	オブジェクトに関する情報を保存するコンテナーを指定します。 型: コンテナー サブノード: Key 親ノード: Delete
Quiet	出力抑制モードを有効または無効にします。 型: 列挙文字列 有効な値: True、False デフォルト値: False 親ノード: Delete
encoding-type	返されるコンテンツのエンコーディングとエンコーディングのタイプを指定します。オブジェクトキーでは UTF-8 文字が使用されますが、XML 1.0 標準では、0 ~ 10 の ASCII 値を含む文字などの特定の制御文字の解析はサポートされていません。XML 1.0 標準でサポートされていない制御文字がオブジェクトキーに含まれている場合は、encoding-type を指定して返されるオブジェクトキーをエンコードできます。 データ型: string デフォルト値: なし

応答の要素

名前	説明
Deleted	正常に削除されたオブジェクトを保存するコンテナーを指定します。 型: コンテナー サブノード: Key 親ノード: DeleteResult
DeleteResult	DeleteMultipleObjects リクエストの結果を保存するコンテナーを指定します。 型: コンテナー サブノード: Deleted 親ノード: なし
Key	OSS によって削除操作が実行されたオブジェクトの名前を指定します。 型: string 親ノード: Deleted
Encoding-type	返される結果におけるエンコーディングのタイプを指定します。リクエストで encoding-type を

	指定した場合は、返される結果において Key がエンコードされます。 型: string 親ノード: Container

詳細分析

1. DeleteMultipleObjects リクエストでは Content-Length フィールドと Content-MD5 フィールドを指定する必要があります。OSS では、受信したメッセージ本文が正しいことがこの 2 つのフィールドに基づいて検証され、削除操作が実行されます。
2. Content-MD5 フィールドの内容を生成するには、MD5 を使用して DeleteMultipleObjects リクエストを暗号化し、128 バイトの配列を取得して、Base64 を使用してその配列をエンコードします。最終的に取得した文字列が Content-MD5 フィールドの内容になります。
3. DeleteMultipleObjects リクエストの応答モードは、デフォルトでは詳細モードです。
4. DeleteMultipleObjects リクエストを使用して存在しないオブジェクトを削除した場合でも、その操作は成功と見なされます。
5. DeleteMultipleObjects リクエストには、最大 2 MB のメッセージ本文を含めることができます。メッセージ本文のサイズが 2 MB を超える場合は、MalformedXML エラーコードが返されます。
6. DeleteMultipleObjects リクエストでは、一度に最大 1,000 件のオブジェクトを削除できます。一度に削除するオブジェクト数が 1,000 を超える場合は、MalformedXML エラーコードが返されます。
7. Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

リクエストの例 I:

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:+z3gBfnFAxBcBDgx27Y/jEfbfu8=

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

<Delete>
<Quiet>false</Quiet>
<Object>
<Key>multipart.data</Key>
</Object>
<Object>
<Key>test.jpg</Key>
</Object>
```

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

応答例:

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

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

リクエストの例 II:

```
POST /?delete HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length:151
Content-MD5: ohhnqLBJFiKkPSBO1eNaUA==
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:WuV0Jks8RyGSNQrBca64kEExJDs=

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

<Delete>
<Quiet>true</Quiet>
<Object>
<Key>multipart.data</Key>
</Object>
<Object>
<Key>test.jpg</Key>
</Object>
<Object>
<Key>demo.jpg</Key>
</Object>
</Delete>
```

応答例:

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

HeadObject

Head Object はファイルの内容を返さずに特定のオブジェクトのメタ情報を返すために使用します。

リクエスト構文

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

リクエストヘッダー

名前	説明
If-Modified-Since	指定した日時が実際の変更日時より前である場合は、200 OK メッセージとオブジェクトのメタが返されます。それ以外の場合は、304 Not Modified メッセージが返されます。 型: string デフォルト値: なし
If-Unmodified-Since	受け取ったパラメーターによって指定された日時がファイルの実際の変更日時と同じかそれ以降である場合は、200 OK メッセージとオブジェクトのメタが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: string デフォルト値: なし
If-Match	受け取った ETag とオブジェクトの ETag が一致する場合は、200 OK メッセージとオブジェクトのメタが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: string デフォルト値: なし
If-None-Match	受け取った ETag とオブジェクトの ETag が一致しない場合は、200 OK メッセージとオブジェク

	トのメタが返されます。それ以外の場合は、304 Not Modified メッセージが返されます。 型: string デフォルト値: なし

詳細分析

- Head Object リクエストの送信後は、200 OK メッセージが返されるかエラーメッセージが返されるかに関係なく、メッセージ本文は返されません。
- Head Object リクエストのヘッダーでは、If-Modified-Since、If-Unmodified-Since、If-Match、If-None-Match の各クエリ条件を設定できます。設定ルールの詳細については、Get Object リクエストの関連フィールドを参照してください。変更が加えられていない場合は、304 Not Modified メッセージが返されます。
- Put Object リクエストの送信時に x-oss-meta- というプレフィックスが付いたユーザーメタ (x-oss-meta-location など) をアップロードした場合は、そのユーザーメタが返されます。
- ファイルが存在しない場合は、404 Not Found メッセージが返されます。
- サーバー側でこのオブジェクトのエントロピ暗号化が行われている場合は、Head Object リクエストに対する応答のヘッダーで x-oss-server-side-encryption が返されます。x-oss-server-side-encryption の値は、オブジェクトのサーバー側の暗号化アルゴリズムを示します。
- ファイルタイプが**シンボリックリンク**の場合、レスポンスヘッダの Content-Length、ETag、Content-Md5パラメータは、対象ファイルのメタ情報となります。 Last-Modifiedパラメータはターゲットファイルとシンボリックリンクの最大値になります。 他のすべてのパラメータは、シンボリックリンクのメタ情報になります。
- ファイルタイプが**シンボリックリンク**で、ターゲットファイルが存在しない場合、システムは404 Not Foundエラーを返します。 エラーコード：SymlinkTargetNotExist。

ファイルタイプが**シンボリックリンク**で、ターゲットファイルタイプがシンボリックリンクの場合、システムは400 Bad Requestエラーを返します。 エラーコード：InvalidTargetType。

バケットタイプがアーカイブの場合、オブジェクトのストレージタイプは応答ヘッダーにx-oss-storage-classで示されます。

バケットタイプがアーカイブで、リストア要求がサブミットされている場合、オブジェクトのリストア状態は、応答ヘッダーにx-oss-restoreによって示されます。</ br>

- リストア要求が送信されないかタイムアウトした場合、フィールドは返されません。</ br>
- リストア要求が送信されてタイムアウトしない場合、返されるx-oss-restoreの値はongoing-request = “true”です。</ br>
- リストア要求が送信され、完了した場合、返されるx-oss-restoreの値は、ngoing-request = “false”、expiry-date = “Sun、16 Apr 2017 08:12:33 GMT”です。 >

例

リクエストの例:

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

リターンの例:

```
HTTP/1.1 200 OK
x-oss-request-id: 559CC9BDC755F95A64485981
x-oss-object-type: Normal
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
```

GetObjectMeta

GetObjectMeta は、バケット内のオブジェクトの基本メタ情報を取得するために使用されますが、バケットのコンテンツを返しません。 メタ情報には、Etag、Size (ファイルサイズ) 、および LastModified が含まれます。

リクエスト構文

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

詳細分析

- GetObjectMeta リクエストが送信された後、システムがOKメッセージを返してもエラーメッセージを返しても、メッセージ本文は返されません。
- GetObjectMeta は、ObjectMeta リクエストのパラメータを指定する必要があります。 指定しな

い場合は、GetObjectのリクエストとして処理されます。
- ファイルが存在しない場合、システムは 404 Not Foundエラーを返します。
- GetObjectMeta ではバケット内のオブジェクトの基本メタ情報のみが返されるため、Header Objectより軽量です。 メタ情報には、Etag、Size (ファイルサイズ) 、および LastModified が含まれます。 サイズは、Content-Length ヘッダーの値で測定されます。
- ファイルタイプが**シンボリックリンク**の場合、シンボリックリンク自体の情報のみが返されます。

例

リクエストの例

```
GET /oss.jpg?objectMeta HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0FmgbrQ0=
```

応答の例

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

PutObjectACL

PutObjectACL

PutObjectACL はオブジェクトのアクセス権限を変更するために使用します。現在オブジェクトで使用できるアクセス権限は、デフォルト、非公開、公開読み取り、公開読み書きの 4 つです。アクセス権限を設定するには、PutObjectACL リクエストで “x-oss-object-acl” ヘッダーを使用します。この操作を実行できるのはバケットオーナーだけです。操作が成功すると、200 が返されます。失敗すると、対応するエラーコードとプロンプトメッセージが返されます。

リクエスト構文

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

オブジェクト ACL の定義

名前	説明
非公開	このACLは、オブジェクトが非公開リソースであることを示します。このオブジェクトの読み取りまたは書き込みを実行できるのは、このオブジェクトのオーナーだけです。
公開読み取り	このACLは、オブジェクトがパブリックに読み取り可能なリソースであることを示します。このオブジェクトの読み取りおよび書き込みを実行できるのは、このオブジェクトのオーナーだけです。その他のユーザーはこのオブジェクトの読み取りのみを実行できます。
公開読み書き	このACLは、オブジェクトがパブリックに読み書き可能なリソースであることを示します。すべてのユーザーがこのオブジェクトの読み取りおよび書き込みを実行できます。
デフォルト	このACLは、オブジェクトがバケットの読み取り/書き込み許可を継承するリソースであることを示します。 つまり、バケットとオブジェクトは同じ権限を持ちます。

詳細分析:

- オブジェクトの読み取り操作には、GetObject、HeadObject、CopyObject、および UploadPartCopy 操作におけるソースオブジェクトの読み取りが含まれます。オブジェクトの書き込み操作には、PutObject、PostObject、AppendObject、DeleteObject、DeleteMultipleObjects、CompleteMultipartUpload、および CopyObject 操作における新しいオブジェクトの書き込みが含まれます。
- x-oss-object-acl は前述の 4 つの権限のいずれかに設定する必要があります。それ以外の場合、OSS は 400 Bad Request メッセージを返し、エラーコードはInvalidArgumentです。
- PutObjectACL を使用すると、オブジェクトの ACL を設定できます。また、オブジェクトを書き込む際に、リクエストヘッダーに x-oss-object-acl を含めてオブジェクトの ACL を設定することもできます。効果は PutObjectACL と同じです。たとえば、PutObject リクエストのヘッダーに x-oss-object-acl が含まれている場合は、オブジェクトのアップロード時にオブジェクトの ACL を設定できます。
- オブジェクトの読み取り権限がないユーザーがこのオブジェクトを読み取ろうとすると、OSS は 403 Forbiddenメッセージを返し、エラーコードは AccessDenied になります。表示されるプロンプトは “You do not have read permission on this object” です。

- オブジェクトの書き込み権限がないユーザーがこのオブジェクトを書き込もうとすると、OSS は 403 Forbidden メッセージを返し、エラーコードは AccessDenied になります。表示されるプロンプトは “You do not have write permission on this object” です。
- PutObjectACL を使用してバケット内のオブジェクトの ACL を変更できるのは、このバケットのオーナーだけです。バケットオーナー以外のユーザーが PutObjectACL を使おうとすると、OSS は 403 Forbidden メッセージを返し、エラーコードは AccessDenied になります。表示されるプロンプトは “You do not have write acl permission on this object” です。
- オブジェクト ACL はバケット ACL よりも優先されます。たとえば、バケット ACL が非公開でオブジェクト ACL が公開読み書きの場合、ユーザーがオブジェクトにアクセスする際には、まずこのオブジェクトの ACL がチェックされます。結果として、このバケットが非公開バケットであっても、このオブジェクトにはすべてのユーザーがアクセスできます。オブジェクトの ACL が設定されていない場合、このオブジェクトの ACL はオブジェクトが配置されているバケットの ACL と同じになります。

例

リクエストの例:

```
PUT /test-object?acl HTTP/1.1
x-oss-object-acl: public-read
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=
```

応答例:

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

GetObjectACL

GetObjectACL はバケット内のオブジェクトへのアクセス権限を取得するために使用します。

リクエスト構文

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

応答の要素

名前	説明
AccessControlList	ACL 情報の格納に使用されるコンテナー。 型: コンテナー 親ノード: AccessControlPolicy
AccessControlPolicy	GetObjectACL の結果を保存するコンテナーを指定します。 型: コンテナー 親ノード: なし
DisplayName	バケットオーナーの名前 (現在、この名前はバケットオーナー ID と同じです)。 型: string 親ノード: AccessControlPolicy.Owner
Grant	オブジェクトの ACL 権限を指定します。 型: 列挙文字列 有効な値: private、public-read、public-read-write 親ノード: AccessControlPolicy.AccessControlList
ID	バケットオーナーのユーザー ID。 型: string 親ノード: AccessControlPolicy.Owner
Owner	バケットオーナーに関する情報の保存に使用されるコンテナー。 型: コンテナー 親ノード: AccessControlPolicy

詳細分析

- GetObjectACL を使用してバケット内のオブジェクトの ACL を取得できるのは、バケットオーナーだけです。バケットオーナー以外が GetObjectACL リクエストを送信した場合は、403 Forbidden メッセージが返されます。エラーコードは AccessDenied です。表示されるプロンプトは “You do not have read acl permission on this object” です。
- GetObjectACL リクエストを送信したがオブジェクトの ACL が設定されていない場合は、OSS から返される ObjectACL がデフォルトになります。これは、このオブジェクトの ACL がバケット ACL と同じであることを示しています。つまり、バケットのアクセス権限が非公開の場合はこのオブジェクトのアクセス権限も非公開になり、バケットのアクセス権限が公開読み書きの場合はこのオブジェクトのアクセス権限も公開読み書きになります。

例

リクエストの例:

```
GET /test-object?acl HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:CTkuxpLAi4XZ+WwIfNm0FmgbrQ0=
```

戻り値の例:

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

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

PostObject

Post Object は、HTML フォームを使用してファイルを指定したバケットにアップロードするために使用します。Post Object は Put Object の代替手段で、ブラウザーに基づいてファイルをバケットにアップロードできます。Post Object のメッセージ本文は、multipart/form-data を使用してエンコードされます。Put Object 操作では、パラメーターは HTTP リクエストヘッダーで転送されます。Post Object 操作では、パラメーターはメッセージ本文のフォームフィールドとして転送されます。

Post object

リクエスト構文

```
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--
```

フォームフィールド

名前	説明	必須かどうか
OSSAccessKeyId	バケットオーナーの Access Key ID を指定します。 型: 文字列	条件による

	デフォルト値: なし 制約: このフォームフィールドは、バケットで公開読み書きが許可されていない場合、または Policy (または Signature) フォームフィールドが指定されている場合は必須です。	
policy	リクエストのフォームフィールドの有効性を指定します。Policy フォームフィールドが含まれていないリクエストは匿名リクエストとして扱われ、公開読み書きが許可されているバケットにのみアクセスできます。詳細については、5.7.4.1「Post Policy」を参照してください。 型: 文字列 デフォルト値: なし 制約: このフォームフィールドは、バケットで公開読み書きが許可されていない場合、または OSSAccessKeyId (または Signature) フォームフィールドが指定されている場合は必須です。	条件による
Signature	Access Key Secret と Policy に基づいて計算される署名情報を指定します。OSS では、この署名情報をチェックして Post Object リクエストの有効性を検証します。詳細については、5.7.4.2「Post Signature」を参照してください。 型: 文字列 デフォルト値: なし 制約: このフォームフィールドは、バケットで公開読み書きが許可されていない場合、または OSSAccessKeyId (または Policy) フォームフィールドが指定されている場合は必須です。	条件による
Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires	REST リクエストヘッダー。詳細については、Put Object の関連する説明を参照してください。 型: 文字列 デフォルト値: なし	オプション
file	ファイルまたはテキストコンテンツを指定します。このフィールドはフォームの最後に指定する必要があります。ブラウザーで Content-Type がファイルタイプに基づいて自動的に設定	必須

	され、ユーザー設定が上書きされます。 OSS では一度に 1 つのファイルのみをアップロードできます。 型: 文字列 デフォルト値: なし	
key	アップロードするファイルのオブジェクト名を指定します。アップロードするファイルの名前をオブジェクト名として使用する必要がある場合は、${filename} 変数を使用します。たとえば、b.jpg ファイルをアップロードする場合に Key フィールドが /user/a/${filename} に設定されていると、最終的なオブジェクト名は /user/a/b.jpg になります。 ファイル名にパスが含まれている場合は、ファイル名からパスを削除します。たとえば、a/b/c/b.jpg ファイルをアップロードする場合は、ファイル名 b.jpg を使用します。Key フィールドが /user/a/${filename} に設定されていると、最終的なオブジェクト名は /user/a/b.jpg になります。 型: 文字列 デフォルト値: なし	必須
success_action_redirect	アップロードが正常に実行された後にクライアントがリダイレクトされる URL を指定します。このフォームフィールドを指定しない場合は、返される結果を success_action_status で指定します。 アップロードに失敗した場合はエラーコードが返され、クライアントはどの URL にもリダイレクトされません。 型: 文字列 デフォルト値: なし	オプション
success_action_status	success_action_redirect が指定されていない場合において、前述のようにアップロードが正常に実行された後にクライアントに返されるステータスコードを指定します。 有効な値には 200、201、および 204 (デフォルト) があります。 このフィールドを 200 または 204 に設定した場合は、空のファイルと対応するステータスコードが返されます。 このフィールドを	オプション

	201 に設定した場合は、XML ファイルと 201 ステータスコードが返されます。 このフィールドを指定しない場合、または無効な値に設定した場合は、空のファイルと 204 ステータスコードが返されます。 型: 文字列 デフォルト値: なし	
x-oss-meta-*	ユーザーが設定したユーザーメタ値を指定します。 OSS ではこの値はチェックされず、使用もされません。 型: 文字列 デフォルト値: なし	オプション
x-oss-server-side-encryption	OSS によってオブジェクトが作成される際のサーバー側の暗号化アルゴリズムを指定します。 型: 文字列 有効な値: AES256	オプション
x-oss-object-acl	OSS によってオブジェクトが作成される際のアクセス権限を指定します。 型: 文字列 有効な値: public-read、private、public-read-write	オプション
x-oss-security-token	このアクセスにSTSの一時的な許可が使用される場合は、該当項目にSecurityToken値に指定する必要があります。 同時に、OSSAccessKeyIdは、ペアになった一時的なAccessKeyIdを使用する必要があります。 署名計算は、一般的なAccessKeyId署名と一貫しています。 型: 文字列 デフォルト値: なし	オプション

応答ヘッダー

名前	説明
x-oss-server-side-encryption	リクエストで x-oss-server-side-encryption を指定した場合は、使用された暗号化アルゴリズムを示すこのヘッダーが応答に含まれます。 型: 文字列

応答の要素

名前	説明

PostResponse	Post Object リクエストの結果を保存するコンテナーを指定します。 型: コンテナー サブノード: Bucket、ETag、Key、Location
Bucket	バケット名を指定します。 型: 文字列 親ノード: PostResponse
ETag	オブジェクトの生成時に作成されるエンティティタグ (ETag) を指定します。Post Object で作成されたオブジェクトの場合、ETag 値はオブジェクトの MD5 値で、オブジェクトのコンテンツが変更されているかどうかを確認するために使用できます。 型: 文字列 親ノード: PostResponse
Location	新たに作成されたオブジェクトの URL を指定します。 型: 文字列 親ノード: PostResponse

詳細分析

- Post Object 操作を実行するには、バケットの書き込み権限が必要です。バケットで公開読み書きが許可されている場合は、署名情報をアップロードしないように選択できます。それ以外の場合は、Post Object 操作で署名の検証を実行する必要があります。Put Object とは異なり、Post Object では Access Key Secret を使用してポリシーの署名を計算します。計算された署名文字列を Signature フォームフィールドの値として使用します。OSS では、この値をチェックして署名の有効性を検証します。
- バケットで公開読み書きが許可されているかどうかに関係なく、OSSAccessKeyId、Policy、Signature のフォームフィールドのいずれかをアップロードすると、残りの 2 つのフォームフィールドは必須になります。残りの 2 つのフォームフィールドを指定しない場合は、エラーコード InvalidArgument が返されます。
- Post Object 操作で送信するフォームエンコーディングは、” multipart/form-data” である必要があります。つまり、ヘッダーの Content-Type は multipart/form-data; boundary=xxxxxx という形式にする必要があります。ここで、boundary は境界文字列です。
- 送信フォームの URL はバケットのドメイン名にすることができます。URL でオブジェクトを指定する必要はありません。つまり、リクエスト行は POST / HTTP/1.1 となり、POST /ObjectName HTTP/1.1 と記述することはできません。
- ポリシーでは、Post Object リクエストのフォームフィールドの有効な値を指定します。OSS では、このポリシーに基づいてリクエストの有効性をチェックします。リクエストが無効である場合は、エラーコード AccessDenied が返されます。ポリシーの有効性をチェックする際、OSS ではポリシー内の無関係なフォームフィールドはチェックされません。
- フォームとポリシーは UTF-8 でエンコードする必要があります。ポリシーは、UTF-8 と Base64 でエンコードされた JSON テキストです。
- Post Object リクエストには、追加のフォームフィールドを含めることができます。OSS では、ポ

リシーに基づいてこのようなフォームフィールドの有効性をチェックします。
- Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。
- Post Object リクエストにヘッダー署名または URL 署名が含まれている場合、OSS ではこれらの署名はチェックされません。
- Put Object リクエストに x-oss-meta- というプレフィックスが付いたフォームフィールドが含まれている場合、このフォームフィールドはユーザーメタ (x-oss-meta-location など) として扱われます。1 つのオブジェクトに対して同様のパラメーターを複数設定することはできますが、すべてのユーザーメタの合計サイズは 8 KB 以下にする必要があります。
- Post Object リクエストの本文全体の長さは 5 GB までです。ファイルサイズが大きすぎる場合は、エラーコード EntityTooLarge が返されます。
- x-oss-server-side-encryption ヘッダーを指定したリクエストをアップロードする場合、このヘッダーの値は AES256 に設定する必要があります。それ以外に設定した場合は、400 メッセージとエラーコード InvalidEncryptionAlgorithmError が返されます。このヘッダーを指定すると、応答ヘッダーにもこのヘッダーが含まれ、アップロードされたオブジェクトの暗号化アルゴリズムが示されます。このオブジェクトのダウンロード時にも、応答ヘッダーには、値がこのオブジェクトの暗号化アルゴリズムに設定されている x-oss-server-side-encryption が含まれます。
- フォームフィールドでは大文字小文字は区別されませんが、その値では区別されます。

例

リクエストの例:

```
POST / HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 344606
Content-Type: multipart/form-data; boundary=9431149156168

--9431149156168
Content-Disposition: form-data; name="key"

/user/a/${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"

eyJleHBpcmF0aW9uIjoiMjAxMy0xMi0wMVQxMjowMDowMFoiLCJjb25kaXRpb25zIjpbWyJjb250ZW50LWxlbmd0aC1
yYW5nZSIsIDAsIDEwNDg1NzYwXSx7ImJ1Y2tldCI6ImFoYWhhIn0sIHsiQSI6ICJhIn0seyJrZXkiOiAiQUJDIn1dfQ==
--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--
```

応答例:

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

Post Policy

Post Object リクエストの policy フォームフィールドは、リクエストの有効性を検証するために使用します。ポリシーは、UTF-8 と Base64 でエンコードされた JSON テキストです。Post Object リクエストが満たす必要がある条件を示します。post フォームフィールドは、公開読み書きが許可されているバケットをアップロードする場合はオプションですが、このフォームフィールドを使用して Post Object リクエストを制限することを強くお勧めします。

ポリシーの例

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

```
}
```

Post Object リクエストでは、ポリシーに expiration と conditions を含める必要があります。

expiration

expiration では、ISO8601 GMT で表されるポリシーの有効期限を指定します。たとえば、”2014-12-01T12:00:00.000Z” は、Post Object リクエストを 2014 年 12 月 1 日 12:00 より前に送信する必要があることを意味します。

conditions

conditions は、Post Object リクエストのフォームフィールドの有効な値を指定する条件リストです。注意: フォームフィールドの値は、OSS でポリシーがチェックされた後に展開されます。したがって、ポリシーで設定されているフォームフィールドの有効な値は、展開前のフォームフィールドの値と同じです。たとえば、key フォームフィールドが user/user1/${filename} に設定されており、ユーザーのファイル名が a.txt である場合は、Post Object リクエストの policy フォームフィールドを [“eq”,”$key”,”$key”,”user/user1/a.txt] ではなく [“eq”,”$key”,”user/user1/\${filename}”] に設定する必要があります。ポリシーでサポートされている条件を次の表に示します。

名前	説明
content-length-range	アップロードするファイルの最大許容サイズと最小許容サイズを指定します。 この条件では content-length-range との一致モードがサポートされます。
Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires	HTTP リクエストヘッダー。 この条件では完全一致モードと前方一致モードがサポートされます。
key	アップロードするファイルのオブジェクト名を指定します。 この条件では完全一致モードと前方一致モードがサポートされます。
success_action_redirect	アップロードが正常に実行された後にクライアントがリダイレクトされる URL を指定します。 この条件では完全一致モードと前方一致モードがサポートされます。
success_action_status	success_action_redirect が指定されていない場合において、アップロードが正常に実行された後に返されるステータスコードを指定します。 この条件では完全一致モードと前方一致モードがサポートされます。
x-oss-meta-*	ユーザーが設定したユーザーメタを指定します。この条件では完全一致モードと前方一致モードがサポートされます。

Post Object リクエストにその他のフォームフィールドが含まれている場合は、その追加のフォームフィールドをポリシーの条件に追加できます。OSS では、条件に含まれていないフォームフィールドの有効性はチ

ェックされません。

条件の一致モード

条件の一致モード	説明
完全一致	フォームフィールドの値が条件で宣言されている値と完全に同じである必要があります。たとえば、key フォームフィールドの値が a でなければならない場合は、条件を { “key”： “a” } または [“eq”， “$key”， “a”] にする必要があります。
前方一致	フォームフィールドの値の先頭が指定した値である必要があります。たとえば、key の値の先頭が /user/user1 でなければならない場合は、条件を [“starts-with”， “$key”， “/user/user1”] にする必要があります。
指定したファイルサイズ	アップロードできるファイルの最大サイズと最小サイズを指定します。たとえば、ファイルの許容サイズが 1 ～ 10 バイトの場合は、条件を [“content-length-range”, 1, 10] にする必要があります。

エスケープ文字

Post Object リクエストの policy フォームフィールドでは、変数を示すために $ を使用します。したがって、$ を記述するにはエスケープ文字 \$ を使用する必要があります。また、JSON 文字列の一部の文字はエスケープされます。Post Object リクエストの policy フォームフィールドの JSON 文字列内の文字を次の表に示します。

エスケープ文字	説明
\/	スラッシュ
\	バックスラッシュ
\”	二重引用符
\$	ドル記号
\b	スペース
\f	改ページ
\n	改行
\r	キャリッジリターン
\t	水平タブ
\uxxxx	Unicode 文字

Post Signature

Post Object リクエストの検証のためには、HTML フォームに policy と signature を含める必要があります。policy では、リクエストで許容される値を指定します。signature を計算するための手順は次のとおりです。

1. UTF-8 でエンコードされたポリシーを作成します。
2. このポリシーを Base64 でエンコードします。エンコーディングの結果が policy フォームフィールドの値になり、この値が署名する文字列として使用されます。
3. AccessKeySecretを使用してこの文字列に署名します。署名方法は、ヘッダーの署名の計算方法と同じです。つまり、署名する文字列を policy フォームフィールドに置き換えます。詳細は ヘッダーへの署名の追加をご覧ください。

サンプル Demo

- web フォームフィールドから OSS にパラメータを渡すデモ： ここをクリックしてください。

コールバック

コールバックを実行するには、関連するコールバックパラメータをOSSに送信された要求に添付するだけです。現在コールバックをサポートするAPIには、PutObject、PostObject、CompleteMultipartUploadなどがあります。

コールバックパラメータを構築する

コールバックパラメータは、Base64でエンコードされたJSON文字列で構成されています。要求コールバックサーバーURL（callbackUrl）とコールバックコンテンツ（callbackBody）を指定することが重要です。詳細なJSONフィールドは次のとおりです。

フィールド	意味	
callbackUrl	- ファイルが正常にアップロードされると、OSSはこのURLにコールバック要求を送信します。リクエストメソッドはPOSTで、bodyはcallbackBodyに指定されたコンテンツです。通常の状況で、このURLが「HTTP / 1.1 200 OK」に応答する必要がある場合、レスポンス本文はJSON形式でなければならず、レスポンスヘッダーContent-Lengthは	必須

	有効な値で、3 MBを超えない必要があります。 - この機能を使用すると、ユーザーは「;」で区切られた5つのURLを設定できます。最初の正常な応答が返されるまで、OSSは要求を1つずつ送信します。 - URLが設定されていないか、値がnullの場合、コールバック設定がないとみなされます。 - HTTPSがサポートされています。 - 中国語文字が正しく処理されるようにするには、callbackUrlをエンコードする必要があります。たとえば、http://example.com/Chinese.php?key=value&Chinese Name=Chinese Valueを 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のように変換します。	
callbackHost	- コールバック要求を開始するためのホストヘッダー値。callbackUrlが設定されている場合にのみ有効です。 - callbackHostが設定されていない場合、callbackUrlのURLが解決され、解決後に生成されたホストはcallbackHostに入力されます。	オプション
callbackBody	- コールバックが開始されたときのリクエスト本文の値。たとえば、key = $（key）&etag = $（etag）&my_var = $（x：my_var）。 - OSSシステム変数、カスタム変数、および定数をサポートします。サポートされているシステム変数については、以下の表で説明します。カスタム変数は、PutObjectとCompleteMultipartのコールバック-varによる送信によってサポートされています。Post Object操作では、各変数はフォームフィールドを介して送信されます。	必須
callbackBodyType	- 開始されたコールバック要求のContent-Type。これは、application/x-www-form-	オプション

	urlencodedとapplication/jsonをサポートし、前者はデフォルト値です。 - Content-Typeがapplication/x-www-form-urlencodedに設定されている場合、callbackBodyの変数はURLエンコードされた値に置き換えられます。Content-Typeがapplication/jsonに設定されている場合、これらの変数はJSON形式に従って置き換えられます。	

JSONの文字列の例は次のとおりです。

```
{
"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}"
}
```

ここで、callbackBodyに設定できるシステム変数には、次のものがあります。具体的には、imageInfoは画像フォーマット用です。非画像形式の場合は空白のままにしてください。

システム変数	**意味**
bucket	バケット
object	オブジェクト
etag	ファイルのetag、ユーザーに返されたetagフィールド
size	オブジェクトのサイズ CompleteMultipartUpload操作中、これはオブジェクト全体のサイズです。
mimeType	リソースタイプ。jpegイメージの場合、リソースタイプはimage/jpegです。
imageInfo.height	画像の高さ
imageInfo.width	画像の幅
imageInfo.format	jpgやpngなどの画像フォーマット

カスタムパラメータ

callback-varパラメータを使用してカスタムパラメータを構成できます。

カスタムパラメータは、キー値のマップです。マップに必要なパラメータを設定できます。POSTコールバック要求を開始するとき、OSSはこれらのパラメータと上記のセクションで説明したシステムパラメータをPOST要求の本体に入れます。これにより、コールバック受信者がこれらのパラメータを簡単に取得できるようになります。

コールバックパラメータの作成と同じ方法でカスタムパラメータを構築できます。カスタムパラメータは、JSON形式で送信することもできます。JSON文字列は、すべてのカスタムパラメータのキー値を含むマップです。**カスタムパラメータのキーは、x：で始まり、小文字にする必要があります。** 2つのカスタムパラメータx：var1とx：var2を設定する必要があり、2つのパラメータの値がそれぞれvalue1とvalue2であると仮定すると、構築されるJSON形式は次のようになります。

```
{
"x:var1":"value1",
"x:var2":"Value2"
}
```

コールバックリクエストを作成する

callbackおよびcallback-varパラメータが構築された後、3つの方法でパラメータをOSSに送信できます。コールバックパラメータは必須で、callback-varパラメータはオプションです。カスタムパラメータを設定しない場合は、callback-varフィールドを追加する必要はありません。前記3つの方法は以下の通りである。

- URLにパラメータを含める。
- ヘッダにパラメータを含める。
- フォームフィールドを使用して、POST要求の本体にパラメーターを組み込む。 **POSTを使用してオブジェクトをアップロードする場合にのみ、このメソッドを使用してコールバックパラメータを指定できます。**

3つの方法は別の方法です。それ以外の場合、OSSはInvalidArgumentエラーを返します。

OSS要求にパラメータを含めるには、最初にBase64を使用して上で構築したJSON文字列をエンコードし、以下に説明するメソッドを使用して文字列をOSS要求に含める必要があります。

- URLにパラメータを含めるには、URLパラメータとして ‘callback = [CallBack]’ または ‘callback-var = [CallBackVar]’ を使用してリクエストと共に送信します。署名のCanonicalizedResourceが計算されると、callbackまたはcallback-varがサブリソースとして考慮されます。
- ヘッダにパラメータを含めるには、 ‘x-oss-callback = [CallBack]’ または ‘x-oss-callback-var = [CallBackVar]’ を頭に使用してリクエストとともに送信します。CanonicalizedOSSHeaders of the signatureが計算されると、x-oss-callback-varとx-oss-callbackが考慮されます。以下に例を示します。

```
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:
eyJjYWxsYmFja1VybCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZHkiOiJidWNrZXQ9JHtidWNrZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pbWVUeXBlfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlaWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLndpZHofSZpbWFnZUluZm8uZm9ybWF0PSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV92YXJ9In0=
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
```

- 独立したフォームフィールドを使用してコールバックパラメータを組み込む必要があるため、POSTを使用してオブジェクトをアップロードするときに、コールバックパラメータを含めるのが少し複雑です。以下の例を参照してください。

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

eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94d3dy1mb3JtLXVybGVuY29kZWQifQ==
```

カスタムパラメータがある場合、**フォームフィールド**にcallback-varパラメータを直接含めることはできません。独立したフォームフィールドを使用して各カスタムパラメータを組み込む必要があります。たとえば、カスタムパラメータのJSONが：

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

POSTリクエストのフォームフィールドは、次のようにする必要があります。

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

eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94d3dy1mb3JtLXVybGVuY29kZWQifQ==

--9431149156168
```

```
Content-Disposition: form-data; name="x:var1"

value1

--9431149156168
Content-Disposition: form-data; name="x:var2"

value2
```

同時に、ポリシーにコールバック条件を追加できます（コールバックが追加されていない場合は、このパラメータでアップロード検証は実行されません）。 例えば：

```
{ "expiration": "2014-12-01T12:00:00.000Z",
"conditions": [
{"bucket": "johnsmith" },
{"callback":
"eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJjYWxsYmFja0JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkIn0="},
["starts-with", "$key", "user/eric/"],
]
}
```

コールバック要求を開始する

ファイルが正常にアップロードされた場合、OSSはPOSTメソッドを使用して、コールバックパラメータとユーザーの要求のカスタムパラメータ（callback-varパラメータ）に基づいてアプリケーションサーバーに特定のコンテンツを送信します。

```
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=D8E8FCA2DC0F896FD7CB4CB0031BA249&size=5&mimeType=text%2Fplain&imageInfo.height=&imageInfo.width=&imageInfo.format=&x:var1=for-callback-test
```

コールバックの結果を返す

たとえば、アプリケーションサーバーは次の応答要求を返します。

```
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"}
```

アップロード結果を返す

次のコンテンツがクライアントに送信されます。

```
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"}
```

CompleteMultipartUploadなどのリクエストの場合、返されたリクエスト本文にはコンテンツが存在することに注意してください (たとえば、XM1形式の情報) 。アップロードコールバック関数を使用すると、元の本文の内容が上書きされます (例： “a” : “b” ‘) 。これを判断と処理に配慮してください。

コールバック署名

コールバックパラメータが設定されると、OSSは、ユーザーが設定したcallbackUrlに基づいて、POSTコールバック要求をユーザーのアプリケーションサーバーに送信します。コールバック要求を受信した後、アプリケーションサーバがコールバック要求がOSSによって開始されているかどうかを確認する場合は、コールバック要求にシグネチャを含めてOSS IDを確認できます。

署名を生成する

署名はOSS側で発生し、RSA Asymmetric Encryptionを使用して署名されます。次のように秘密鍵を使用して署名を暗号化できます。

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

説明：private_keyは、OSSにのみ知られている秘密鍵を示します。パスは、コールバック要求のリソースパスを示します。query_stringはクエリ文字列を示します。本文は、コールバックのメッセージ本文を示します。したがって、署名は以下のステップになります。

- 署名する文字列を取得する：リソースパスのURLがデコードされ、初期クエリ文字列、キャリッジリターン、およびコールバックメッセージ本文によって追加されます。

- RSA署名：秘密鍵を使用して目的の文字列に署名します。署名のハッシュ関数はMD5です。
- Base64を使用して、署名付きの結果をエンコードして最終的な署名を取得します。署名をコールバック要求の承認ヘッダーに入れます。

以下に例を示します。

```
POST /index.php?id=1&index=2 HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 18
authorization:
kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==

bucket=yonghu-test
```

パスは / index.php、query_stringは ? id = 1＆index = 2であり、本文は bucket = yonghu-testであり、最終的な署名結果はkKQeGTRccDKyHB3H9vF + xYMSrmhMZjzzl2 / kdD1ktNVgbWEfYTQG0G2SU / RaHBovRCE8OkQDjC3uG33esH2txA ==です。

署名を検証する

署名検証は、署名の逆のプロセスです。署名はアプリケーションサーバーによって検証され、プロセスは次のようになります。

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

フィールドは、署名処理中に記述されたのと同じ意味を持ちます。public_keyは、公開鍵を示します。認可は、コールバックヘッダーの署名を示します。署名の検証は、次の手順で構成されます。

- コールバック要求のx-oss-pub-key-urlヘッダーには、Base64でエンコードされた公開鍵のURLが格納されます。次のように、ヘッダーをBase64でデコードして公開鍵を取得する必要があります。

```
public_key = urlopen(base64_decode(Value of the x-oss-pub-key-url header))
```

x-oss-pub-key-urlヘッダの値はhttp：//gosspublic.alicdn.com/やhttps：//gosspublic.alicdn.com/で始まらなければならないことに注意してください公開鍵がOSSによって確実に提供されるようにします。

- Base64でデコードされた署名を取得します。

```
signature = base64_decode(Value of the authorization header)
```

- 署名プロセスで説明したのと同じ方法で、署名する文字列を取得します。

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

- 署名を確認します

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

上記のサンプルを例として使用します。

- 公開鍵のURLを取得します。aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==をBase64http：// gosspublic.alicdn.com / callback_pub_key_v1.pem`で複合化します。
- 署名ヘッダ kKQeGTRccDKyHB3H9vF + xYMSrmhMZjzzl2 / kdD1ktNVgbWEfYTQG0G2SU / RaHBovRCE8OkQDjC3uG33esH2txA ==は、Base64でデコードされます（デコードされた結果は、表示できない文字列であるため表示できません）。
- 署名文字列、url_decode（ “index.php” ）+ “？id = 1＆index = 2” + “bucket = yonghu-test” を取得します。次に、MD5チェックを実行します。
- 署名を確認します。

アプリケーションサーバーの例

Pythonは、アプリケーションサーバーが署名を検証する方法を示すための例として使用されています。この例では、M2Cryptoライブラリをインストールする必要があります。

```
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()
```

他の言語で実装されているアプリケーションサーバーは、次のとおりです。

Javaバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：パッケージを展開し、 java -jar oss-callback-server-demo.jar 9000を実行します（9000はポート番号で、必要に応じて指定できます）

PHPのバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：プログラムをApache環境にデプロイします。PHP言語の特徴は、環境がいくつかのヘッダを取得することに依存していることを決定します。この例を参照して、自分の環境を変更することができます。

Pythonバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：パッケージを展開し、 python callback_app_server.pyを直接実行します。このプログラムを実行するには、RSA依存関係をインストールする必要があります。

C#のバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：パッケージを解凍し、 README.mdを参照してください。

Goバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：パッケージを解凍し、 README.mdを参照してください。

Rubyバージョン：

- ダウンロードアドレス：ここをクリック
- 実行方法：ruby aliyun_oss_callback_server.rb

特別な指示

- 入力コールバックパラメータまたはcallback-varパラメータが無効な場合は、エラーコード “InvalidArgument” とともに400エラーが返されます。無効な状況は次のとおりです。
- PutObject（）およびCompleteMultipartUpload（）インタフェースでは、callback（x-oss-callback）またはcallback-var（x-oss-callback-var）パラメータがURLおよびヘッダーフィールドに同時に入力されます。
- callbackまたはcallback-varパラメータが長すぎます（5KB以上）。PostObject（）はcallback-varパラメータがないのでこの制限の対象にはならず、以下の場合も同様です。
- callbackまたはcallback-varがBase64でエンコードされていない
- Base64のデコード後、callbackまたはcallback-varパラメータが有効なJSON形式ではありません
- callbackパラメータの解決後、callbackUrlフィールドに5つ以上のURLが含まれているか、またはURLの入力ポートが無効です（たとえば{" callbackUrl "：" 10.101.166.30:test "、" callbackBody "：" test "})
- callbackパラメータの解決後、callbackBodyフィールドは空白です。
- callbackパラメータの解決後、callbackBodyTypeフィールドの値は、 “application/x-www-form-urlencoded” または “application/json” です。
- callback-varパラメータの解決後、callbackBodyフィールドには無効な形式の変数が含まれています。有効な形式は $ {var}です。
- callback-varパラメータ解決後、フォーマットは期待されるJSONフォーマットではありません。予想される形式は次のとおりです。{" x：var1 "：" value1 "、" x：var2 "：" value2 "...}
- callback失敗すると、システムは203エラーを返し、エラーコード “CallbackFailed” を返します。コールバックの失敗は、アプリケーションサーバーがコールバック要求を受信していないことを意味するのではなく、OSSが予期されるコールバック応答を受信しなかったことを示します（たとえば、アプリケーションサーバーからの応答はJSON形式ではありません）。さらに、この時点までに、ファイルは正常にOSSにアップロードされています。
- アプリケーションサーバーからOSSに返される応答には、Content-Lengthヘッダーが含まれていなければならず、本文のサイズは1 MBを超えることはできません。

Put Symlink

Putシンボリックリンクは、シンボリックリンクを作成するために使用されます。

リクエスト構文

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

リクエストヘッダー

名前	説明
x-oss-symlink-target	シンボリックリンクが指示するターゲットファイルを示します。 タイプ：string 有効な値：命名規則はオブジェクトの命名規則と同じです。

詳細分析

1. ObjectNameの場合と同様に、TargetObjectNameはURLエンコードされます。
2. シンボリックリンクのターゲットファイルタイプは、シンボリックリンクにすることはできません。
3. シンボリックリンクを作成するときは、
   - ターゲットファイルが存在するかどうかをチェックしない。
   - 対象のファイルタイプが有効かどうかをチェックしない。
   - ターゲットファイルへのアクセスをチェックしないでください。前述のチェックは、GetObjectがターゲットファイルのAPIにアクセスする必要があるまで行われません。
4. 追加するファイルがすでに存在し、ファイルアクセス権を持っている場合、新しく追加されたファイルは既存のファイルを上書きし、システムは200 OKを返します。
5. PutSymlink要求に接頭引数x-oss-meta-が付いている場合、パラメータはユーザーメタとして扱われます（例：x-oss-meta-location）。1つのオブジェクトに複数の同様のパラメータを設定できますが、すべてのユーザーメタの合計サイズは8 KBを超えることはできません。
6. バケットタイプがアーカイブの場合、このインターフェイスを呼び出すことはできません。それ以外の場合は、エラーコード “OperationNotSupported” とともにエラー400が返されます。

例

リクエスト例：

```
PUT /link-to-oss?symlink HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Cache-control: no-cache
```

```
Content-Disposition: attachment;filename=oss_download.jpg
Date: Tue, 08 Nov 2016 02:00:25 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:kZoYNv66bsmc10+dcGKw5x2PRrk=
x-oss-symlink-target: oss.jpg
```

リターン例：

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

Get Symlink

シンボリックリンクの取得

シンボリックリンクの取得操作は、読み取り権限が必要なシンボリックリンクを取得するために使用されます。

要求の構文

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

レスポンスヘッダ

名前	説明
x-oss-symlink-target	シンボリックリンクが指し示すターゲットファイル。 タイプ：文字列

詳細分析

1. シンボリックリンクが存在しない場合、システムは404 Not Foundエラーを返します。エラーコード：NoSuchKey。

例

リクエスト例：

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

リターン例：

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

オブジェクトの復元

オブジェクトの復元

アーカイブタイプのオブジェクトを読み取るには、復元操作を実行して、サーバーにオブジェクトを復元させます。オブジェクトのタイプがStandardまたはIAの場合は、Restoreインタフェースを呼び出さないでください。

リストア操作前後のアーカイブされたオブジェクトの状態変更プロセスを以下に示します。

1. アーカイブタイプのオブジェクトは、最初にアーカイブされます。
2. 復元要求が開始されると、サーバーはオブジェクトの復元を開始します。
3. サーバーの復元が完了すると、オブジェクトが復元され、オブジェクトを読み取ることができます。
4. 修復の状態は、デフォルトでは1日間続き、最大7日まで延長できます。有効期限が切れた後、オブジェクトは再びアーカイブされます。

アーカイブされたオブジェクトに対して復元操作を実行すると、データ取得コストが発生します。復元または既に復元されているアーカイブオブジェクトに対して別の復元要求を開始しても、それ以上のデータ取得コストは発生しません。

リクエスト構文

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

詳細分析

1. 最初にオブジェクトのRestoreインタフェースが呼び出された場合、システムは202を返します。
2. リストアインターフェースが呼び出され、リストアがすでに進行中で、インターフェースが2度目に呼び出された場合、システムはエラーコード “RestoreAlreadyInProgress” を返して409を返します。これは、サーバーがリストア操作を実行しており、完了まで最大で4時間待たなければなりません。
3. Restoreインターフェースが呼び出され、リストアが完了した後、インターフェースが再度呼び出されると、システムは200を返し、ダウンロード可能時間は1日（最大7日）延長されます。
4. オブジェクトが存在しない場合、システムは404を返します。
5. アーカイブタイプではないオブジェクトに対して復元要求が開始された場合、エラーコード “OperationNotSupported” とともにエラー400が返されます。

例

最初に開始されたリストアリクエストの例

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

戻り値の例

```
HTTP/1.1 202 Accepted
Date: Sat, 15 Apr 2017 07:45:28 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74
```

リストアが完了していないときに再度呼び出されたリクエストの例

```
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=
```

戻り値の例

```
HTTP/1.1 409 Conflict
Date: Sat, 15 Apr 2017 07:45:29 GMT
Content-Length: 556
Connection: keep-alive
Server: AliyunOSS
x-oss-request-id: 5374A2880232A65C23002D74

<?xml version="1.0" encoding="UTF-8"?>
<Error>
<Code>RestoreAlreadyInProgress</Code>
<Message>The restore operation is in progress.</Message>
<RequestId>58EAF141461FB42C2B000008</RequestId>
<HostId>10.101.200.203</HostId>
</Error>
```

リストアが完了したときに再度呼び出されたリクエストの例

```
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=
```

戻り値の例

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

マルチパートアップロード操作

マルチパートアップロードの概要

マルチパートアップロードの概要

OSS でファイルをアップロードするには、PUT Object インターフェイスのほかに、マルチパートアップロードモードを使用することもできます。マルチパートアップロードモードを適用できるシナリオの例を以下に示します (これがすべてではありません)。

- ブレークポイントアップロードをサポートする必要がある場合
- アップロードするファイルのサイズが 100 MB を超えている場合
- ネットワークの条件が悪く、OSS サーバーとの接続が頻繁に切断される場合
- ファイルをアップロードする前に、ファイルのサイズを特定できない場合

InitiateMultipartUpload

Initiate Multipart Upload

データをマルチパートアップロードモードで転送する前に、Initiate Multipart Upload インターフェイスを呼び出して、マルチパートアップロードイベントを開始するように OSS に通知する必要があります。Initiate Multipart Upload インターフェイスは、このマルチパートアップロードイベントを特定するために OSS サーバーによって作成されたグローバルに一意なアップロード ID を返します。この ID に基づいて、マルチパートアップロードの中止やマルチパートアップロードの照会などの操作を開始できます。

リクエスト構文

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

リクエスト パラメーター

マルチパートアップロードモードを初期化するとき、encoding-typeパラメーターを利用して返されるエンコードキーを指定可能です。

名前	説明
encoding-type	返されるエンコードキーを指定します。 現在

	URLエンコードのみサポートします。キーはUTF-8のエンコードを採用します。 型: 文字列 デフォルト値: なし オプション値: url

リクエストヘッダー

名前	説明
Cache-Control	オブジェクトがダウンロードされる際の Web ページのキャッシュ動作を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Disposition	オブジェクトがダウンロードされる際のオブジェクトの名前を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Content-Encoding	オブジェクトがダウンロードされる際のコンテンツのエンコーディング形式を指定します。詳細については、『RFC2616』を参照してください。 型: 文字列 デフォルト値: なし
Expires	有効期限をミリ秒単位で指定します。詳細については、『RFC2616』を参照してください。 型: 整数 デフォルト値: なし
x-oss-server-side-encryption	このオブジェクトの各パートのアップロードに使用するサーバー側の暗号化アルゴリズムを指定します。OSS は、アップロードされた各パートをサーバー側の暗号化に基づいて格納します。 型: 文字列 有効な値: AES256

応答の要素

名前	説明
Bucket	マルチパートアップロードイベントを開始したバケットの名前。 型: 文字列 親ノード: InitiateMultipartUploadResult
InitiateMultipartUploadResult	Initiate Multipart Upload リクエストの結果の格納に使用されるコンテナー。 型: コンテナー サブノード: Bucket、Key、UploadId 親ノード: なし

Key	マルチパートアップロードイベントを開始したオブジェクトの名前。 型: 文字列 親ノード: InitiateMultipartUploadResult
UploadId	マルチパートアップロードイベントの一意の ID。 型: 文字列 親ノード: InitiateMultipartUploadResult
EncodingType	返す結果にエンコードタイプを指定します。リクエストの中、エンコードタイプが指定された場合、エンコードされたキーが返されます。 型: 文字列 親ノード: Container

詳細分析

- この操作を使用して認証署名を計算する場合は、” ?uploads” を “CanonicalizedResource” に追加する必要があります。
- Initiate Multipart Upload リクエストでは、標準の HTTP リクエストヘッダー Cache-Control、Content-Disposition、Content-Encoding、Content-Type、Expires と、” x-oss-meta- “というプレフィックスが付いたユーザー定義のヘッダーがサポートされています。これらのヘッダーの具体的な意味については、「PUT Object」インターフェイスを参照してください。
- Initiate Multipart Upload リクエストは、同じ名前の既存のオブジェクトに影響しません。
- Initiate Multipart Upload リクエストを受信すると、サーバーはリクエスト本文を XML 形式で返します。リクエスト本文には Bucket、Key、UploadID の 3 つの要素があります。その後のマルチパート操作のために UploadID を記録しておいてください。
- Initiate Multipart Upload リクエストで x-oss-server-side-encryption ヘッダーが設定されている場合、サーバーはこのヘッダーを応答ヘッダーで返します。各パートのアップロード時に、サーバーはエントロピ暗号化に基づいてそれらのパートを自動的に格納します。現在、OSS サーバーでは 256 ビットの高度暗号化標準 (AES256) のみがサポートされています。その他の標準の値を指定した場合、OSS サーバーはエラー 400 と対応するエラープロンプト InvalidEncryptionAlgorithmError を返します。各パートをアップロードする際に、x-oss-server-side-encryption リクエストヘッダーを追加する必要はありません。このリクエストヘッダーを指定した場合、OSS はエラー 400 と対応するエラープロンプト InvalidArgument を返します。

例

リクエストの例:

```
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:/cluRFtRwMTZpC2hTj4F67AGdM4=
```

戻り値の例:

```
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>
```

UploadPart

Upload Part

マルチパートアップロードイベントを開始したら、指定したオブジェクト名とアップロード ID に基づいて、データをパートに分けてアップロードできます。アップロードされた各パートには、1 ~ 10,000 の範囲のパート番号が割り当てられます。同じアップロード ID の場合、このパート番号によって、このデータパートだけでなくファイル全体におけるこのパートの位置も識別されます。同じパート番号を使用して新しいデータをアップロードすると、このパート番号で識別される既存のデータが上書きされます。最後のパート以外のパートの最小サイズは 100 KB です。最後のパートのサイズには制約はありません。

リクエスト構文

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

詳細分析

- Initiate Multipart Upload インターフェイスを呼び出してデータパートをアップロードする前に、

このインターフェイスを呼び出して、OSS サーバーによって発行されるアップロード ID を取得する必要があります。
- マルチパートアップロードモードでは、最後のパート以外のパートはすべて 100 KB を超えている必要があります。ただし、Upload Part インターフェイスでは、アップロードされたパートのサイズがすぐに検証されるわけではありません (そのパートが最後のパートかどうかがわからないため)。アップロードされたパートのサイズは、マルチパートアップロードが完了して初めて検証されます。
- OSS は、受信したデータパートの MD5 値を ETag ヘッダーに配置し、それをユーザーに返します。
- パート番号の範囲は 1 ~ 10,000 です。パート番号がこの範囲を超えている場合、OSS は InvalidArgument エラーコードを返します。
- Initiate Multipart Upload インターフェイスを呼び出したときに x-oss-server-side-encryption リクエストヘッダーを指定した場合、OSS はアップロードされたパートを暗号化し、Upload Part 応答ヘッダーで x-oss-server-side-encryption ヘッダーを返します。x-oss-server-side-encryption の値は、このパートに使用するサーバー側の暗号化アルゴリズムを示します。詳細については、「Initiate Multipart Upload」インターフェイスを参照してください。
- ネットワーク経由で転送されたデータにエラーがないことを確認するために、リクエストに Content-MD5 を含めます。OSS ではアップロードされたデータの MD5 値が計算され、ユーザーがアップロードした MD5 値と比較されます。それらの値が一致しない場合、OSS は InvalidDigest エラーコードを返します。

例

リクエストの例:

```
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/lICfXEvPmmSW86bBAfMmUmWjI=

[6291456 bytes data]
```

戻り値の例:

```
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
```

UploadPartCopy

UploadPartCopy を使用すると、既存のオブジェクトからデータをコピーし、そのデータのパートをアップロードできます。UploadPartCopy インターフェイスを呼び出すには、Upload Part リクエストに x-oss-copy-source ヘッダーを追加します。1 GB を超えるオブジェクトをコピーする場合は、UploadPartCopy メソッドを使用する必要があります。1 GB 未満のオブジェクトをコピーする場合は、「Copy Object」を参照してください。

> **注：**UploadPartCopy 操作では、ソースバケットとターゲットバケットは同じリージョン内になければなりません。

リクエスト構文

```
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
```

リクエストヘッダー

共通リクエストヘッダーを除く、次に示す Upload Part Copy リクエストのヘッダーを使用して、コピー元のソースオブジェクトのアドレスとコピー範囲を指定します。

名前	説明
x-oss-copy-source	コピー元のソースアドレスを指定します (リクエストするには、ソースオブジェクトの読み取り権限が必要です)。 型: 文字列 デフォルト値: なし
x-oss-copy-source-range	コピー元のソースオブジェクトのコピー範囲。たとえば、範囲を bytes=0-9 に設定した場合は、バイト 0 ～ 9 が転送されます。このリクエストヘッダーは、ソースオブジェクト全体をコピーする場合は不要です。 型: 整数 デフォルト値: なし

次のリクエストヘッダーは、x-oss-copy-source で指定したソースオブジェクトに対して使用します。

名前	説明

x-oss-copy-source-if-match	ソースオブジェクトの ETAG 値とユーザーが提示した ETAG 値が一致する場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-none-match	ユーザーが指定した日時以降にソースオブジェクトが変更されていない場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-unmodified-since	受け取ったパラメーターによって指定された日時がファイルの変更日時と同じかそれ以降である場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-modified-since	ユーザーが指定した日時以降にソースオブジェクトが変更されている場合は、Copy Object 操作が実行されます。それ以外の場合は、412 HTTPエラーメッセージ(indicating preprocessing failure)が返されます。 型: 文字列 デフォルト値: なし

応答の要素

名前	説明
x-oss-copy-source-if-match	ソースオブジェクトの ETAG 値とユーザーが提示した ETAG 値が一致する場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-none-match	ユーザーが指定した日時以降にソースオブジェクトが変更されていない場合は、Copy Object 操作が実行されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-unmodified-since	受け取ったパラメーターによって指定された日時がファイルの変更日時と同じかそれ以降である場合は、ファイルが正常に転送されて 200 OK メッセージが返されます。それ以外の場合は、412 Precondition Failed メッセージが返されます。 型: 文字列 デフォルト値: なし
x-oss-copy-source-if-modified-since	ユーザーが指定した日時以降にソースオブジェク

	トが変更されている場合は、Copy Object 操作が実行されます。それ以外の場合は、412 HTTPエラーメッセージ(indicating preprocessing failure)が返されます。 型: 文字列 デフォルト値: なし

詳細分析

- Initiate Multipart Upload インターフェイスを呼び出してデータパートをアップロードする前に、このインターフェイスを呼び出して、OSS サーバーによって発行されるアップロード ID を取得する必要があります。
- マルチパートアップロードモードでは、最後のパート以外のパートはすべて 100 KB を超えている必要があります。ただし、Upload Part インターフェイスでは、アップロードされたパートのサイズがすぐに検証されるわけではありません (そのパートが最後のパートかどうかがわからないため)。アップロードされたパートのサイズは、マルチパートアップロードが完了して初めて検証されます。
- x-oss-copy-source-range リクエストヘッダーを指定しない場合は、ソースオブジェクト全体がコピーされます。このリクエストヘッダーを指定した場合は、返されるメッセージにファイル全体の長さとコピー範囲が示されます。たとえば、Content-Range: bytes 0-9/44 は、ファイル全体の長さが 44 でコピー範囲が 0 ~ 9 であることを示しています。指定した範囲が範囲ルールに従っていない場合は、ファイル全体がコピーされ、結果に Content-Range は含まれません。
- Initiate Multipart Upload インターフェイスを呼び出したときに x-oss-server-side-encryption リクエストヘッダーを指定した場合、OSS はアップロードされたパートを暗号化し、Upload Part 応答ヘッダーで x-oss-server-side-encryption ヘッダーを返します。x-oss-server-side-encryption の値は、このパートに使用するサーバー側の暗号化アルゴリズムを示します。詳細については、「Initiate Multipart Upload」インターフェイスを参照してください。
- この操作は添付アップロードメソッドにより作成されたobjectをコピーできません。
- バケットタイプがアーカイブの場合、このインターフェースを呼び出すことはできません。それ以外の場合は、エラーコード “OperationNotSupported” とともにエラー400が返されます。

例

リクエストの例:

```
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/lICfXEvPmmSW86bBAfMmUmWjI=
x-oss-copy-source: /oss-example/ src-object
x-oss-copy-source-range:bytes=100-6291756
```

戻り値の例:

```
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>
```

CompleteMultipartUpload

Complete Multipart Upload

すべてのデータパートをアップロードしたら、Complete Multipart Upload API を呼び出して、ファイル全体のマルチパートアップロードを完了する必要があります。この操作では、すべての有効なデータパートのリスト (パート番号と ETag を含む) を提供する必要があります。送信したパートリストを OSS が受信すると、各データパートの有効性が個別に検証されます。すべてのデータパートが検証されると、OSS はこれらのパートを 1 つの完全なオブジェクトに結合します。

リクエスト構文

```
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>
```

リクエスト パラメーター

マルチパートアップロードモードを初期化するとき、encoding-typeパラメーターを利用して返されるエンコードキーを指定可能です。

名前	説明
encoding-type	返されるエンコードキーを指定します。 現在URLエンコードのみサポートします。キーはUTF-8のエンコードを採用します。 型: 文字列 デフォルト値: なし オプション値: url

リクエストの要素

名前	説明
CompleteMultipartUpload	Complete Multipart Upload リクエストのコンテンツの格納に使用されるコンテナー。 型: コンテナー サブノード: 1 つ以上の Part 要素 親ノード: なし
ETag	データパートが正常にアップロードされるとOSS から返される ETag 値。 型: 文字列 親ノード: Part
Part	アップロードされたデータパートの格納に使用されるコンテナー。 型: コンテナー サブノード: ETag、PartNumber 親ノード: InitiateMultipartUploadResult
PartNumber	パート番号。 型: 整数 親ノード: Part

応答の要素

名前	説明
Bucket	バケット名を指定します。 型: 文字列 親ノード: CompleteMultipartUploadResult
CompleteMultipartUploadResult	Complete Multipart Upload リクエストの結果の格納に使用されるコンテナー。 型: コンテナー サブノード: Bucket、Key、ETag、Location 親ノード: なし
ETag	ETag (エンティティタグ) は、オブジェクトの生成時に作成され、オブジェクトのコンテンツを示すために使用されます。オブジェクトは、Complete Multipart Upload リクエストに基づ

	いて作成されます。ETag の値は、オブジェクトのコンテンツの UUID です。この ETag の値を使用して、オブジェクトのコンテンツが変更されているかどうかを確認できます。 型: 文字列 親ノード: CompleteMultipartUploadResult
Location	新たに作成されたオブジェクトの URL を指定します。 型: 文字列 親ノード: CompleteMultipartUploadResult
Key	新たに作成されたオブジェクトの名前。 型: 文字列 親ノード: CompleteMultipartUploadResult
EncodingType	返す結果にエンコードタイプを指定します。リクエストの中、エンコードタイプが指定された場合、エンコードされたキーが返されます。 型: 文字列 親ノード: Container

詳細分析

- OSS は、Complete Multipart Upload リクエストを受信すると、最後のパートを除くすべてのパートが 100 KB より大きいことを確認し、ユーザーから送信されたパートリストの各パート番号と ETag を調べます。したがって、データパートをアップロードする際には、パート番号だけでなく、パートが正常にアップロードされるたびに OSS から返される ETag 値も記録する必要があります。
- OSS の Complete Multipart Upload リクエストの処理には時間がかかります。その間にクライアントが OSS から切断されても、リクエストの処理は続行されます。
- 送信するパートリストのパート番号は、連続していなくてもかまいません (1 つ目のパート番号が 1 で、2 つ目のパート番号が 5 など)。
- Complete Multipart Upload リクエストが正常に処理されると、対応するアップロード ID が無効になります。
- 同じオブジェクトが複数の異なるアップロード ID を持つ場合もあります。いずれかのアップロード ID が完了しても、そのオブジェクトの他のアップロード ID には影響しません。
- Initiate Multipart Upload インターフェイスを呼び出したときに x-oss-server-side-encryption リクエストヘッダーを指定した場合、OSS は Complete Multipart Upload 応答ヘッダーで x-oss-server-side-encryption ヘッダーを返します。x-oss-server-side-encryption の値は、このオブジェクトに使用するサーバー側の暗号化アルゴリズムを示します。
- Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

リクエストの例:

```
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>
```

戻り値の例:

```
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>
```

AbortMultipartUpload

このインターフェイスを使用すると、指定したアップロード ID に基づいてマルチパートアップロードイベントを中止できます。マルチパートアップロードイベントが中止されると、そのアップロード ID を使用して操作を実行できなくなり、アップロードされたデータパートが削除されます。

リクエスト構文

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

詳細分析

- マルチパートアップロードイベントを中止しても、アップロード中のパートは削除されません。したがって、同時アクセスが存在する場合は、Abort Multipart Upload インターフェイスを何度か呼び出して、OSS のスペースを完全にリリースする必要があります。
- 入力したアップロード ID が存在しない場合は、エラー 404 が返されます。エラーコードは NoSuchUpload です。

例

リクエストの例:

```
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/lICfXEvPmmSW86bBAfMmUmWjI=
```

戻り値の例:

```
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
```

ListMultipartUploads

ListMultipartUploads インターフェイスを使用すると、実行中のすべてのマルチパートアップロードイベント、つまり開始されたが完了または中止されていないマルチパートアップロードイベントをリストできます。OSS から返されるリスト結果には、最大 1000 件のマルチパートアップロードメッセージが表示されます。OSS から返されるリスト結果のマルチパートアップロードメッセージ数を指定する場合は、リクエストに

max-uploads パラメーターを追加できます。また、OSS から返されるリスト結果の IsTruncated 要素は、その他のマルチパートアップロードメッセージがあるかどうかを示します。

リクエスト構文

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

リクエストパラメーター

名前	説明
delimiter	オブジェクト名をグループ化するために使用する文字。指定したプレフィックスから最初の delimiter までの名前を持つオブジェクトはすべて要素のグループ (CommonPrefixes) として機能します。 型: 文字列
max-uploads	1 件のリクエストに対して返されるマルチパートアップロードイベントの最大数。指定されていない場合のデフォルト値は 1000 です。max-uploads の値は 1000 以下にする必要があります。 型: 文字列
key-marker	upload-id-marker パラメーターと組み合わせて使用して、返される結果の開始位置を指定します。 l upload-id-marker パラメーターを設定していない場合、クエリ結果には、辞書式順序のすべてのオブジェクト名が key-marker パラメーターの値を超えているマルチパートイベントが表示されます。 l upload-id-marker パラメーターを設定している場合、クエリ結果には、辞書式順序のすべてのオブジェクト名が key-marker パラメーターの値を超えているマルチパートイベントと、オブジェクト名が key-marker パラメーターの値と同じだがアップロード ID が upload-id-marker パラメーターの値を超えているすべてのマルチパートアップロードイベントが表示されます。 型: 文字列
prefix	返されるオブジェクトキーに指定に応じたプレフィックスが付与されるという制限を適用します。プレフィックスを使用するクエリから返されるキーにはそのプレフィックスも含まれることに注意してください。 型: 文字列
upload-id-marker	key-marker パラメーターと組み合わせて使用して、返される結果の開始位置を指定します。

	key-marker を設定していない場合、upload-id-marker は無視されます。 l key-marker を設定している場合、クエリ結果には、辞書式順序のすべてのオブジェクト名が key-marker の値を超えているマルチパートイベントと、オブジェクト名が key-marker の値と同じだがアップロード ID が upload-id-marker の値を超えているすべてのマルチパートアップロードイベントが表示されます。 型: 文字列
encoding-type	返されるコンテンツのエンコーディングとエンコーディングのタイプを指定します。オブジェクトキーでは UTF-8 文字が使用されますが、XML 1.0 標準では、0 ~ 10 の ASCII 値を含む文字などの特定の制御文字の解析はサポートされていません。XML 1.0 標準でサポートされていない制御文字がオブジェクトキーに含まれている場合は、encoding-type を指定して返されるオブジェクトキーをエンコードできます。 データ型: 文字列 デフォルト値: なし

応答の要素

名前	説明
ListMultipartUploadsResult	ListMultipartUploads リクエストの結果の格納に使用されるコンテナー。 型: コンテナー サブノード: Bucket、KeyMarker、UploadIdMarker、NextKeyMarker、NextUploadIdMarker、MasUploads、Delimiter、Prefix、CommonPrefixes、IsTruncated、Upload 親ノード: なし
Bucket	バケット名を指定します。 型: 文字列 親ノード: ListMultipartUploadsResult
EncodingType	返される結果におけるエンコーディングのタイプを指定します。リクエストで encoding-type を指定した場合は、返される結果において Delimiter、KeyMarker、Prefix、NextKeyMarker、Key などの要素がエンコードされます。 型: 文字列 親ノード: ListMultipartUploadsResult
KeyMarker	リスト内の開始オブジェクトの位置。 型: 文字列 親ノード: ListMultipartUploadsResult
UploadIdMarker	リスト内の開始アップロード ID の位置。 型: 文字列 親ノード: ListMultipartUploadsResult

NextKeyMarker	今回返されなかった結果がある場合、応答リクエストには、次のリクエストにおける KeyMarker の値を示す NextKeyMarker 要素が含まれます。 型: 文字列 親ノード: ListMultipartUploadsResult
NextUploadMarker	今回返されなかった結果がある場合、応答リクエストには、次のリクエストにおける UploadMarker の値を示す NextUploadMarker 要素が含まれます。 型: 文字列 親ノード: ListMultipartUploadsResult
MaxUploads	OSS から返されるアップロードの最大数。 型: 整数 親ノード: ListMultipartUploadsResult
IsTruncated	返されたマルチパートアップロードの結果リストが切り捨てられているかどうかを指定します。”true” は返されなかった結果があることを示し、”false” はすべての結果が返されたことを示します。 型: 列挙文字列 有効な値: true と false デフォルト値: False 親ノード: ListMultipartUploadsResult
Upload	マルチパートアップロードイベントに関する情報の格納に使用されるコンテナー。 型: コンテナー サブノード: Key、UploadId、Initiated 親ノード: ListMultipartUploadsResult
Key	マルチパートアップロードイベントを開始したオブジェクトの名前。 型: 文字列 親ノード: Upload
UploadId	マルチパートアップロードイベントの ID。 型: 文字列 親ノード: Upload
Initiated	マルチパートアップロードイベントの開始日時。 型: 日付 親ノード: Upload

詳細分析

- “max-uploads” パラメーターの最大値は 1000 です。
- OSS から返される結果は、オブジェクト名の辞書式順序に基づいて昇順でリストされます。同じオブジェクトの場合、結果は時間の昇順でリストされます。
- prefix パラメーターを使用すると、バケット内のオブジェクトをグループ単位で柔軟に管理できます (フォルダー機能に似ています)。
- List Multipart Uploads リクエストでは、prefix、marker、delimiter、upload-id-marker、max-keys の 5 つのリクエストパラメーターがサポートされています。これらのパラメーターの組み合

わせに基づいて、マルチパートアップロードイベントを照会して目的のクエリ結果を得るためのルールを設定できます。

例

リクエストの例:

```
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+dcivn7kwBMvOY=
```

応答の例:

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

ListParts

List Parts

ListParts コマンドを使用すると、正常にアップロードされた特定のアップロード ID のすべてのパートをリストできます。

リクエスト構文

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

リクエストパラメーター

名前	説明
uploadId	マルチパートアップロードイベントの ID。 型: 文字列 デフォルト値: なし
max-parts	OSS の応答に表示されるパートの最大数。 型: 整数 デフォルト値: 1,000
part-number-marker	特定のリストの開始位置。パートは、そのパート番号がこのパラメーターの値より大きい場合にのみリストされます。 型: 整数 デフォルト値: なし
encoding-type	返されるコンテンツのエンコーディングとエンコーディングのタイプを指定します。オブジェクトキーでは UTF-8 文字が使用されますが、XML 1.0 標準では、0 ~ 10 の ASCII 値を含む文字などの特定の制御文字の解析はサポートされていません。XML 1.0 標準でサポートされていない制御文字がオブジェクトキーに含まれている場合は、encoding-type を指定して返されるオブジェクトキーをエンコードできます。 データ型: 文字列 デフォルト値: なし

応答の要素

名前	説明
ListPartsResult	List Parts リクエストの結果の格納に使用されるコンテナー。 型: コンテナー サブノード: Bucket、Key、UploadId、PartNumberMarker、NextPartNumberMarker、MaxParts、IsTruncated、Part 親ノード: なし
Bucket	バケット名を指定します。 型: 文字列 親ノード: ListPartsResult
EncodingType	返される結果におけるエンコーディングのタイプを指定します。リクエストで encoding-type を指定した場合は、返される結果において Key がエンコードされます。 型: 文字列 親ノード: ListPartsResult
Key	オブジェクト名。 型: 文字列 親ノード: ListPartsResult
UploadId	アップロードイベントの ID。 型: 文字列 親ノード: ListPartsResult
PartNumberMarker	リスト結果内のパート番号の開始位置。 型: 整数 親ノード: ListPartsResult
NextPartNumberMarker	今回返されなかった結果がある場合、応答リクエストには、次のリクエストにおける PartNumberMarker の値を示す NextPartNumberMarker 要素が含まれます。 型: 整数 親ノード: ListPartsResult
MaxParts	返されたリクエストに表示されるパートの最大数。 型: 整数 親ノード: ListPartsResult
IsTruncated	List Parts に対して返された結果リストが切り捨てられているかどうか。”true” は返されなかった結果があることを示し、”false” はすべての結果が返されたことを示します。 型: 列挙文字列 有効な値: true と false 親ノード: ListPartsResult
Part	パート情報の格納に使用されるコンテナー。 型: 文字列 サブノード: PartNumber、LastModified、ETag、Size

	親ノード: ListPartsResult
PartNumber	パート番号。 型: 整数 親ノード: ListPartsResult.Part
LastModified	パートのアップロード日時。 型: 日付 親ノード: ListPartsResult.part
ETag	アップロードされたパートのコンテンツ内の ETag 値。 型: 文字列 親ノード: ListPartsResult.Part
Size	アップロードされたパートのサイズ。 型: 整数 親ノード: ListPartsResult.Part

詳細分析

- ListParts では、max-parts と part-number-marker の 2 つのリクエストパラメーターがサポートされています。
- max-parts パラメーターの最大値は 1,000 で、デフォルト値も 1,000 です。
- OSS から返される結果は、パート番号に基づいて昇順でリストされます。
- ネットワーク伝送でエラーが発生する可能性があるため、List Parts の結果 (パート番号と ETag 値) を使用して Complete Multipart の最終パートリストを生成しないことをお勧めします。

例

リクエストの例:

```
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:4qOnUMc9UQWqkz8wDqD3lIsa9P8=
```

応答の例:

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

Cross-Origin Resource Sharing

はじめに

CORS (Cross-Origin Resource Sharing) を使用すると、Web アプリケーションが他のドメインのリソースにアクセスできるようになります。この CORS のサポートにより、OSS では、より柔軟な Web アプリケーションの開発が実現されます。OSS のインターフェイスを使用して、クロスドメインアクセスのさまざまな権限を簡単に制御できます。

PutBucketcors

PutBucketcors 操作を使用すると、指定したバケットの CORS ルールを設定できます。元のルールが存在す

る場合は上書きされます。

リクエスト構文

```
PUT /?cors HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
<CORSRule>
<AllowedOrigin>the origin you want allow CORS request from</AllowedOrigin>
<AllowedOrigin>...</AllowedOrigin>
<AllowedMethod>HTTP method</AllowedMethod>
<AllowedMethod>...</AllowedMethod>
<AllowedHeader> headers that allowed browser to send</AllowedHeader>
<AllowedHeader>...</AllowedHeader>
<ExposeHeader> headers in response that can access from client app</ExposeHeader>
<ExposeHeader>...</ExposeHeader>
<MaxAgeSeconds>time to cache pre-fight response</MaxAgeSeconds>
</CORSRule>
<CORSRule>
...
</CORSRule>
...
</CORSConfiguration >
```

リクエストの要素

名前	説明	必須
CORSRule	CORS ルールのコンテナー。バケットごとに最大 10 のルールが許可されます。 型: container 親ノード: CORSConfiguration	はい
AllowedOrigin	クロスドメインリクエストが許可されているオリジンを示します。複数の要素を使用して、許可されるオリジンを複数指定できます。 各ルールでワイルドカード “* ” を 1 つ使用できます。” * “ が指定されている場合、すべてのオリジンのクロスドメインリクエストが許可されます。 型: string 親ノード: CORSRule	はい

AllowedMethod	クロスドメインリクエストで許可されるメソッドを指定します。 型: enumeration (GET、PUT、DELETE、POST、HEAD) 親ノード: CORSRule	はい
AllowedHeader	OPTIONS プリフェッチコマンドの Access-Control-Request-Headers で指定されたヘッダーを許可するかどうかを制御します。Access-Control-Request-Headers で指定された各ヘッダーが AllowedHeader の値と一致している必要があります。各ルールでワイルドカード “* ” を 1 つ使用できます。 型: string 親ノード: CORSRule	いいえ
ExposeHeader	応答ヘッダーを指定して、ユーザーがアプリケーション (Javascript の XMLHttpRequest オブジェクトなど) からアクセスできるようにします。 ワイルドカード “* ” は使用できません。 型: string 親ノード: CORSRule	いいえ
MaxAgeSeconds	特定のリソースに対するブラウザープリフェッチ (OPTIONS) リクエストの結果のキャッシュ期間を指定します。単位は秒です。 CORSRule ごとに 1 つ指定できます。 型: integer 親ノード: CORSRule	いいえ
CORSConfiguration	バケットの CORS ルールのコンテナー。 型: container 親ノード: なし	はい

詳細分析

- デフォルトでは、CORS はバケットで無効になっており、 すべてのクロスドメインリクエストのオリジンが禁止されます。
- アプリケーションで CORS を使用するには (ブラウザーの XMLHttpRequest 機能を使用して www.a.com から OSS にアクセスする場合など)、このインターフェイスを使用して手動で CORS ルールをアップロードして、CORS を有効にする必要があります。このルールは XML ドキュメントに記述します。
- 各バケットの CORS 設定は複数の CORS ルールによって指定されます。バケットごとに最大 10 の

ルールが許可されます。アップロードする XML ドキュメントはサイズが 16 KB 以下である必要があります。
- OSS は、クロスドメインリクエスト (または OPTIONS リクエスト) を受信すると、バケットの CORS ルールを読み取り、関連する権限を確認します。各ルールを順番に調べて、最初に一致したルールを使用してリクエストを承認し、対応するヘッダーを返します。一致するルールがない場合、CORS ヘッダーは含まれません。
- CORS ルールに一致するには 3 つの条件を満たしている必要があります。まず、リクエストの Origin が AllowedOrigin と一致している必要があります。次に、リクエストのメソッド (GET、PUT など) や、OPTIONS リクエストの Access-Control-Request-Method ヘッダーに対応するメソッドが、AllowedMethod と一致している必要があります。最後に、OPTIONS リクエストの Access-Control-Request-Headers に含まれている各ヘッダーが AllowedHeader と一致している必要があります。
- Content-MD5 リクエストヘッダーをアップロードした場合は、本文の Content-MD5 が計算されて、両者が同じであるかどうかが確認されます。異なる場合は、エラーコード InvalidDigest が返されます。

例

バケットの CORS ルールを追加する例:

```
PUT /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:KU5h8YMUC78M30dXqf3JxrTZHiA=

<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
<CORSRule>
<AllowedOrigin>*</AllowedOrigin>
<AllowedMethod>PUT</AllowedMethod>
<AllowedMethod>GET</AllowedMethod>
<AllowedHeader>Authorization</AllowedHeader>
</CORSRule>
<CORSRule>
<AllowedOrigin>http://www.a.com</AllowedOrigin>
<AllowedOrigin>http://www.b.com</AllowedOrigin>
<AllowedMethod>GET</AllowedMethod>
<AllowedHeader> Authorization</AllowedHeader>
<ExposeHeader>x-oss-test</ExposeHeader>
<ExposeHeader>x-oss-test1</ExposeHeader>
<MaxAgeSeconds>100</MaxAgeSeconds>
</CORSRule>
</CORSConfiguration >
```

応答の例:

```
HTTP/1.1 200 OK
```

```
x-oss-request-id: 50519080C4689A033D00235F
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

GetBucketcors

GetBucketcors 操作を使用すると、指定したバケットの現在の CORS ルールを取得できます。

リクエスト構文

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

応答の要素

名前	説明
CORSRule	CORS ルールのコンテナー。バケットごとに最大 10 のルールが許可されます。 型: container 親ノード: CORSConfiguration
AllowedOrigin	クロスドメインリクエストが許可されているオリジンを示します。複数の要素を使用して、許可されるオリジンを複数指定できます。 各ルールでワイルドカード “* ” を 1 つ使用できます。 ” * ” が指定されている場合、すべてのオリジンのクロスドメインリクエストが許可されます。 型: string 親ノード: CORSRule
AllowedMethod	クロスドメインリクエストで許可されるメソッドを指定します。 型: enumeration (GET、PUT、DELETE、POST、HEAD) 親ノード: CORSRule
AllowedHeader	OPTIONS プリフェッチコマンドの Access-Control-Request-Headers で指定されたヘッダーを許可するかどうかを制御します。Access-Control-Request-Headers で指定された各ヘッダーが AllowedHeader の値と一致している必要があります。各ルールでワイルドカード “* ” を

	1 つ使用できます。 型: string 親ノード: CORSRule
ExposeHeader	応答ヘッダーを指定して、ユーザーがアプリケーション (Javascript の XMLHttpRequest オブジェクトなど) からアクセスできるようにします。ワイルドカード “* ” は使用できません。 型: string 親ノード: CORSRule
MaxAgeSeconds	特定のリソースに対するブラウザープリフェッチ (OPTIONS) リクエストの結果のキャッシュ期間を指定します。単位は秒です。 CORSRule ごとに 1 つ指定できます。 型: integer 親ノード: CORSRule
CORSConfiguration	バケットの CORS ルールのコンテナー。 型: container 親ノード: なし

詳細分析

1. バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
2. CORS ルールを取得できるのはバケットオーナーだけです。他のユーザーが表示しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。
3. CORS ルールが存在しない場合は、” 404 Not Found” エラーが返されます。エラーコードは NoSuchCORSConfiguration です。

例

リクエストの例:

```
Get /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc: BuG4rRK+zNhH1AcF51NNHD39zXw=
```

応答の例 (CORS ルールが既に設定されている場合):

```
HTTP/1.1 200
x-oss-request-id: 50519080C4689A033D00235F
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
Content-Length: 218
Server: AliyunOSS
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration>
<CORSRule>
<AllowedOrigin>*</AllowedOrigin>
<AllowedMethod>GET</AllowedMethod>
<AllowedHeader>*</AllowedHeader>
<ExposeHeader>x-oss-test</ExposeHeader>
<MaxAgeSeconds>100</MaxAgeSeconds>
</CORSRule>
</CORSConfiguration>
```

DeleteBucketcors

Delete Bucket cors を使用すると、指定したバケットで CORS 機能を無効にして、すべてのルールを削除できます。

リクエスト構文

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

詳細分析

- バケットが存在しない場合は、エラー “404 no content” が返されます。エラーコードは NoSuchBucket です。
- バケットの CORS ルールを削除できるのはバケットオーナーだけです。自分のものではないバケットを操作しようとすると、エラー 403 Forbidden が返されます。エラーコードは AccessDenied です。

例

リクエストの例:

```
DELETE /?cors HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:LnM4AZ1OeIduZF5vGFWicOMEkVg=
```

応答の例:

```
HTTP/1.1 204 No Content
x-oss-request-id: 5051845BC4689A033D0022BC
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

OptionObject

ブラウザーは、クロスドメインリクエストを送信する前に、特定のオリジン、HTTP メソッド、ヘッダー情報を含むプリフライトリクエスト (OPTIONS) を OSS に送信して、実際のリクエストを送信するかどうかを決定します。

OSS では、Put Bucket cors インターフェイスを使用してバケットで CORS を有効にすることができます。CORS を有効にすると、ブラウザーのプリフライトリクエストを許可するかどうかが、指定したルールに基づいて評価されるようになります。このリクエストが許可されない場合や、CORS が無効になっている場合は、エラー 403 Forbidden が返されます。

リクエスト構文

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.oss-cn-hangzhou.aliyuncs.com
Origin:Origin
Access-Control-Request-Method:HTTP method
Access-Control-Request-Headers:Request Headers
```

リクエストヘッダー

名前	説明
Origin	リクエストのオリジン。クロスドメインリクエストを識別するために使用されます。 型: string デフォルト値: なし
Access-Control-Request-Method	実際のリクエストで使用されるメソッド。 型: string デフォルト値: なし
Access-Control-Request-Headers	実際のリクエストで使用されるヘッダー (単純なヘッダーは除く)。 型: string

	デフォルト値: なし

応答ヘッダー

名前	説明
Access-Control-Allow-Origin	リクエストに含まれるオリジン。このヘッダーは、このリクエストが許可されない場合は含まれません。 型: string
Access-Control-Allow-Methods	リクエストで使用される HTTP メソッド。このヘッダーは、このリクエストが許可されない場合は含まれません。 型: string
Access-Control-Allow-Headers	リクエストに含まれるヘッダーリスト。禁止されているヘッダーがリクエストに含まれている場合は、このヘッダーは含まれず、リクエストが拒否されます。 型: string
Access-Control-Expose-Headers	クライアントの JavaScript アプリケーションからアクセスできるヘッダーリスト。 型: string
Access-Control-Max-Age	ブラウザーがプリフライト結果をバッファーに保持できる期間。単位は秒です。 型: integer

例

リクエストの例:

```
OPTIONS /testobject HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Origin:http://www.example.com
Access-Control-Request-Method:PUT
Access-Control-Request-Headers:x-oss-test
```

応答の例:

```
HTTP/1.1 200 OK
x-oss-request-id: 5051845BC4689A033D0022BC
Date: Fri, 24 Feb 2012 05:45:34 GMT
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-oss-test
Connection: keep-alive
Content-Length: 0
```

```
Server: AliyunOSS
```