ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

Object Storage Service

Developer Guide

Product Version: v3.16.2

Document Version: 20220913

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Usage notes

Before using an OSS API to implement a specific function, developers need to understand the two methods that are used to call OSS API operations and choose the appropriate method according to the actual situation.

> **Notice** If you encounter problems during use, contact Alibaba Cloud technical support.

OSS API operations can be called by using ASAPI or OSS SDKs. Select an appropriate calling method based on your actual scenarios.

- If you want to perform operations on OSS buckets in a resource set of a specified organization, such as creating buckets, use ASAPI SDKs to call the corresponding OSS API operation. For more information, see related topics in ASAPI Developer Guide of *OSS Developer Guide*.
- If you want to perform operations on objects (such as object upload) in a bucket without the resource set and organization specified, you can use an OSS SDK to call the specific operation. For more information, see related topics in SDK Reference of *OSS Developer Guide*.

> **Notice** To perform operations on objects in a bucket, you must have the required permissions.

2.API reference

2.1. Overview

Apsara Stack Object Storage Service (OSS) is a secure, cost-effective, and highly reliable cloud storage service. It enables you to store a large amount of data in the cloud. You can upload data to and download data from OSS anytime, anywhere, and on any Internet device by calling the simple RESTful APIs. OSS enables you to develop a variety of services that are capable of processing large amounts of data. These services include multimedia sharing websites, online storage, and personal and corporate data backups.

> **Note** Before you call these operations, ensure that you understand the OSS service instructions and usage agreements.

Instructions

OSS API reference provides request syntax, parameters, sample requests, and sample responses for operations. To implement secondary development, use SDK packages. For more information about how to install and use SDK, see OSS SDK reference.

Terms

Term	Description
bucket	The container used to store objects in OSS. Every object is contained in a bucket.
object	Objects are the basic unit for data operations in OSS. Objects are also known as files. Each object has metadata, data, and a key. A key can uniquely identify an object in a bucket.
endpoint	The domain name used to access OSS. OSS uses HTTP RESTful APIs to provide services. Different regions are accessed by using different endpoints. A region has different endpoints for access over the internal network and for access over the Internet. For more information, see Obtain an endpoint.
AccessKey	The access credential that is used to identify the requester. An AccessKey pair consists of an AccessKey ID and an AccessKey secret. The AccessKey ID is used to verify the identity of the user, while the AccessKey secret is used to encrypt and verify the signature string. You must keep your AccessKey secret confidential. For more information, see Obtain an AccessKey pair.

2.2. Access control

2.2.1. Authenticate users by using signatures

OSS uses an AccessKey pair, which includes an AccessKey ID and an AccessKey secret to implement symmetric encryption and verify the identity of a requester. The AccessKey ID is used to verify the identity of the user, while the AccessKey secret is used to encrypt and verify the signature string. You must keep your AccessKey secret confidential.

> **Note** For more information about how to obtain an AccessKey pair, see Obtain an AccessKey pair.

AccessKey pairs can be categorized based on the account types.

- AccessKey pair of an Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has full permissions on its resources.
- AccessKey pair of a RAM user: A RAM user is created and authorized by an Alibaba Cloud account. The AccessKey pair of a RAM user has limited permissions on specified resources.
- AccessKey pair of an STS temporary access credential: A temporary access credential is generated by an Alibaba Cloud account or a RAM user. The AccessKey pair of a temporary credential has limited permissions on specified resources within the validity period of the credential. An STS temporary access credential becomes invalid after its validity period is exceeded.

Before you send a request to OSS as an individual user, you must generate a signature string in the specified format for the request and use your AccessKey secret to encrypt the signature string and generate a verification code. After OSS receives the request, OSS finds the AccessKey secret based on your AccessKey ID, and uses the AccessKey secret to decrypt the signature string and verification code. Then, OSS calculates a verification code and compares it against the decrypted verification code. If the two verification codes are the same, OSS determines that the request is valid. Otherwise, OSS rejects the request and returns HTTP status code 403.

2.2.2. Add signatures to the Authorization header

You can include the Authorization header in the HTTP request to carry signature information and indicate that the requester has been authorized.

Calculation of the Authorization header

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

- `AccessKeySecret`: the key required for a signature.
- `VERB`: the HTTP request method such as PUT, GET, POST, HEAD, or DELETE.
- `\n`: the line break.
- `Content-MD5`: optional. This parameter specifies the Content-MD5 value of the message content (excluding headers). The value is obtained by calculating the 128-bit hash value and then encoding the value in Base64. This request header can be used for the message validity check. The message content is valid if the received message content is the same as the content that is sent. Example: eB5eJF1ptWaXm4bijSPyxw==. For more information, visit RFC 2616 Content-MD5.

- `Content-Type`: optional. This parameter specifies the type of the requested content. Example: application/octet-stream.
- `Date`: the time when the request is sent. The time must be in GMT. Example: Sun, 22 Nov 2015 08:16:38 GMT.
- `CanonicalizedOSSHeaders`: the HTTP headers in alphabetical order. The prefix of the HTTP headers is x-oss-.
- `CanonicalizedResource`: the OSS resource you want to access.

Specifically, the values of Date and CanonicalizedResource are required. If the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes, the OSS server rejects the request and returns HTTP status code 403.

Create CanonicalizedOSSHeaders

All HTTP headers whose prefixes are x-oss- are called canonicalized OSS headers. Process of creating canonicalized OSS headers:

1. The names of all HTTP request headers whose prefixes are x-oss- must be in lowercase letters. For example, `X-OSS-Meta-Name: TaoBao` is converted to `x-oss-meta-name: TaoBao`.
2. If the request is sent by using the AccessKey ID and AccessKey secret obtained by using STS, you must add the obtained security-token value to the signature string in the `x-oss-security-token:security-token` format.
3. Sort all obtained HTTP request headers in alphabetical order.
4. Delete any space on either side of a delimiter between the request header and content. For example, `x-oss-meta-name: TaoBao` is converted to `x-oss-meta-name:TaoBao`.
5. Separate all content and headers with the (`\n`) delimiter to form the final canonicalized OSS headers.

> **Note**
> - CanonicalizedOSSHeaders is optional. The (`\n`) delimiter at the end can be removed.
> - To create only one canonicalized OSS header, add the (`\n`) delimiter to the end of the header. Example: `x-oss-meta-a\n`.
> - To create multiple canonicalized OSS headers, add the (`\n`) delimiter to the end of each header. Example: `x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n`.

Create CanonicalizedResource

Destination OSS resources specified in the request sent by the user is called canonicalized resources. Process of creating canonicalized resources:

1. Set CanonicalizedResource to empty character string ("").
2. Specify the OSS resource you want to access in the following format: `/BucketName/ObjectName`. Set CanonicalizedResource to `/BucketName/` to access a bucket or to the forward slash (`/`) delimiter to access OSS.
3. If the requested resource contains subresources, sort all subresources in alphabetical order and separate them with the ampersand (`&`) delimiter. The question mark (`?`) and subresource string are added to the end of the CanonicalizedResource string. In this case, CanonicalizedResource is in the format of `/BucketName/ObjectName?acl&uploadId=UploadId`
4. If query strings (also called HTTP request parameters) are specified in the request, sort these query strings and request values in alphabetical order. Separate each pair of parameters and values with the ampersand (`&`) delimiter. Add the concatenated content to CanonicalizedResource as parameters. In this case, CanonicalizedResource is in the format of `/BucketName/ObjectName?acl&response-content-type=ContentType&uploadId=UploadId`.

> **Note**
> - Subresources supported by OSS include but are not limited to: 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, and response-content-encoding.
> - Three types of subresources are available:
>   - Resource identifiers such as acl, append, uploadId, and symlink subresources
>   - Subresources that specify the response header fields. Example: `response-***`.
>   - Methods to process objects. Example: `x-oss-process`.

Rules to calculate signatures for the Authorization header

1. A signature string must be in the UTF-8 format. A signature string that contains Chinese characters must be encoded in UTF-8. The encoded signature string is used together with AccessKeySecret to calculate the final signature.
2. The HMAC-SHA1 method defined in RFC 2104 is used to calculate the final signature. In this method, the Key is `AccessKeySecret`.
3. Content-Type and Content-MD5 are optional in a request. If the request requires signature verification, the null value must be replaced with a line break (`\n`).
4. Among all non-standard HTTP headers, only headers that start with `x-oss-` require signature strings. Other non-standard HTTP headers are ignored by OSS. For example, the x-oss-magic header in the preceding example requires a signature string.
5. Headers that start with `x-oss-` must comply with the following conventions before signature verification:
   - The header names must be in lowercase.
   - The headers are sorted in alphabetical order.
   - No spaces exist on either side of a colon (:) between the header name and value.
   - Each header is followed by a line break (`\n`). If no headers are used, `CanonicalizedOSSHeaders` is set to null.

Examples

Assume that the AccessKey ID is `4***7`, and the AccessKey secret is `O***V`.

Signature examples

Request	Signature string calculation formula	Signature string
`PUT /nelson HTTP/1.0 Content-MD5: eB5eJF1ptWaXm4bijSPyxw== Content-Type: text/html Date: Thu, 17 Nov 2005 18:49:58 GMT Host: oss-example.regionid.example.com X-OSS-Meta-Author: ***@bar.com X-OSS-Magic: a***a`	`Signature = base64(hmac-sha1(AccessKeySecret,VERB + "\n" + Content-MD5 + "\n"+ Content-Type + "\n" + Date + "\n" + CanonicalizedOSSHeaders+ CanonicalizedResource))`	`"PUT\n ***w==\n text/html\n Thu, 17 Nov 2005 18:49:58 GMT\n x-oss-magic:a***a\nx-oss-meta-author:***@bar.com\n/oss-example/nelson"`

An example of the signature calculation method:

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

The signature calculation result must be 2***A= based on Authorization = "OSS" + AccessKeyId + ":" + Signature. Authorization that consists of OSS 4***7:2***A= and the Authorization header is sent:

```
PUT /nelson HTTP/1.0
Authorization:OSS 4***7:2***A=
Content-Md5: e***w==
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.regionid.example.com
X-OSS-Meta-Author: ***@bar.com
X-OSS-Magic: a***a
```

Detail analysis

- If the imported AccessKey ID does not exist or is inactive, 403 Forbidden is returned. Error code: InvalidAccessKeyId.
- If the Authorization value format in the request header is incorrect, 400 Bad Request is returned. Error code: InvalidArgument.
- All OSS requests must use GMT stipulated in HTTP/1.1. The date format is `date1 = 2DIGIT SP month SP 4DIGIT; day month year`. Example. 02 Jun 1982. In this format, day uses two digits. Therefore, Jun 2, 2 Jun 1982, and 2-Jun-1982 are all invalid formats.
- If no date is entered or the date is in the invalid format during signature verification, 403 Forbidden is returned. Error code: AccessDenied.
- The difference between the time on the client when the request is sent and the time on the server when the request is received must be within 15 minutes. Otherwise, 403 Forbidden is returned. Error code: RequestTimeTooSkewed.
- If the AccessKey ID is active but OSS determines that the signature of the request is invalid, 403 Forbidden is returned. The correct signature string for verification and encryption is returned in the response. You can check whether the signature string is valid based on the response of OSS. Sample response:

```
<? 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 6
3 6c
 </StringToSignBytes>
 <RequestId>
 1***2
 </RequestId>
 <HostId>
 regionid.example.com
 </HostId>
 <SignatureProvided>
 y***8=
 </SignatureProvided>
 <StringToSign>
 GET
Wed, 11 May 2011 07:59:25 GMT
/oss-example? acl
 </StringToSign>
 <OSSAccessKeyId>
 A***Q
 </OSSAccessKeyId>
</Error>
```

OSS SDK automatically implements signatures in your requests. You do not need to manually calculate your signature when you use OSS SDK. For more information about the signature implementation for specific programming languages, see the OSS SDK code. The following table describes the sample code used to implement signatures in various SDKs.

Sample code used to implement signatures in various SDKs

SDK	Signature implementation
Java SDK	OSSRequestSigner.java

SDK	Signature implementation
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

FAQ

Example: Calculate the Content-MD5 value of message content "123456789". Correct method:

1. Calculate the 128-bit MD5 hash.
2. Encode the 128-bit MD5 hash (but not the 32-bit string) in Base64.

Python is used in this example. Correct calculation code:

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

Note that you must use hash.digest() to calculate the 128-bit MD5 hash.

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

A common incorrect method is to directly encode the 32-bit string in Base64. Example: Call the hash.hexdigest() function to obtain a printable 32-bit string.

```
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
```

Result of encoding the incorrect MD5 value in Base64:

```
>>> base64.b64encode(hash.hexdigest())
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

2.2.3. Generate a signed URL

In addition to using an Authorization field in the header, you can also add signature information to a URL so that you can forward the URL to a third party for authorized access.

Implementation

- The following code provides an example on how to generate a signed URL:

  ```
  http://oss-example.regionid.example.com/oss-api.pdf?
  AccessKeyId=AccessKeyId&Expires=1141889120&Signature=vjb***%3D
  ```

- A signed URL must include at least the following parameters: Signature, Expires, and AccessKeyId.
  - `Expires` specifies the timeout time of the URL. The value of this parameter follows the UNIX time format and must be in UTC. The value is the number of seconds that have elapsed since 00:00:00 Thursday, 1 January 1970. If the time OSS receives the URL request is later than the value of Expires that is included in the signature, a request timeout error code is returned. For example, the current time is 1141889060. To create a URL that is scheduled to expire in 60 seconds, you can set the value of Expires to 1141889120.
  - `AccessKeyId` specifies the AccessKey ID in the AccessKey pair.

- `Signature` specifies the signature information. For all requests and header fields that OSS supports, the algorithm of signing the string for a URL is basically the same as that of signing the string for a header.

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

  The following differences are listed:

  - When a signed URL is generated, the Expires parameter replaces the Date parameter.
  - Signatures cannot be included in a URL and a header at the same time.
  - If more than one Signature, Expires, or AccessKeyId value is imported, the first imported value is used.
  - Before OSS authenticates a request by using a signature, OSS checks the request time to determine whether it is later than the time specified in Expires.
  - When you add a signature string to a URL, you must encode the URL.

- When you add a signature to a URL for temporary access, you must include `security-token`. Format:

  ```
  http://oss-example.regionid.example.com/oss-api.pdf?
  AccessKeyId=AccessKeyId&Expires=1141889120&Signature=vjb***%3D&security-token=SecurityToken
  ```

Sample code

The following code provides an example on how to add a signature to a URL in Python:

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

OSS SDK provides methods for adding a signature to a URL. For more information, see the "Authorized access" topic in SDK documents. The following table describes the implementation of a signed URL for OSS SDK.

URL signature implementation files for OSS SDK

SDK	URL signature method	Implementation file
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

Detail analysis

- If you adopt the approach of generating a signed URL, the authorized data is exposed on the Internet before the authorization period is exceeded. We recommend that you assess the use risks in advance.
- You can add a signature to a URL in PUT and GET requests.
- When a signature is added to a URL, the sequence of the Signature, Expires, and AccessKeyId parameters can be swapped. If one or more of the Signature, Expires, and AccessKeyId parameters are missing, 403 Forbidden is returned. Error code: AccessDenied.
- If the current access time is later than the Expires value that is set in the request or the time format is invalid, 403 Forbidden is returned. Error code: AccessDenied.
- If a URL includes one or more of the Signature, Expires, and AccessKeyId parameters and the header includes signature information, 400 Bad Request is returned. Error code: InvalidArgument.
- When the signature string is generated, Date is replaced by Expires, but the headers such as content-type and content-md5 defined in the preceding section are still included. (Although Date still exists in the request header, you do not need to add Date to the signature string.)

2.2.4. Authorize temporary access

OSS allows you to use STS to authorize temporary access. You can use STS to grant a third-party application or your RAM user an access credential that specifies the custom validity period and permissions.

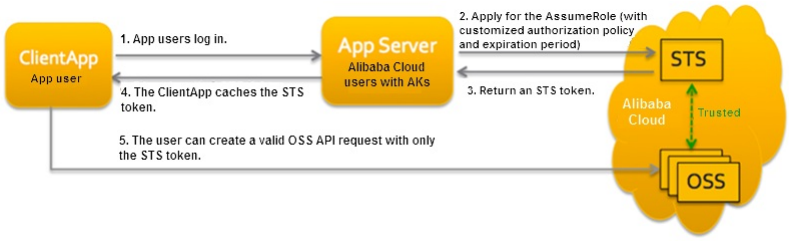
Introduction to STS

STS is a web service that provides temporary access tokens for cloud computing users. You can use STS to grant a third-party application or federated user (whose user ID is managed by yourself) an access credential with a custom validity period and permissions. The third-party application or federated user can use the access credential to directly call the operations provided by Alibaba Cloud services or use the Alibaba Cloud service SDKs to access the operations.

- You do not need to expose your long-term AccessKey pair to a third-party application. You need only to generate an access token and send it to the third-party application. You can customize the permissions and validity period of this token.
- The access token automatically becomes invalid when it expires.

The following figure uses an application as an example to show the authorization process.

Authorization process



Detailed solution description:

1. Log on as an application user. You can manage your application user ID. You can customize the ID management system or use an external web account or open ID. AppServer can define the principle of least privilege (POLP) of each valid application user.
2. AppServer requests a security token from STS. Before you call STS API operations, AppServer determines the POLP (described in policy syntax) of the app user and the validity period of the temporary security token. Then, AppServer calls the AssumeRole API operation of STS to obtain the security token. For more information about how to manage and use roles, see "Role management" in RAM User Guide.
3. STS returns a valid access credential to AppServer, including a security token, a temporary AccessKey pair (AccessKey ID and AccessKey secret), and the expiration time.
4. AppServer returns the access credential to ClientApp. ClientApp caches this credential. When the credential becomes invalid, ClientApp must request a new valid access credential from AppServer. For example, if the access credential is valid for one hour, ClientApp can request AppServer to update the access credential every 30 minutes.
5. ClientApp uses the cached access credential to call API operations provided by Alibaba Cloud services. Cloud services detect the STS access credential, verify the credential based on the STS service, and respond to the user request.

For more information about STS security tokens, see "Role management" in RAM User Guide. The key of the process is to call the AssumeRole operation of STS to obtain a valid access credential. You can also use STS SDK to call this API operation.

Use an STS credential to create a request that includes a signature

After the client of the user obtains an STS temporary credential, the client of the user uses the security token and temporary AccessKey pair (AccessKey ID and AccessKey secret) in the credential to create a signature. The method to create a signature for authorized access is the same as using the AccessKey pair of a root account to add a signature to the Authorization header. Take note of the following items:

- The signature key used by the user is the temporary AccessKey pair (AccessKey ID and AccessKey secret) that STS provides.
- The user needs to add the security token to the request header or in the URI as a request parameter. Only one of these two methods can be used. If both methods are selected, OSS returns InvalidArgument.
  - `x-oss-security-token:SecurityToken` is included in the request header. When CanonicalizedOSSHeaders of the signature is calculated, x-oss-security-token is taken into consideration.
  - security-token=SecurityToken is included in the URL. When CanonicalizedResource of the signature is calculated, security-token is taken into consideration as a subresource.

2.2.5. Bucket ACL

You can set the access control list (ACL) when you create a bucket, or modify the ACL for a created bucket. Only bucket owners can set or modify the ACL for buckets.

The following list describes the ACLs for buckets:

- Public read/write: Any users, including anonymous users can perform the PutObject, GetObject, and DeleteObject operations on the objects in the bucket. The bucket owner is charged for these operations. Exercise caution when you specify this ACL.
- Public read: Only the bucket owner can perform the PutObject and DeleteObject operations on the objects in the bucket. Other users, including anonymous users can perform only the GetObject operation on the objects in the bucket.
- Private: Only the bucket owner can perform the PutObject, GetObject, and DeleteObject operations on the objects in the bucket.

When you create a bucket, OSS sets the bucket ACL to private if you do not specify the bucket ACL. Only the bucket owner can call the PutBucketACL operation to modify the ACL of an existing bucket.

2.3. Common HTTP headers

This topic describes common request and response headers that are used for RESTful APIs.

Common request headers

In OSS, common HTTP request headers are used for RESTful APIs. These request headers can be used in all OSS requests. The following table lists the specific definitions of the request headers.

Common request headers

Header	Description
Authorization	The authentication information used to authenticate a request. Type: string Default value: null Scenario: non-anonymous requests

Header	Description
Content-Length	The content length of an HTTP request that is defined in RFC 2616. For more information, visit RFC 2616. Type: string Default value: null Scenario: requests that need to submit data to OSS
Content-Type	The content type of an HTTP request that is defined in RFC 2616. For more information, visit RFC 2616. Type: string Default value: null Scenario: requests that need to submit data to OSS
Date	The time in GMT stipulated in HTTP/ 1.1. Example: Wed, 05 Sep. 2012 23:00:00 GMT. Type: string Default value: null
Host	The value of the host to access. Format: `<bucketname>.regionid.example.com` . Type: string Default value: null

Common response headers

In OSS, common HTTP response headers are used for RESTful APIs. These response headers can be used in all the OSS requests. The following table lists the specific definitions of the response headers.

Common response headers

Header	Description
Content-Length	The content length of an HTTP request that is defined in RFC 2616. For more information, visit RFC 2616. Type: string Default value: null Scenario: requests that need to submit data to OSS
Connection	The connection status between the client and the OSS server. Type: enumeration Valid values: open and close Default value: null
Date	The time in GMT stipulated in HTTP/ 1.1. Example: Wed, 05 Sep. 2012 23:00:00 GMT. Type: string Default value: null
ETag	The entity tag (ETag) that is created to identify the content of the object when the object is created. If an object is created by using the PutObject request, the ETag value of the object is the MD5 hash of the object content. If an object is created by using another method, the ETag value of the content is the universally unique identifier (UUID) of the object content. The ETag value of the object can be used to check whether the object content is modified. Type: string Default value: null
Server	The server that generates a response. Type: string Default value: AliyunOSS
x-oss-request-id	The UUID that is created by the OSS server to identify the response. If you encounter any problems when you use OSS, you can contact OSS support personnel and provide this field to help them locate the problems. Type: string Default value: null

2.4. Service-relevant operations

2.4.1. GetService (ListBuckets)

You can call this operation to obtain all buckets that you own. The forward slash (/) in the request structure represents the root directory.

Usage notes

- The GetService operation is valid only for users who are authenticated.
- If no information for user authentication is provided in a request (anonymous access), OSS returns 403 Forbidden and error code AccessDenied.

- When all buckets are returned, the returned XML does not contain the Prefix, Marker, MaxKeys, IsTruncated, and NextMarker nodes. If not all results are returned, the preceding nodes are added, in which NextMarker is used to assign the marker for the subsequent query.

Request structure

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

Request parameters

When you call GetService (ListBucket), you can set the parameters described in the following table to limit the list of buckets returned so that only specified results are returned.

Parameter	Type	Required	Example	Description
prefix	String	No	example	The prefix that the names of returned buckets must contain. If this parameter is not specified, prefix information is not used to filter the returned buckets.
marker	String	No	examplebucket	The name of the object after which the list begins. If this parameter is not specified, all results are returned.
max-keys	String	No	100	The maximum number of buckets that can be returned this time. Valid values: 1 to 1000 Default value: 100

Response parameters

Parameter	Type	Example	Description
ListAllMyBucketsResult	Container	N/A	The container that stores the result of the GetService (ListBucket) request.
Prefix	String	example	The prefix that the names of returned buckets contain. This node is available only when not all buckets are returned.
Marker	String	examplebucket	The name of the bucket after which the list begins. This node is available only when not all buckets are returned.
MaxKeys	String	1	The maximum number of returned buckets in the response. This node is available only when not all buckets are returned.
IsTruncated	Enumerated string	true	Indicates whether all results are returned. Valid values: • `true`: indicates that not all of the results are returned this time. • `false`: indicates that all of the results are returned this time. This node is available only when not all buckets are returned.
NextMarker	String	xz02tphky6fjfiuc0	The marker for the next GetService (ListBucket) request to return the remaining results. This node is available only when not all buckets are returned.
Owner	Container	N/A	The container that stores the information about the bucket owner.
ID	String	17464******02745	The user ID of the bucket owner.
DisplayName	String	17464******02745	The name of the bucket owner, which is the same as the user ID.
Buckets	Container	N/A	The container that stores the information about multiple buckets.
Bucket	Container	N/A	The container that stores the bucket information.
Name	String	examplebucket	The name of the bucket.
CreateDate	Date	2011-12-01T12:27:13.000Z	The time when the bucket was created. Format: `yyyy-mm-ddThh:mm:ss.timezone`.
Location	String	oss-cn-shenzhen	The data center in which the bucket is located.
ExtranetEndpoint	String	oss-cn-shenzhen.aliyuncs.com	The public endpoint of the bucket.
IntranetEndpoint	String	oss-cn-shenzhen-internal.aliyuncs.com	The internal endpoint of the bucket.

Examples

- Sample Request 1

```
GET / HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host: regionid.example.com
Authorization: OSS n***i:C***I=
```

- Sample Response 1

```
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: 5***4
<? xml version="1.0" encoding="UTF-8"? >
<ListAllMyBucketsResult>
 <Owner>
 <ID>17464******02745</ID>
 <DisplayName>17464******02745</DisplayName>
 </Owner>
 <Buckets>
 <Bucket>
 <CreationDate>2015-12-17T18:12:43.000Z</CreationDate>
 <ExtranetEndpoint>regionid.example.com</ExtranetEndpoint>
 <IntranetEndpoint>regionid-internal.example.com</IntranetEndpoint>
 <Location>oss-cn-shanghai</Location>
 <Name>app-base-oss</Name>
 </Bucket>
 <Bucket>
 <CreationDate>2014-12-25T11:21:04.000Z</CreationDate>
 <ExtranetEndpoint>regionid.example.com</ExtranetEndpoint>
 <IntranetEndpoint>regionid-internal.example.com</IntranetEndpoint>
 <Location>regionid</Location>
 <Name>examplebucket</Name>
 </Bucket>
 </Buckets>
</ListAllMyBucketsResult>
```

- Sample Request 2

```
GET /? prefix=xz02tphky6fjfiuc&max-keys=1 HTTP/1.1
Date: Thu, 15 May 2014 11:18:32 GMT
Host:  regionid.example.com
Authorization: OSS n***i:C***I=
```

- Sample Response 2

```
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: 5***5
<? xml version="1.0" encoding="UTF-8"? >
<ListAllMyBucketsResult>
 <Prefix>xz02tphky6fjfiuc</Prefix>
 <Marker></Marker>
 <MaxKeys>1</MaxKeys>
 <IsTruncated>true</IsTruncated>
 <NextMarker>xz02tphky6fjfiuc0</NextMarker>
 <Owner>
 <ID>17464******02745</ID>
 <DisplayName>17464******02745</DisplayName>
 </Owner>
 <Buckets>
 <Bucket>
 <CreationDate>2014-05-15T11:18:32.000Z</CreationDate>
 <ExtranetEndpoint>regionid.example.com</ExtranetEndpoint>
 <IntranetEndpoint>oss-cn-hangzhou-internal.example.com</IntranetEndpoint>
 <Location>oss-cn-hangzhou</Location>
 <Name>examplebucket</Name>
 </Bucket>
 </Buckets>
</ListAllMyBucketsResult>
```

2.5. Bucket-relevant operations

2.5.1. PutBucket

You can call this operation to create a bucket. The region where the new bucket is located is consistent with the region in the endpoint that is contained in a request. After the data center of the bucket is determined, all objects in this bucket are stored in the corresponding region.

Request structure

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

Usage notes

When you call the PutBucket operation, take note of the following items:

- Anonymous access is not supported.
- You can create up to 100 buckets in the same region by using an Alibaba Cloud account.

Request headers

Header	Type	Required	Example	Description
x-oss-acl	String	No	private	The access control list (ACL) for a bucket. Default value: private. Valid values: • *public-read-write* • *public-read* • *private*

For more information about other common request headers included in PutBucket requests, see Common request headers.

Request elements

Element	Type	Required	Example	Description
StorageClass	String	No	Standard	The storage class of the bucket. Default value: Standard. Valid storage: *Standard*

Response headers

The response headers involved in the PutBucket operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT / HTTP/1.1
Host: BucketName.regionid.example.com
Date: Fri, 24 Feb 2012 03:15:40 GMT
x-oss-acl: private
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:77Dvh5wQgIjWjwO/KyRt8dOP****
<? xml version="1.0" encoding="UTF-8"? >
<CreateBucketConfiguration>
    <StorageClass>Standard</StorageClass>
</CreateBucketConfiguration>
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Fri, 24 Feb 2012 03:15:40 GMT
Location: /***
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.5.2. PutBucketACL

You can call this operation to modify the access control list (ACL) of a bucket.

Request structure

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

Usage notes

When you call the PutBucketAcl operation, take note of the following items:

- To call this operation, you must have write permissions on the bucket.
- PutBucketAcl uses the overwriting semantics. A new ACL overwrites the existing one.
- If the specified bucket for which you want to set ACL does not exist when you call this operation, a new bucket is created.

Request headers

Header	Type	Required	Example	Description
x-oss-acl	String	Yes	private	The ACL that you want to set for the bucket. This header is included in PutBucketAcl requests to set the ACL of the bucket. If this header is not included, the ACL settings do not take effect. Valid values: public-read-write, public-read, and private • public-read-write: Any users, including anonymous users can read and write objects in the bucket. Exercise caution when you set the ACL of a bucket to public-read-write. • public-read: Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket. Exercise caution when you set the ACL of a bucket to public-read. • private: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization.

For more information about other common request headers involved in the PutBucketAcl operation, see Common request headers.

Response headers

For more information about other common response headers involved in the PutBucketAcl operation, see Common response headers.

Examples

Sample requests

```
PUT /? acl HTTP/1.1
x-oss-acl: public-read
Host: BucketName.regionid.example.com
Date: Fri, 24 Feb 2012 03:21:12 GMT
Authorization: OSS q***c:K***=
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Fri, 24 Feb 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

If the specified ACL does not exist, OSS returns HTTP status code 400.

Sample error responses

```
HTTP/1.1 400 Bad Request
x-oss-request-id: 5***9
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" encoding="UTF-8"? >
<Error>
  <Code>InvalidArgument</Code>
  <Message>no such bucket access control exists</Message>
  <RequestId>5***9</RequestId>
  <HostId>***-test.example.com</HostId>
  <ArgumentName>x-oss-acl</ArgumentName>
  <ArgumentValue>error-acl</ArgumentValue>
</Error>
```

2.5.3. PutBucketLogging

Bucket owners can enable logging for their buckets. When this feature is enabled, OSS automatically records detailed information about the requests to this bucket, and follows the user-specified rules to write access logs as an object to the user-specified bucket on an hourly basis.

Notice OSS provides logging for bucket owners to understand and analyze access to buckets. OSS does not guarantee that the bucket access logs contain all access records.

Usage notes

- The source bucket to which an object belongs and the destination bucket where the processed object is stored must belong to the same user.
- To disable logging for a bucket, you need only to send a request that contains an empty BucketLoggingStatus element. For more information about the specific method, see the following sample request.
- All PutBucketLogging requests must include signatures. Anonymous access is not supported.
- When the source bucket is deleted, the corresponding logging rules are also deleted.
- OSS generates bucket access log objects on an hourly basis. However, requests in the previous hour may be recorded in the log object generated for the subsequent

hour.

- In the naming conventions for log objects generated by the OSS, UniqueString is just a UUID that OSS generates to uniquely identify the object.
- Each time OSS generates a bucket access log object, OSS performs a PUT operation and records the storage space that the object uses. However, OSS does not record the traffic generated by the PUT operation. After log objects are generated, you can perform operations on these objects as common objects.
- OSS ignores all query string parameters whose values contain the x- prefix. However, these parameters are recorded in access log objects. To identify a special request from a large number of access logs, you can add a query string parameter whose value contains the x- prefix to the URL of the request. Example
  - `http://oss-example.regionid.example.com/aliyun-logo.png`
  - `http://oss-example.regionid.example.com/aliyun-logo.png?x-user=admin`

  When OSS processes the preceding two requests, the results are the same. However, you can search for access logs by x-user=admin to locate the marked request.
- A hyphen (`-`) may be contained in any field in OSS logs. The hyphen (-) indicates that data is unknown or the field is invalid for the current request.
- More fields will be added to OSS logs. We recommend that developers consider potential compatibility issues when you develop log processing tools.

Request structure

```
PUT /? logging HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Authorization: SignatureValue
Host: BucketName.regionid.example.com
<? xml version="1.0" encoding="UTF-8"? >
<BucketLoggingStatus>
 <LoggingEnabled>
 <TargetBucket>TargetBucket</TargetBucket>
 <TargetPrefix>TargetPrefix</TargetPrefix>
 </LoggingEnabled>
</BucketLoggingStatus>
```

Request elements

Element	Type	Required	Example	Description
BucketLoggingStatus	Container	Yes	N/A	The container that stores the logging status information.
LoggingEnabled	Container	No	N/A	The container that stores the access log information. This element is required only when logging is enabled.
TargetBucket	String	This element is required when logging is enabled.	doc-log	The bucket that stores access logs.
TargetPrefix	String	No	MyLog-	The prefix of the access log object.

Response headers

Response headers involved in the PUTBucketLogging operation contain only common response headers. For more information, see Common response headers.

The name format of the object that stores access logs

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

TargetPrefix is specified by the user. YYYY-mm-DD-HH-MM-SS indicates the time when an access log object is created. YYYY indicates the year (four-digit). mm indicates the month (two-digit). DD indicates the day (two-digit). HH indicates the hour (two-digit). MM indicates the minute (two-digit). SS indicates the second (two-digit). UniqueString indicates the string generated by OSS. An example for the name of an access log object:

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

In the preceding example, MyLog- is the object name prefix specified by the user. oss-example is the name of the source bucket. 2012-09-10-04-00-00 is the object creation time (UTC+8). 0000 is the string generated by OSS.

Log object format

Element	Example	Description
Remote IP	119.140.142.11	The IP address from which a request originates. The proxy or your firewall may block this field.
Reserved	N/A	A reserved field.
Reserved	N/A	A reserved field.
Time	[02/May/2012:00:00:04 +0800]	The time when OSS received a request.
Request-URI	"GET /aliyun-logo.png HTTP/1.1"	The requested URI, including query string parameters.
HTTP Status	200	The HTTP status code that OSS returned.
SentBytes	5576	The traffic consumed when a user downloads logs from OSS.
RequestTime (ms)	71	The length of time in milliseconds used to complete the request.

Element	Example	Description
Referer	http://www.aliyun.com/product/oss	The HTTP header that identifies the address of the web page which is linked to the resource being requested.
User-Agent	curl/7.15.5	The user-agent header in an HTTP request.
HostName	oss-example.regionid.example.com	The domain name that is requested.
Request ID	505B01695037C2AF032593A4	The UUID used to uniquely identify this request.
LoggingFlag	true	Indicates whether the access logging feature is enabled.
Requester Aliyun ID	1657136103983691	The Alibaba Cloud ID of the requester. The value of this field is a hyphen (-) for anonymous access.
Operation	GetObject	The type of a request.
Bucket	oss-example	The name of the bucket that you requested.
Key	/aliyun-logo.png	The key of the object that is requested.
ObjectSize	5576	The size of an object.
Server Cost Time (ms)	17	The length of time for the OSS server to process this request. Unit: milliseconds.
Error Code	NoSuchBucket	The error code that OSS returned.
Request Length	302	The length of the request. Unit: bytes.
UserID	1657136103983691	The ID of a bucket owner.
Delta DataSize	280	The variation of the size of a bucket. The value of this field is a hyphen (-) if the bucket size does not change.
Sync Request	-	Indicates whether the request is forwarded to the origin by using Content Delivery Network (CDN). The value of this field is a hyphen (-) if this is not a CDN back-to-origin request.
Reserved	-	A reserved field.

Examples

Sample requests where bucket logging is enabled:

```
PUT /? logging HTTP/1.1
Host: ***.regionid.example.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS q***c:K***=
<? xml version="1.0" encoding="UTF-8"? >
<BucketLoggingStatus>
<LoggingEnabled>
<TargetBucket>doc-log</TargetBucket>
<TargetPrefix>MyLog-</TargetPrefix>
</LoggingEnabled>
</BucketLoggingStatus>
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

Sample requests where bucket logging is disabled:

```
PUT /? logging HTTP/1.1
Host: ***.regionid.example.com
Content-Type: application/xml
Content-Length: 86
Date: Fri, 04 May 2012 04:21:12 GMT
Authorization: OSS q***c:K***=
<? xml version="1.0" encoding="UTF-8"? >
<BucketLoggingStatus>
</BucketLoggingStatus>
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Fri, 04 May 2012 04:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.5.4. PutBucketWebsite

You can call this operation to set the static website hosting mode for a bucket.

Usage notes

- Static websites are the websites where all web pages consist of static content, such as scripts in JavaScript run on the client. OSS does not support content that needs to be processed by the server, such as PHP, JSP, and APS.NET content.
- After a bucket is set to the static website hosting mode, OSS returns the index page for anonymous access to the root domain name of the static website, and returns the result of GetBucket for signed access to the root domain name of the static website.

Request structure

```
PUT /? website HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.regionid.example.com
Authorization: SignatureValue
<? xml version="1.0" encoding="UTF-8"? >
<WebsiteConfiguration>
    <IndexDocument>
        <Suffix>index.html</Suffix>
    </IndexDocument>
    <ErrorDocument>
        <Key>errorDocument.html</Key>
    </ErrorDocument>
</WebsiteConfiguration>
```

Request parameters

- The following table describes the parameters that you can configure in the WebsiteConfiguration field in a request.

Parameter	Type	Required	Description
WebsiteConfiguration	Container	Yes	The root node. Parent nodes: none

- The following table describes the parameters that you can configure in the IndexDocument field in a request.

Element	Type	Required	Example	Description
IndexDocument	Container	Conditional. You must specify at least one of the following containers: IndexDocument, ErrorDocument, and RoutingRules.	N/A	The container for the default homepage. Parent nodes: WebsiteConfiguration
Suffix	String	Conditional. This parameter must be specified if IndexDocument is specified.	index.html	The default homepage. If this parameter is set, the default homepage is returned for access to objects whose names end with a forward slash (/). Parent nodes: IndexDocument
SupportSubDir	String	No	false	Specifies whether a request sent to access a subfolder is redirected to the default homepage of the subfolder. Valid values: ◦ A value of true indicates that the request is redirected to the default homepage of the subfolder. ◦ A value of false indicates that the request is redirected to the default homepage of the root folder instead of that of the subfolder. If the default homepage for access to bucket.oss-cn-hangzhou.aliyuncs.com/subdir/ is set to index.html. A value of false indicates that the request is redirected to bucket.oss-cn-hangzhou.aliyuncs.com/index.html, and a value of true indicates that the request is redirected to bucket.oss-cn-hangzhou.aliyuncs.com/subdir/index.html. Default value: false Parent nodes: IndexDocument

Element	Type	Required	Example	Description
Type	Enumeration	No	0	Specifies the operation to perform when the default homepage is set, the name of the accessed object does not end with a forward slash (/), and the object does not exist. **Note** This parameter takes effect only when SupportSubDir is set to true. It takes effect after RoutingRule and before ErrorFile. If the default homepage for access to bucket.oss-cn-hangzhou.aliyuncs.com/abc is set to index.html and the abc object does not exist. The following content describes the operations that correspond to the valid values of Type: ○ *0*: OSS checks whether the object named abc/index.html, which is in the `Object + Forward slash (/) + Homepage` format exists. If the object exists, OSS returns 302 and the Location header `/abc/`, which is in the `Forward slash (/) + Object + Forward slash (/)` format, in the response is URL-encoded. If the object does not exist, OSS returns 404 and continues to check ErrorFile. ○ *1*: OSS returns 404 and NoSuchKey and continues to check ErrorFile. ○ *2*: OSS checks whether abc/index.html exists. If abc/index.html exists, the content of the object is returned. If abc/index.html does not exist, OSS returns 404 and continues to check ErrorFile. Default value: 0 Parent nodes: IndexDocument

- The following table describes the parameters that you can configure in the ErrorDocument field in a request.

Parameter	Type	Required	Description
ErrorDocument	Container	Conditional. You must specify at least one of the following containers: IndexDocument, ErrorDocument, and RoutingRules.	The container used to store the default 404 page. Parent nodes: WebsiteConfiguration
Key	Container	Conditional. This parameter must be specified if ErrorDocument is specified.	The default 404 page. If this parameter is specified, the 404 page is returned when the target object does not exist. Parent nodes: ErrorDocument

- The following table describes RoutingRules, RoutingRule, and RuleNumber you can configure in a request.

Parameter	Type	Required	Example	Description
RoutingRules	Container	Conditional. You must specify at least one of the following containers: IndexDocument, ErrorDocument, and RoutingRules.	N/A	The container used to store RoutingRule. Parent nodes: WebsiteConfiguration
RoutingRule	Container	No	N/A	Specifies redirection-based back-to-origin rules or mirroring-based back-to-origin rules. You can specify a maximum of 20 rules. Parent nodes: RoutingRules
RuleNumber	String	Conditional. This parameter must be specified if RoutingRule is specified.	1	The sequence number used to match and execute redirection rules. Redirection rules are matched based on this parameter. If a match succeeds, the rule is executed and the subsequent rules are not executed. Parent nodes: RoutingRule

- The following table describes RoutingRules, RoutingRule, and Condition in a request.

Parameter	Type	Required	Example	Description
Condition	Container	Conditional. This parameter must be specified if RoutingRule is specified.	N/A	The matching conditions. If all of the specified conditions are met, the rule is executed. A rule is determined as matched only when the rule meets the conditions specified by all nodes in Condition. Parent nodes: RoutingRule

Parameter	Type	Required	Example	Description
KeyPrefixEquals	String	No	abc/	Specifies that only objects whose names contain the specified prefix match the rule. Parent nodes: Condition
HttpErrorCodeReturnedEquals	HTTP status code	No	404	Specifies that the rule is matched only when the specified object is accessed and the specified status code is returned. If the redirection rule is a mirroring-based back-to-origin rule, the parameter value must be 404. Parent nodes: Condition
IncludeHeader	Container	No	N/A	Specifies that the rule is matched only when the specified header is included in the request and the header value is equal to the specified value. You can specify up to five headers. Parent nodes: Condition
Key	String	Conditional. This parameter must be specified if IncludeHeader is specified.	host	Specifies that the rule is matched only when the specified header is included in the request and the header value is equal to the value specified by Equals. Parent nodes: IncludeHeader
Equals	String	Conditional. This parameter must be specified if IncludeHeader is specified.	test.oss-cn-beijing-internal.aliyuncs.com	Specifies that the rule is matched only when the header specified by Key is included in the request and the header value is equal to the specified value. Parent nodes: IncludeHeader
KeySuffixEquals	String	No	zip	Specifies that only objects whose names contain the specified suffix match the rule. The default value is null, which indicates that no suffix is specified. Parent nodes: Condition

- The following table describes RoutingRules, RoutingRule, and Redirect you can configure in a request.

Parameter	Type	Required	Example	Description
Redirect	Container	Conditional. This parameter must be specified if RoutingRule is specified.	N/A	The operation to perform after the rule is matched. Parent nodes: RoutingRule
RedirectType	String	Conditional. This parameter must be specified if Redirect is specified.	Mirror	The redirection type. Valid values: ◦ Mirror: mirroring-based back-to-origin ◦ External: external redirection. OSS returns the 3xx HTTP redirect code and the Location header for your to redirect the request to another IP address. ◦ Internal: internal redirection. OSS redirects the access from Object1 to Object2 based on the rule. In this case, the user accesses Object2 instead of Object1. ◦ AliCDN: redirection based on Alibaba Cloud CDN. OSS adds an additional header to the request, which is different from the External type. After CDN identifies the header, CDN redirects the access to the specified IP address and returns the obtained data instead of the redirect request to the user. Parent nodes: Redirect
PassQueryString	Boolean	No	true	Specifies whether parameters in the original request are contained in the redirect request when redirection-based or mirroring-based back-to-origin is performed. For example, if the PassQueryString parameter is set to true and the `? a=b&c=d` parameter is contained in the request sent to OSS, this parameter is added to the Location header in the redirect request when the HTTP redirect code is 302. For example, if the request is `Location:www.test.com? a=b&c=d` and the redirecting type is mirroring-based back-to-origin, the ?a=b&c=d parameter is also contained in the back-to-origin request. Valid values: true and false Default value: false Parent nodes: Redirect

Parameter	Type	Required	Example	Description
MirrorURL	String	Conditional. This parameter must be specified if RedirectType is set to Mirror.	http://www.example.com/	Specifies the IP address of the origin in mirroring-based back-to-origin. This parameter takes effect only when the value of RedirectType is Mirror. URLs that start with http:// or https:// must end with a forward slash (/). OSS adds the object name to the string to form the back-to-origin URL. For example, the object to access is myobject. If the specified URL is `http://www.example.com/`, the back-to-origin URL is `http://www.test.com/myobject`. If the specified URL is `http://www.test.com/dir1/`, the back-to-origin URL is `http://www.test.com/dir1/myobject`. Parent nodes: Redirect
MirrorPassQueryString	Boolean	No	true	This parameter plays the same role as PassQueryString and has a higher priority than PassQueryString. This parameter takes effect only when the value of RedirectType is Mirror. Default value: false Parent nodes: Redirect
MirrorFollowRedirect	Boolean	No	true	Specifies whether the access is redirected to the specified Location if the origin returns a 3XX HTTP status code and when the origin receives a mirroring-based back-to-origin request. For example, when a mirroring-based back-to-origin request is initiated, the origin returns 302, and Location is specified. ◦ In this case, if the value of MirrorFollowRedirect is true, OSS continues to send requests to the IP address specified by Location. A request can be redirected for a maximum of 10 times. If a request is redirected for more than 10 times, a mirroring-based back-to-origin failure is returned. ◦ If the value of MirrorFollowRedirect is false, OSS returns 302 and passes through the Location. This parameter takes effect only when the value of RedirectType is Mirror. Default value: true Parent node: Redirect
MirrorCheckMd5	Boolean	No	false	Specifies whether OSS checks the MD5 hash of the body of the response returned by the origin. When the value of this parameter is true and the response returned by the origin includes the Content-Md5 header, OSS checks whether the MD5 hash of the obtained data matches the header value. If the MD5 hash of the obtained data does not match the header value, OSS does not store the data. This parameter takes effect only when the value of RedirectType is Mirror. Default value: false Parent nodes: Redirect
MirrorHeaders	Container	No	N/A	Specifies the headers contained in the response that is returned when you use mirroring-based back-to-origin. This parameter takes effect only when the value of RedirectType is Mirror. Parent nodes: Redirect
PassAll	Boolean	No	true	Specifies whether OSS passes through all request headers ◦ except for reserved headers and headers that start with oss-, x-oss-, and x-drs-, such as content-length, authorization2, authorization, range, and date ◦ to the origin. This parameter takes effect only when the value of RedirectType is Mirror. Default value: false Parent nodes: MirrorHeaders
Pass	String	No	myheader-key1	Specifies the headers to pass through to the origin. The header can be up to 1,024 bytes in length and can contain only letters, digits, and hyphens (-). This parameter takes effect only when the value of RedirectType is Mirror. You can specify up to 10 headers. Parent nodes: MirrorHeaders

Parameter	Type	Required	Example	Description
Remove	String	No	myheader-key3	Specifies the headers that are not allowed to be passed through to the origin. The header can be up to 1,024 bytes in length. The character set of this parameter is the same as that of Pass. This parameter takes effect only when the value of RedirectType is Mirror. You can specify up to 10 headers. This parameter is used together with PassAll. Parent nodes: MirrorHeaders
Set	Container	No	N/A	Specifies the headers that are sent to the origin. The specified headers are configured in the data returned by the origin regardless of whether the headers are contained in the request. You can specify up to 10 header sets. This parameter takes effect only when the value of RedirectType is Mirror. Parent nodes: MirrorHeaders
Key	String	Conditional. This parameter must be specified if Set is specified.	myheader-key5	Specifies the key of the header. The key can be up to 1,024 bytes in length. The character set of this parameter is the same as that of Pass. This parameter takes effect only when the value of RedirectType is Mirror. Parent nodes: Set
Value	String	Conditional. This parameter must be specified if Set is specified.	myheader-value5	Specifies the value of the header. The value can be up to 1,024 bytes in length and cannot contain `\r\n`. This parameter takes effect only when the value of RedirectType is Mirror. Parent nodes: Set
Protocol	String	Conditional. This parameter must be specified if the value of RedirectType is External or AliCDN.	http	The protocol used for redirection. For example, if you access the object named test, Protocol is set to https, and Hostname is set to `www.example.com`, the Location header is `https://www.example.com/test`. This parameter takes effect only when the value of RedirectType is External or AliCDN. Valid values: *http* and *https*. Parent nodes: Redirect
HostName	String	Conditional. This parameter must be specified if the value of RedirectType is External or AliCDN.	www.example.com	The domain name used for redirection, which must comply with the naming conventions for domain names. For example, if you access the object named test, Protocol is set to https, and Hostname is set to `www.example.com`, the Location header is `https://www.example.com/test`. Parent nodes: Redirect
ReplaceKeyPrefixWith	String	No	abc/	Specifies the string that is used to replace the prefix of the object name in the redirect request. If the prefix of the object is empty, this string is added before the object name. **Note** You can specify only one of the ReplaceKeyWith and ReplaceKeyPrefixWith nodes in a rule. If KeyPrefixEquals is set to abc/ and ReplaceKeyPrefixWith is set to def/, when you access the abc/test.txt object, the Location header is `http://www.example.com/def/test.txt`. Parent nodes: Redirect
EnableReplacePrefix	Boolean	No	true	If this parameter is set to true, the prefix of object names is replaced with the value specified by ReplaceKeyPrefixWith. If this parameter is not specified or empty, the prefix of object names is truncated. **Note** When the ReplaceKeyWith parameter is not empty, this parameter cannot be set to true. Default value: false Parent nodes: Redirect

Parameter	Type	Required	Example	Description
ReplaceKeyWith	String	No	prefix/${key}.suffix	Specifies the string that is used to replace the requested object name when the request is redirected. This parameter can be a variable. The ${key} variable that indicates the object name in the request is supported. If ReplaceKeyWith is set to `prefix/${key}.suffix`, when you access the object named test, the Location header is `http://www.example.com/prefix/test.suffix`. Parent nodes: Redirect
HttpRedirectCode	HTTP status code	Conditional. This parameter must be specified if the value of RedirectType is External or AliCDN.	301	The HTTP redirect code in the response. This parameter takes effect only when the value of RedirectType is External or AliCDN. Valid values: 301, 302, and 307 Parent nodes: Redirect

For more information about other common request elements of PutBucketWebsite, see Common request headers.

Response elements

For more information about the response elements of PutBucketWebsite, see Common response headers.

Examples

- Sample requests

```
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 qn6qrrqx******k53otfjbyc:KU5h8YM******0dXqf3JxrTZHiA=
<? xml version="1.0" encoding="UTF-8"? >
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>index.html</Suffix>
      <SupportSubDir>true</SupportSubDir>
      <Type>0</Type>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
</WebsiteConfiguration>
```

- Sample responses

```
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
```

- Complete sample code

```
PUT /? website HTTP/1.1
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 2064
Host: test.oss-cn-hangzhou-internal.aliyuncs.com
Authorization: OSS a1nBN******QMf8u:sNKIHT6ci/z231yIT5vYnetD****
User-Agent: aliyun-sdk-python-test/0.4.0
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>index.html</Suffix>
    <SupportSubDir>true</SupportSubDir>
    <Type>0</Type>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
      <RuleNumber>1</RuleNumber>
      <Condition>
        <KeyPrefixEquals>abc/</KeyPrefixEquals>
        <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
      </Condition>
      <Redirect>
        <RedirectType>Mirror</RedirectType>
        <PassQueryString>true</PassQueryString>
        <MirrorURL>http://www.test.com/</MirrorURL>
        <MirrorPassQueryString>true</MirrorPassQueryString>
        <MirrorFollowRedirect>true</MirrorFollowRedirect>
        <MirrorCheckMd5>false</MirrorCheckMd5>
        <MirrorHeaders>
          <PassAll>true</PassAll>
          <Pass>myheader-key1</Pass>
          <Pass>myheader-key2</Pass>
          <Remove>myheader-key3</Remove>
          <Remove>myheader-key4</Remove>
          <Set>
            <Key>myheader-key5</Key>
            <Value>myheader-value5</Value>
          </Set>
        </MirrorHeaders>
      </Redirect>
    </RoutingRule>
    <RoutingRule>
      <RuleNumber>2</RuleNumber>
      <Condition>
        <KeyPrefixEquals>abc/</KeyPrefixEquals>
        <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
        <IncludeHeader>
          <Key>host</Key>
          <Equals>test.oss-cn-beijing-internal.aliyuncs.com</Equals>
        </IncludeHeader>
      </Condition>
      <Redirect>
        <RedirectType>AliCDN</RedirectType>
        <Protocol>http</Protocol>
        <HostName>www.test.com</HostName>
        <PassQueryString>false</PassQueryString>
        <ReplaceKeyWith>prefix/${key}.suffix</ReplaceKeyWith>
        <HttpRedirectCode>301</HttpRedirectCode>
      </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Fri, 27 Jul 2018 09:03:18 GMT
Content-Length: 0
Connection: keep-alive
x-oss-request-id: 5B5ADFD6ED3CC49176CBE29D
x-oss-server-time: 47
```

2.5.5. PutBucketReferer

You can call this operation to configure a Referer whitelist and specify whether to allow a request whose Refer field is empty.

Usage notes

PutBucketReferer replaces the existing configured whitelist with the whitelist specified by RefererList. When empty RefererList that contains no Referer request elements is uploaded, this operation overwrites the configured whitelist. The existing RefererList is deleted.

Request structure

```
PUT /? referer HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.oss.example.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>
```

Request elements

Element	Type	Required	Example	Description
RefererConfiguration	Container	Yes	N/A	The container that contains the Referer configurations. Type: container Child nodes: AllowEmptyReferer and RefererList Parent node: none
AllowEmptyReferer	Boolean	Yes	true	Specifies whether to allow a request whose Refer field is empty. Default value: true
RefererList	Container	Yes	N/A	The container that contains the Referer whitelist.
Referer	String	No	http://www.aliyun.com	The Referer whitelist.

Response headers

The response headers involved in the PutBucketReferer operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests that exclude the Referer element

```
PUT /? referer HTTP/1.1
Host: BucketName.oss.example.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS q***c:K***=
<? xml version="1.0" encoding="UTF-8"? >
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer >
< RefererList />
</RefererConfiguration>
```

Sample requests that include the Referer element

```
PUT /? referer HTTP/1.1
Host: BucketName.oss.example.com
Content-Length: 247
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS q***c:K***=
<? 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>
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.5.6. PutBucketLifecycle

Configures lifecycle for a bucket. After a lifecycle rule is enabled, Object Storage Service (OSS) automatically deletes objects that match the rule or converts the storage class for the objects based on the rule on a regular basis.

Usage notes

When you call the PutBucketLifecycle operation, take note of the following items:

- Only the bucket owner or RAM users that have the PutBucketLifecycle permission can initiate a PutBucketLifecycle request.
- If no lifecycle rules are configured for a bucket, a lifecycle rule is created after you call PutBucketLifecycle. If a lifecycle rule is configured for the bucket, a new lifecycle rule is created and the existing lifecycle rule is overwritten.
- PutBucketLifecycle uses overwriting semantics. New lifecycle rules overwrite the existing rules. To add lifecycle rules for a bucket, call GetBucketLifecycle to obtain the existing lifecycle configurations of the bucket, add new lifecycle configurations, and call PutBucketLifecycle to update the lifecycle configurations for the bucket.
- When you call the PutBucketLifecycle operation, you can set a validity period or expiration date for objects or parts that are uploaded in incomplete multipart upload tasks.

Request syntax

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

Request headers

A PutBucketLifecycle request contains only common request headers. For more information, see Common request headers.

Request elements

Element	Type	Required	Example	Description
CreatedBeforeDate	String	Either Days or CreatedBeforeDate required	2002-10-11T00:00:00.000Z	The date based on which the lifecycle rules are implemented. OSS performs the specified operation on data whose last modified date is before this date. The time follows the ISO 8601 standard and be at 00:00:00 UTC. For example, 2002-10-11T00:00:00.000Z indicates that objects that are last updated before 2002-10-11T00: 00: 00.000Z are deleted or the storage classes of these objects are converted to another storage class. Objects that are last updated at or after this time are not deleted or their storage classes are not converted to another storage class.
Days	Positive integer	Either Days or CreatedBeforeDate required	1	The number of days from when the objects were last modified to when the lifecycle rule takes effect.
Expiration	Container	No	N/A	The delete operation to perform on objects based on the lifecycle rule.
AbortMultipartUpload	Container	No	N/A	The operation to perform on incomplete multipart upload tasks.
ID	String	No	rule1	The unique ID of a lifecycle rule. The ID of a lifecycle rule can be up to 255 bytes in length. If you do not specify a unique ID, OSS automatically generates a unique ID for the lifecycle rule.
LifecycleConfiguration	Container	Yes	N/A	The container that stores lifecycle configurations. The container can contain up to 1,000 lifecycle rules.
Prefix	String	Yes	data	The prefix in the names of the objects to which the rule applies. The prefixes specified by different rules cannot overlap. • If Prefix is specified, this rule applies only to objects whose names contain the specified prefix in the bucket. • If Prefix is not specified, this rule applies to all objects in the bucket.
Rule	Container	Yes	N/A	Identifies the rule.
Status	String	Yes	Enabled	Specifies whether to run the rule. A value of Enabled indicates that OSS runs the rule. A value of Disabled indicates that OSS ignores the rule.

Response headers

The response to a PutBucketLifecycle request contains only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT /?lifecycle HTTP/1.1
Host: BucketName.oss.example.com
Content-Length: 443
Date: Mon, 14 Apr 2014 01:08:38 GMT
Authorization: OSS q***c:K***=
<?xml version="1.0" encoding="UTF-8"?>
</LifecycleConfiguration>
 <Rule>
 <ID>rule1</ID>
 <Prefix>logs/</Prefix>
 <Status>Enabled</Status>
 <Expiration>
 <Days>1</Days>
 </Expiration>
 <AbortMultipartUpload>
 <Days>1</Days>
 </AbortMultipartUpload>
 </Rule>
 <Rule>
 <ID>rule2</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>
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
Date: Mon, 14 Apr 2014 01:17:10 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.5.7. GetBucket (ListObjects)

You can call this operation to list the information about all objects in a bucket.

Detail analysis

- User metadata of objects is not returned for the GetBucket request.
- If listing cannot be complete at a time because of the max-keys setting, `<NextMarker>` is added to the returned result, which indicates that the subsequent listing can start from this marker. The value of NextMarker is still in the list result.
- During a conditional query, even if the marker does not exist in the list, items after the marker are returned in alphabetical order. If the value of max-keys is smaller than 0 or greater than 1000, OSS returns 400 Bad Request and error code InvalidArgument.
- The prefix and marker parameters are used to list the returned objects by page. The parameter value must be smaller than 1,024 bytes in length.
- If prefix is set to a folder name in the request, the objects whose names contain this prefix are listed, including all objects and subfolders in the folder. If prefix is set to a folder name and delimiter is set to a forward slash (/) in the request, the names of the objects and subfolders in the folder are listed in the CommonPrefixes element. Objects and subfolders are not listed. For example, a bucket contains the following objects: fun/test.jpg, fun/movie/001.avi, and fun/movie/007.avi. If prefix is set to fun/, the three objects are returned. If prefix is set to fun/ and delimiter is set to a forward slash (/), fun/test.jpg and fun/movie/are returned.

Request structure

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

Request parameters

Parameter	Type	Required	Example	Description
delimiter	String	No	/	The character used to group objects by name. If you specify the Delimiter parameter in the request, the response contains the CommonPrefixes element. Objects whose names contain the same string from the prefix to the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes. Default value: null

Parameter	Type	Required	Example	Description
marker	String	No	test1.txt	The name of the object after which the list begins. If this parameter is specified, objects whose names are alphabetically greater than the marker parameter value are returned. The marker parameter is used to list the returned objects by page, and this parameter value must be smaller than 1,024 bytes in length. Even if the specified marker does not exist in the list during a conditional query, the list starts from the object whose name is alphabetically greater than the marker parameter value. Default value: null
max-keys	String	No	200	The maximum number of objects to return. If the listing cannot be complete at a time because max-keys is specified, the NextMarker element is included in the response as the marker for next listing. Valid values: 1 to 1000 Default value: 100
prefix	String	No	fun	The prefix that the returned object names must contain. • The value of prefix must be smaller than 1,024 bytes in length. • If you specify a prefix to query objects, the names of the returned objects still contain the prefix. If prefix is set to a folder name in the request, the objects whose names contain this prefix are listed, including all objects and subfolders in the folder. If prefix is specified and delimiter is set to a forward slash (/), only the objects in the folder are listed. The subfolders are grouped together as a single result in CommonPrefixes. Objects and folders in the subfolders are not listed. For example, a bucket contains the following objects: fun/test.jpg, fun/movie/001.avi, and fun/movie/007.avi. If prefix is set to fun/, the three objects are returned. If prefix is set to fun/ and delimiter is set to a forward slash (/), fun/test.jpg and fun/movie/ are returned. Default value: null
encoding-type	String	No	URL	The encoding type of the object name in the response. Default value: null Valid value: URL **Notice** The delimiter, marker, prefix, NextMarker, and Key values are UTF-8 encoded. If delimiter, marker, prefix, NextMarker, and Key contain control characters that are not supported by the XML 1.0 standard, you can specify encoding-type to encode Delimiter, Marker, Prefix, NextMarker, and Key in the response.

Response elements

Element	Type	Example	Description
Contents	Container	N/A	The container that stores the returned object metadata. Parent nodes: ListBucketResult
CommonPrefixes	String	N/A	If the Delimiter parameter is specified in the request, the response contains the CommonPrefixes element. Objects whose names contain the same string from the prefix to the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes. Parent nodes: ListBucketResult
Delimiter	String	/	The character used to group objects by name. If you specify the Delimiter parameter in the request, the response contains the CommonPrefixes element. Objects whose names contain the same string from the prefix to the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes. Parent nodes: ListBucketResult
EncodingType	String	N/A	The encoding type of the object name in the response. In addition, you can use the encoding-type parameter to encode the following elements in the response: Delimiter, Marker, Prefix, NextMarker, and Key. Parent nodes: ListBucketResult
DisplayName	String	user_example	The name of the object owner. Parent nodes: ListBucketResult, Contents, and Owner

Element	Type	Example	Description
ETag	String	5B3C1A2E053D763E1B002CC607C5A0FE1****	The entity tag (ETag). An ETag is created when the object is created to identify the content of an object. • If an object is created using a PutObject request, the ETag value is the MD5 hash of the object content. • If an object is created using other methods, the ETag value is the UUID of the object content. • The ETag value of an object can be used to check whether the object content is changed. However, we recommend that you do not use the ETag of an object as the MD5 hash of the object to verify data integrity. Parent nodes: ListBucketResult and Contents
ID	String	0022012****	The user ID of the bucket owner. Parent nodes: ListBucketResult, Contents, and Owner
IsTruncated	Enumerated string	false	Indicates whether the returned results are truncated. Valid values: *true* and *false* • true indicates that not all of the results are returned this time. • false indicates that all of the results are returned this time. Parent nodes: ListBucketResult
Key	String	fun/test.jpg	The key of the object. Parent nodes: ListBucketResult and Contents
LastModified	Date	2012-02-24T08:42:32.000Z	The last modified time of the object. Parent nodes: ListBucketResult and Contents
ListBucketResult	Container	N/A	The container that stores the result of the GetBucket request. Child nodes: Name, Prefix, Marker, MaxKeys, Delimiter, IsTruncated, NextMarker, and Contents Parent node: none
Marker	String	test1.txt	The name of the object after which the listing begins. Parent nodes: ListBucketResult
MaxKeys	String	100	The maximum number of returned objects in the response. Parent nodes: ListBucketResult
Name	String	oss-example	The name of the bucket. Parent nodes: ListBucketResult
Owner	Container	N/A	The container that stores the information about the bucket owner. Child nodes: DisplayName and ID Parent nodes: ListBucketResult
Prefix	String	fun/	The prefix that the names of returned objects contain. Parent nodes: ListBucketResult
Size	String	344606	The size of the returned object. Unit: byte. Parent nodes: ListBucketResult and Contents
StorageClass	String	Standard	The storage class of the object. Parent nodes: ListBucketResult and Contents

For more information about other common response headers involved in the GetBucket operation, such as `x-oss-request-id` and `Content-Type`, see Common response headers.

Examples

- Sample requests for the simple GetBucket operation

```
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/ZghT7cGa0ykbo****
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
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>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user-example</DisplayName>
      </Owner>
</Contents>
<Contents>
      <Key>fun/movie/007.avi</Key>
      <LastModified>2012-02-24T08:43:27.000Z</LastModified>
      <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user-example</DisplayName>
      </Owner>
</Contents>
<Contents>
      <Key>fun/test.jpg</Key>
      <LastModified>2012-02-24T08:42:32.000Z</LastModified>
      <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user-example</DisplayName>
      </Owner>
</Contents>
<Contents>
      <Key>oss.jpg</Key>
      <LastModified>2012-02-24T06:07:48.000Z</LastModified>
      <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user-example</DisplayName>
      </Owner>
</Contents>
</ListBucketResult>
```

- Sample requests that have the prefix parameter specified

```
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/ZghT7cGa0ykbo****
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
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>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user_example</DisplayName>
      </Owner>
</Contents>
<Contents>
      <Key>fun/movie/007.avi</Key>
      <LastModified>2012-02-24T08:43:27.000Z</LastModified>
      <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user_example</DisplayName>
      </Owner>
</Contents>
<Contents>
      <Key>fun/test.jpg</Key>
      <LastModified>2012-02-24T08:42:32.000Z</LastModified>
      <ETag>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user_example</DisplayName>
      </Owner>
</Contents>
</ListBucketResult>
```

- Sample requests that have the prefix and delimiter parameters specified

```
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:DNrnx7xHk3sgysx7I8U9I9IY****
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906****
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>"5B3C1A2E053D763E1B002CC607C5A0FE1****"</ETag>
      <Type>Normal</Type>
      <Size>344606</Size>
      <StorageClass>Standard</StorageClass>
      <Owner>
          <ID>0022012****</ID>
          <DisplayName>user_example</DisplayName>
      </Owner>
</Contents>
<CommonPrefixes>
      <Prefix>fun/movie/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

- Sample requests that have the marker parameter specified

  In this example, the value of max-keys is set to 2, which indicates that the maximum number of objects to return is two.

```
GET /? max-keys=2&marker=test1.txt HTTP/1.1
Host: test-bucket-xxx.oss-cn-shenzhen.aliyuncs.com
Accept-Encoding: identity
Accept: */*
Connection: keep-alive
User-Agent: aliyun-sdk-python/2.11.0(Darwin/18.2.0/x86_64;3.4.1)
date: Tue, 26 May 2020 08:39:48 GMT
authorization: OSS LTAIBlVW9xxxxxxx:MmY11jLlO8UlAqjqHK3Ckp****
```

  Sample responses

  The value of NextMarker in the response is the marker for next listing.

```
HTTP/1.1 200 OK
Server: AliyunOSS
Date: Tue, 26 May 2020 08:39:48 GMT
Content-Type: application/xml
Content-Length: 1032
Connection: keep-alive
x-oss-request-id: 5ECCD5D4881816373582xxx
x-oss-server-time: 3
<? xml version="1.0" encoding="UTF-8"? >
<ListBucketResult>
  <Name>test-bucket-xxx</Name>
  <Prefix></Prefix>
  <Marker>test1.txt</Marker>
  <MaxKeys>2</MaxKeys>
  <Delimiter></Delimiter>
  <EncodingType>url</EncodingType>
  <IsTruncated>true</IsTruncated>
  <NextMarker>test100.txt</NextMarker>
  <Contents>
    <Key>test10.txt</Key>
    <LastModified>2020-05-26T07:50:18.000Z</LastModified>
    <ETag>"C4CA4238A0B923820DCC509A6F75****"</ETag>
    <Type>Normal</Type>
    <Size>1</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>1305433xxx</ID>
      <DisplayName>1305433xxx</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>test100.txt</Key>
    <LastModified>2020-05-26T07:50:20.000Z</LastModified>
    <ETag>"C4CA4238A0B923820DCC509A6F75****"</ETag>
    <Type>Normal</Type>
    <Size>1</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>1305433xxx</ID>
      <DisplayName>1305433xxx</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

2.5.8. GetBucketAcl

You can call this operation to obtain the access control list (ACL) of a bucket.

Request structure

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

Request headers

For more information about the common request headers involved in the GetBucketACL operation, see Common request headers.

Response elements

Element	Type	Example	Description
AccessControlPolicy	Container	N/A	The container that contains the result of the GetBucketACL request.
Owner	Container	N/A	The container that stores the information about the bucket owner.
ID	String	0022012****	The user ID of the bucket owner.
DisplayName	String	0022012****	The name of the bucket owner, which is the same as the user ID.
AccessControlList	Container	N/A	The container that contains the ACL information.
Grant	Enumerated string	public-read	The ACL for the bucket. Valid values: private, public-read, and public-read-write

Examples

Sample requests

```
GET /? acl HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 04:11:23 GMT
Authorization: OSS q***c:C***=
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***B
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>0022012****</ID>
        <DisplayName>0022012****</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>public-read</Grant>
    </AccessControlList>
</AccessControlPolicy>
```

2.5.9. GetBucketLocation

You can call this operation to view the location information about the data center to which a bucket belongs.

Request structure

```
GET /? location HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

For more information about the common request headers involved in the GetBucketLocation operation, see Common request headers.

Response elements

Element	Type	Example	Description
LocationConstraint	String	oss-cn-hangzhou	The region to which the bucket belongs.

Examples

Sample requests

```
Get /? location HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 04 May 2012 05:31:04 GMT
Authorization: OSS q***c:c***=
```

Sample responses

```
HTTP/1.1 200
x-oss-request-id: 5***B
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 >
```

2.5.10. GetBucketInfo

You can call this operation to view the information about a bucket.

You can view the following information when you call GetBucketInfo:

- The time when the bucket was created
- The public endpoint
- The internal endpoint
- The bucket owner information
- The bucket access control list (ACL)

Request structure

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

Request headers

For more information about the common request headers involved in the GetBucketInfo operation, see Common request headers.

Response elements

Element	Type	Example	Description
BucketInfo	Container	N/A	The container that stores the bucket information content.
Bucket	Container		The container that stores the specific information about the bucket.
CreationDate	Date	2013-07-31T10:56:21.000Z	The time when the bucket was created.
ExtranetEndpoint	String	oss-cn-hangzhou.aliyuncs.com	The public endpoint.
IntranetEndpoint	String	oss-cn-hangzhou-internal.aliyuncs.com	The internal endpoint.
Location	String	cn-hangzhou	The region of the data center to which the bucket belongs.
Name	String	oss-example	The name of the bucket.
Owner	Container	N/A	The container that stores the information about the bucket owner.
ID	String	2***3	The user ID of the bucket owner.
DisplayName	String	2***3	The name of the bucket owner, which is the same as the user ID.
AccessControlList	Container	N/A	The container that contains the ACL information.
Grant	Enumeration	private	The ACL for the bucket. Valid values: private, public-read, and public-read-write

Examples

Sample requests

```
Get /? bucketInfo HTTP/1.1
Host: BucketName.oss.example.com
Date: Sat, 12 Sep 2015 07:51:28 GMT
Authorization: OSS q***c: B***=
```

Sample success responses when the information about a bucket is obtained

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>cn-hangzhou</Location>
    <Name>oss-example</Name>
    <Owner>
      <DisplayName>2***3</DisplayName>
      <ID>2***3</ID>
    </Owner>
    <AccessControlList>
      <Grant>private</Grant>
    </AccessControlList>
  </Bucket>
</BucketInfo>
```

Sample responses when the specified bucket is not found

```
HTTP/1.1 404
x-oss-request-id: 5***B
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>5***4</RequestId>
  <HostId>nosuchbucket.oss.example.com</HostId>
  <BucketName>nosuchbucket</BucketName>
</Error>
```

Sample responses when you are not authorized to access information about a bucket

```
HTTP/1.1 403
x-oss-request-id: 5***C
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>5***E</RequestId>
  <HostId>BucketName.oss.example.com</HostId>
</Error>
```

2.5.11. GetBucketLogging

You can call this operation to view the logging configurations of a bucket.

Request structure

```
GET /? logging HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

For more information about the common request headers involved in the GetBucketLogging operation, see Common request headers.

Response elements

Element	Type	Example	Description
BucketLoggingStatus	Container	N/A	The container that stores the logging status information.
LoggingEnabled	Container	N/A	The container that stores the access log information. This element is required only when logging is enabled.
TargetBucket	String	mybucketlogs	The bucket that stores access logs.
TargetPrefix	String	mybucket-access_log/	The prefix of the access log object.

Examples

Sample requests

```
Get /? logging HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 04 May 2012 05:31:04 GMT
Authorization: OSS q***c:c***=
```

Sample responses for buckets that have logging rules configured

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>
```

Sample responses for buckets that have no logging rules configured

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>
```

2.5.12. GetBucketWebsite

You can call this operation to view the static website hosting status of a bucket.

Request structure

```
GET /? website HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

For more information about the common request headers involved in the GetBucketWebsite operation, see Common request headers.

Response elements

Element	Type	Example	Description
ErrorDocument	Container	N/A	The parent element of the Key child element.
IndexDocument	Container	N/A	The parent element of the Suffix child element.
Key	String	error.html	The file name used when the 404 HTTP status code is returned.
Suffix	String	index.html	The index file name added when a folder URL is returned. This element cannot be empty or contain a slash (/). For example, if the index file is set to index.html, regionid.example.com/mybucket/mydir/ included in a request is converted into regionid.example.com/mybucket/index.html by default.
WebsiteConfiguration	Container	N/A	The container for the request.

Examples

Sample requests

```
Get /? website HTTP/1.1
Host: BucketName.regionid.example.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS q***c:B***=
```

Sample success responses

```
HTTP/1.1 200
x-oss-request-id: 5***B
Date: Thu, 13 Sep 2012 07:51:28 GMT
Connection: keep-alive
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>
```

Sample error responses

```
HTTP/1.1 404
x-oss-request-id: 5***B
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>5***F</RequestId>
    <HostId>BucketName.regionid.example.com</HostId>
</Error>
```

2.5.13. GetBucketReferer

You can call this operation to view the Referer configurations of a bucket.

Request structure

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

Request headers

For more information about the common request headers involved in the GetBucketReferer operation, see Common request headers.

Response elements

Element	Type	Example	Description
RefererConfiguration	Container	N/A	The container that contains the Referer configurations.
AllowEmptyReferer	Enumerated string	true	Specifies whether to allow a request whose Refer field is empty. Valid values: true and false Default value: true
RefererList	Container	N/A	The container that contains the Referer whitelist.
Referer	String	http://www.aliyun.com	The Referer whitelist.

Examples

Sample requests

```
Get /? referer HTTP/1.1
Host: BucketName.oss.example.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS q***: B***=
```

Sample responses to requests that include the Referer element

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>
```

Sample responses to requests that exclude the Referer element

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>
```

2.5.14. GetBucketLifecycle

You can call this operation to view the lifecycle configurations of a bucket.

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

Request headers

For more information about the common request headers involved in the GetBucketLifecycle operation, see Common request headers.

Response elements

Element	Type	Example	Description
CreatedBeforeDate	String	2002-10-11T00:00:00.000Z	The date based on which to run lifecycle rules. OSS performs the specified operation on data whose last modified date is before this date. Specify the time in the ISO 8601 standard. The time must be at UTC 00:00:00. For example, 2002-10-11T00:00:00.000Z indicates that objects last updated before 2002-10-11T00: 00: 00.000Z are deleted or the storage classes of these objects are converted to another storage class. Objects last updated at or after this time are not deleted or their storage classes are not converted to another storage class.
Days	Positive integer	1	The number of days within which objects can be retained after the objects are last modified.
Expiration	Container	N/A	The delete operation on the objects based on the lifecycle rule.
AbortMultipartUpload	Container	N/A	The delete operation on the multipart upload tasks that are incomplete.
ID	String	rule1	The unique ID of a lifecycle rule. The ID can be a maximum of 255 bytes in length. If the value of this parameter is null or not specified, OSS generates a unique ID for the lifecycle rule.
LifecycleConfiguration	Container	N/A	The container that stores lifecycle configurations. The container can contain up to 1,000 rules.
Prefix	String	logs/	The prefix of names of objects to which the rule applies. • If Prefix is specified, this rule applies only to objects whose names contain the specified prefix in the bucket. • If Prefix is not specified, this rule applies to all objects in the bucket.
Rule	Container	N/A	Identifies the rule.
Status	String	Enabled	Specifies whether to run the rule. A value of Enabled indicates that OSS runs the rule. A value of Disabled indicates that OSS ignores the rule.

Examples

Sample requests

```
Get /? lifecycle HTTP/1.1
Host: BucketName.oss.example.com
Date: Mon, 14 Apr 2014 01:17:29 GMT
Authorization: OSS q***:c***=
```

Sample responses for buckets that have lifecycle rules configured

```
HTTP/1.1 200
x-oss-request-id: 5***B
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>rule1</ID>
 <Prefix>logs/</Prefix>
 <Status>Enabled</Status>
 <Expiration>
 <Days>1</Days>
 </Expiration>
 <AbortMultipartUpload>
 <Days>1</Days>
 </AbortMultipartUpload>
 </Rule>
 <Rule>
 <ID>rule2</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>
```

Sample responses for buckets that have no lifecycle rules configured

```
HTTP/1.1 404
x-oss-request-id: 5***B
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>5***9</RequestId>
  <HostId> BucketName.oss.example.com</HostId>
</Error>
```

2.5.15. GetBucketTags

Obtains the tags for a bucket.

Request syntax

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

Response elements

Element	Type	Description
Tagging	Container	Indicates the container used to store the returned tags for a bucket. Parent node: None
TagSet	Container	Indicates the container used to store the returned tags for a bucket. Parent node: Tagging

Element	Type	Description
Tag	Container	Indicates the container used to store the returned tags for a bucket. Parent node: TagSet
Key	String	Indicates the key of a tag. Parent node: Tag
Value	String	Indicates the value of a tag. Parent node: Tag

Detail analysis

- If the target bucket does not exist, the 404 No Content error is returned with the error code: NoSuchBucket.
- Only the bucket owner and authorized RAM users can view the tags for a bucket. Otherwise, the 403 Forbidden error is returned with the error code: AccessDenied.
- If no tags are added for the bucket, OSS returns an XML message body in which value of Tagging is null.

Examples

- Request example:

```
GET /?tagging
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 20 Dec 2018 13:09:13 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:ceOEyZavKY4QcjoUWYSpYbJ3****
```

- Response example:

```
200 (OK)
content-length: 237
server: AliyunOSS
x-oss-request-id: 5C1B2D24B90AD5490CFE368E
date: Thu, 20 Dec 2018 13:12:21 GMT
content-type: application/xml
<?xml version="1.0" encoding="UTF-8"?>
<Tagging>
  <TagSet>
    <Tag>
      <Key>testa</Key>
      <Value>value1-test</Value>
    </Tag>
    <Tag>
      <Key>testb</Key>
      <Value>value2-test</Value>
    </Tag>
  </TagSet>
</Tagging>
```

2.5.16. DeleteBucket

You can call this operation to delete a bucket.

Request structure

```
DELETE / HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Usage notes

- To prevent unexpected data loss, OSS does not allow users to delete non-empty buckets.
- If you delete a non-empty bucket, OSS returns 409 Conflict and error code BucketNotEmpty.
- Only the bucket owner can delete the bucket. If you delete a bucket when you are not authorized, OSS returns 403 Forbidden and error code AccessDenied.

Request headers

The request headers involved in the DeleteBucket operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the DeleteBucket operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
DELETE / HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 05:31:04 GMT
Authorization: OSS q***:c***=
```

Sample responses

```
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
```

2.5.17. DeleteBucketLogging

You can call this operation to disable logging for a bucket.

Request structure

```
DELETE /? logging HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

The request headers involved in the DeleteBucketLogging operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the DeleteBucketLogging operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
DELETE /? logging HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 05:35:24 GMT
Authorization: OSS q***c:6***=
```

Sample responses

```
HTTP/1.1 204 No Content
x-oss-request-id: 5***B
Date: Fri, 24 Feb 2012 05:35:24 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

2.5.18. DeleteBucketWebsite

You can call this operation to disable static website hosting for a bucket.

Request structure

```
DELETE /? website HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

Request headers involved in the DeleteBucketWebsite operation contain only common request headers. For more information, see Common request headers.

Response headers

Response headers involved in the DeleteBucketWebsite operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
DELETE /? website HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS q***:L***=
```

Sample responses

```
HTTP/1.1 204 No Content
x-oss-request-id: 5***B
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

2.5.19. DeleteBucketTagging

You can call DeleteBucketTagging to delete all tags associated with a bucket.

Request syntax

```
DELETE /? tagging HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Detail analysis

1. If the bucket does not exist, OSS returns the 404 Not Found message. Error code: NoSuchBucket.
2. If the bucket is not associated with any tags, OSS returns the 204 No Content message.

Examples

Sample request:

```
DELETE /? tagging HTTP/1.1
Host: ***.regionid.example.com
Date: Thu, 23 Jun 2015 15:39:18 GMT
Authorization: OSS q***c:C***=
```

Sample response:

```
HTTP/1.1 204 No Content
x-oss-request-id: 5***B
Date: Thu, 23 Jun 2015 15:39:18 GMT
Connection: close
Content-Length: 0
Server: AliyunOSS
```

2.5.20. DeleteBucketTags

Deletes the tags added for a bucket.

Request syntax

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

Detail analysis

- If the target bucket does not exist, the 404 No Content error is returned with the error code: NoSuchBucket.
- Only the bucket owner can delete the tags added for a bucket. If you try to delete the tags for a bucket owned by another user, the 403 Forbidden error is returned with the error code: AccessDenied.
- If no tags are added for the bucket or the key of the specified tag does not exist, the HTTP status code 204 is returned.

Examples

- Request example 1 (Delete all tags for a bucket):

  ```
  DELETE /?tagging HTTP/1.1
  Host: oss-example.oss-cn-hangzhou.aliyuncs.com
  Date: Tue, 25 Dec 2018 17:35:24 GMT
  Authorization: OSS qn6qrrqxo2oawuk53otf****:6ZVHOehYzxoC1yxRydPQs/Cn****
  ```

  Response example:

  ```
  HTTP/1.1 204 No Content
  x-oss-request-id: 5C22E0EFD127F6810B1A92A8
  Date: Tue, 25 Dec 2018 17:35:24 GMT
  Connection: keep-alive
  Content-Length: 0
  Server: AliyunOSS
  ```

- Request example 2 (Delete specified tags for a bucket, for example, tags of which the keys are k1 and k2):

```
DELETE /?tagging=k1,k2 HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
Date: Tue, 25 Dec 2018 17:35:24 GMT
Authorization: OSS qn6qrrqxo2oawuk53otf****:6ZVHOehYzxoC1yxRydPQs/Cn****
```

Response example:

```
HTTP/1.1 204 No Content
x-oss-request-id: 5C22E0EFD127F6810B1A92A8
Date: Tue, 25 Dec 2018 17:35:24 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

2.6. Object-relevant operations

2.6.1. PutObject

Uploads objects.

Request syntax

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

Usage notes

When you call the PutObject operation, take note of the following items:

- The object that you want to upload cannot be larger than 5 GB in size.
- By default, if you upload an object that has the same name as that of an accessible existing object, the existing object is overwritten by the uploaded object and 200 OK is returned.
- OSS does not use folders. All elements are stored as objects. You can create a simulated folder by creating an object 0 KB in size whose name ends with a forward slash (/).
- To ensure data integrity, Object Storage Service (OSS) provides multiple methods to check the MD5 hash of the data that you want to upload. To perform MD5 verification based on the Content-MD5 header, add the header to the request. When the Content-MD5 header value is specified in a request, OSS calculates the MD5 hash of the sent data and compares the calculated MD5 hash with the Content-MD5 value specified in the request. If the two values do not match, the operation fails and 400 (Bad Request) is returned.

Request headers

Header	Type	Required	Example	Description
Cache-Control	String	No	no-cache	The caching behavior of the web page when the object is downloaded. For more information, visit RFC 2616.
Content-Disposition	String	No	attachment;filename=oss_download.jpg	The name of the object when the object is downloaded. For more information, visit RFC 2616.
Content-Encoding	String	No	utf-8	The content encoding format of the object when the object is downloaded. For more information, visit RFC 2616. Type: string. Default value: null
Content-MD5	String	No	uk8Os/oShU2MMhcajpeAVA==	The Content-MD5 value of the message content (excluding headers). The value is obtained by calculating the 128-bit hash value and encoding the value in Base64 based on RFC 1864. This request header can be used to check the message validity. The message content is valid if the received message content is the same as the content that is sent. Although this request header is optional, we recommend that you use it for end-to-end checks.
Expires	String	No	Fri, 28 Feb 2012 05:38:42 GMT	The expiration time. For more information, visit RFC 2616. Note OSS does not limit or verify this value.
x-oss-server-side-encryption	String	No	KMS	The method that is used to encrypt the object on the OSS server when the object is created. Valid values: *AES256* and *KMS*或*SM4* If you specify this parameter, this parameter is returned in the response header and the uploaded object is encrypted and stored. When you download the object, the x-oss-server-side-encryption header is included in the response and the header value is set to the algorithm that is used to encrypt the object.

Header	Type	Required	Example	Description
x-oss-server-side-data-encryption	String	No	SM4	The algorithm that is used to encrypt the object that you want to upload. If this header is not specified, the object is encrypted by using AES-256. This parameter is valid only when x-oss-server-side-encryption is set to KMS. Valid values: SM4
x-oss-server-side-encryption-key-id	String	No	3dea0cae-5fd6-4bf8-8574-87c557******	The ID of the customer master key (CMK) that is managed by Key Management Service (KMS). This parameter is valid only when x-oss-server-side-encryption is set to KMS.
x-oss-object-acl	String	No	private	The access control list (ACL) of the object. Valid values: public-read, private, and public-read-write

Response headers

The response to a PutObject request contains only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT /oss.jpg HTTP/1.1
Host: ***.regionid.example.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 q***:k***=
[344606 bytes of object data]
```

Sample responses

```
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: 5***A
x-oss-bucket-version: 1***9
ETag: "A***2"
```

2.6.2. GetObject

You can call this operation to query an object. To call this operation, you must have read permissions on the object.

Detail analysis

- You can specify the Range parameter in the GetObject request to implement resumable data transfer. We recommend that you use this parameter when you download large objects.
- If the Range parameter is specified in the request, the response contains the length of the entire object and the range of bytes returned for this operation. For example, Content-Range: bytes 0-9/44 indicates that the length of the entire object is 44 bytes and the range of bytes returned in response to the request is from byte 0 to byte 9. If the value of Range is invalid, the entire object is returned and the response returned by OSS does not include Content-Range.
- Parameters in the GetObject request cannot be customized for OSS to return corresponding headers in the response if the access is anonymous.
- If this object is encrypted by using a server-side encryption algorithm based on entropy encoding, OSS automatically returns the decrypted object and x-oss-server-side-encryption in the response when OSS receives the GetObject request. The value of x-oss-server-side-encryption indicates the server-side encryption algorithm of the object.
- To obtain an object compressed in the GZIP format, add Accept-Encoding:gzip to the display mode in the request. OSS determines whether to return data compressed in the GZIP format based on the Content-Type value and size of the object. If the object is compressed in the GZIP format, the response returned by OSS does not include the ETag value of the object. OSS supports GZIP for objects whose Content-Type header is set to one of the following values: HTML, JavaScript, CSS, XML, RSS, and JSON. The object must be at least 1 KB in size.

Request syntax

```
GET /ObjectName HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
Range: bytes=ByteRange (optional)
```

Request parameters

OSS allows you to customize parameters in the GET request for OSS to return corresponding headers in the response if the GET request includes a signature. The following table describes the headers that you can specify to return in the response to a GET request.

Header	Type	Required	Example	Description
response-content-type	String	No	text	The content-type header in the response.
response-content-language	String	No	Chinese	The content-language header in the response.
response-expires	String	No	Fri, 24 Feb 2012 17:00:00 GMT	The expires header in the response.
response-cache-control	String	No	No-cache	The cache-control header in the response. This parameter is empty by default.
response-content-disposition	String	No	attachment; filename:testing.txt	The content-disposition header in the response. This parameter is empty by default.
response-content-encoding	String	No	utf-8	The content-encoding header in the response. This parameter is empty by default.

Request headers

Header	Type	Required	Example	Description
Range	String	No	bytes=100-900	The range of data of the object that you want to obtain. For example, if you specify bytes=0-9, OSS transfers byte 0 to byte 9, a total of 10 bytes.
If-Modified-Since	String	No	Fri, 13 Nov 2015 14:47:53 GMT	The object transfer condition. If the specified time is earlier than the actual modified time of the object, the system transfers the object and returns 200 OK. Otherwise, the system returns 304 Not Modified. The time must be in GMT.
If-Unmodified-Since	String	No	Fri, 13 Nov 2015 14:47:53 GMT	The object transfer condition. If the specified time is later than or equal to the actual modified time of the object, OSS transfers the object and returns 200 OK. Otherwise, OSS returns 412 Precondition Failed. The time must be in GMT.
If-Match	String	No	5B3C1A2E0563E1B002CC607C****	The object transfer condition. If the specified ETag value of the object matches the actual ETag value, OSS transfers the object and returns 200 OK. Otherwise, OSS returns 412 Precondition Failed.
If-None-Match	String	No	5B3C1A2E0563E1B002CC607C****	The object transfer condition. If the specified ETag value does not match the actual ETag value, OSS transfers the object and returns 200 OK. Otherwise, OSS returns 304 Not Modified. Type: string. This parameter is empty by default.

Response headers

If the requested object is a symbolic link, the content of the object is returned. Response headers `Content-Length`, `ETag`, and `Content-Md5` indicate the metadata of the requested object. The `Last-Modified` header indicates the later one of the time points when the requested object and the symbolic link are last modified. All other headers indicate the metadata of the symbolic link.

Header	Type	Example	Description
x-oss-server-side-encryption	String	x-oss-server-side-encryption	If the requested object is encrypted by using a server-side encryption algorithm based on entropy encoding, OSS automatically decrypts the object and returns the decrypted object after OSS receives the GetObject request. OSS includes x-oss-server-side-encryption in the response to indicate the encryption algorithm that is used to encrypt the object on the server.

Examples

Common sample request

```
GET /oss.jpg HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 06:38:30 GMT
Authorization: OSS q***:U***=
```

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 3***1
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: "5***E "
Content-Type: image/jpg
Content-Length: 344606
Server: AliyunOSS
[344606 bytes of object data]
```

Sample request that includes the Range parameter

```
GET //oss.jpg HTTP/1.1
Host:***. regionid.example.com
Date: Fri, 28 Feb 2012 05:38:42 GMT
Range: bytes=100-900
Authorization: OSS q***:q***=
```

Sample response

```
HTTP/1.1 206 Partial Content
x-oss-request-id: 2***9
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: "5***E "
Accept-Ranges: bytes
Content-Range: bytes 100-900/344606
Content-Type: image/jpg
Content-Length: 801
Server: AliyunOSS
[801 bytes of object data]
```

Sample request that includes custom headers in the response

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

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
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: "5***E "
Content-Length: 344606
Connection: keep-alive
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]
```

2.6.3. CopyObject

You can call this operation to copy objects within a bucket or between different buckets in the same region.

To create a copy of an existing object in Object Storage Service (OSS), you can iniate a PUT request to OSS, and add x-oss-copy-source to the PUT request to specify the source object you want to copy. OSS determines that this request is a copy operation and performs the copy operation on the server side. If the copy operation is successful, OSS returns the information of the destination object to you. You can call this operation to copy objects smaller than or equal to 1 GB. To copy an object larger than 1 GB, you must use UploadPartCopy. For more information, see UploadPartCopy.

Usage notes

- If the source object URL is the same as the destination object URL, the metadata of the source object is replaced regardless of the value of x-oss-metadata-directive.
- Request headers for the CopyObject operation start with x-oss- and must be added to a string-to-sign.
- If the x-oss-server-side-encryption header is specified in the CopyObject request and the value of the parameter is valid, the destination object is encrypted on the server after the CopyObject operation is performed regardless of whether the source object is encrypted on the server. In addition, the CopyObject response header contains the x-oss-server-side-encryption header, the value of which is set to the encryption algorithm of the destination object. When this destination object is downloaded, the response also contains the x-oss-server-side-encryption header, the value of which is set to the encryption algorithm of this destination object. If the x-oss-server-side-encryption request header is not specified for the CopyObject operation, the destination object is the data that is not encrypted on the server regardless of the source object is encrypted on the server.
- Objects created by using the AppendObject operation cannot be copied.

Request syntax

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

Request headers

Header	Type	Required	Example	Description
x-oss-copy-source	String	Yes	/oss-example/oss.jpg	The path of the source object. This parameter is empty by default.
x-oss-copy-source-if-match	String	No	5B3C1A2E053D763E1B002CC607C5****	The object copy condition. If the ETag value of the source object is the same as the ETag value specified in the request, OSS copies the object and returns 200 OK. Otherwise, OSS returns 412 Precondition Failed, which indicates that the preprocessing has failed. This parameter is empty by default.
x-oss-copy-source-if-none-match	String	No	5B3C1A2E053D763E1B002CC607C5****	The object copy condition. If the ETag value of the source object is different from the ETag value specified in the request, OSS copies the object and returns 200 OK. Otherwise, OSS returns 304 Not Modified, which indicates that the preprocessing has failed. This parameter is empty by default.
x-oss-copy-source-if-unmodified-since	String	No	2019-04-09T07:01:56.000Z	The object copy condition. If the specified time is the same as or later than the modified time of the object, OSS copies the object and returns 200 OK. Otherwise, OSS returns 412 Precondition Failed, which indicates that the preprocessing has failed. This parameter is empty by default.
x-oss-copy-source-if-modified-since	String	No	2019-04-09T07:01:56.000Z	The object copy condition. If the source object is modified after the specified time, OSS copies the object. Otherwise, OSS returns 304 Not Modified, which indicates that the preprocessing has failed. This parameter is empty by default.
x-oss-metadata-directive	String	No	COPY	The method used to configure the metadata of the destination object. Valid values: • *COPY*: The metadata of the source object is copied to the destination object. This is the default value. The x-oss-server-side-encryption attribute of the source object is not copied to the destination object. The x-oss-server-side-encryption parameter in the CopyObject request specifies the method used to encrypt the destination object. • *REPLACE*: The metadata specified in the request instead of the metadata of the source object is used as the metadata of the destination object.
x-oss-server-side-encryption	String	No	AES256	The entropy coding-based encryption algorithm that OSS uses to encrypt an object when you create the object. Valid values: *AES256* and *KMS*. • If the x-oss-server-side-encryption header is not specified in the CopyObject request, the destination object is not encrypted on the server regardless of whether the source object is encrypted on the server. • If the x-oss-server-side-encryption header is specified in the CopyObject request, the destination object is encrypted on the server after the CopyObject operation is performed regardless of whether the source object is encrypted on the server. In addition, the response to a CopyObject request contains the x-oss-server-side-encryption header, the value of which is set to the algorithm used to encrypt the destination object. When the destination object is downloaded, the x-oss-server-side-encryption header is included in the response header and its value is set to the algorithm used to encrypt the object.
x-oss-server-side-encryption-key-id	String	No	9468da86-3509-4f8d-a61e-6eab1eac****	The ID of the customer master key (CMK) managed by KMS. This parameter is valid only when x-oss-server-side-encryption is set to KMS.
x-oss-object-acl	String	No	private	The ACL of the destination object when the object is created. Valid values: *public-read*, *private*, *public-read-write*, and *default*.

For more information about common headers in the CopyObject operation, such as Host and Date, see Common request headers.

Response headers

For more information about common response headers in the response to the CopyObject operation, such as Content-Length and Connection, see Common response headers.

Response elements

Element	Type	Example	Description
CopyObjectResult	Container	N/A	The result of the CopyObject operation. This parameter is empty by default.
ETag	String	5B3C1A2E053D763E1B002CC607C5****	The ETag value of the destination object. Parent node: CopyObjectResult
LastModified	String	Fri, 24 Feb 2012 07:18:48 GMT	The time when the destination object was last modified. Parent node: CopyObjectResult

Examples

Sample request

```
PUT /copy_oss.jpg HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 07:18:48 GMT
x-oss-copy-source: /oss-example/oss.jpg
Authorization: OSS q***:g***=
```

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
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>"5***E"</ETag>
</CopyObjectResult>
```

2.6.4. AppendObject

You can call this operation to upload an object by appending the object to an existing object. Objects created by using the AppendObject operation are append objects, whereas objects uploaded by using the PutObject operation are normal objects.

Usage notes

- You can perform AppendObject only for an append object. For example, if a normal object that shares the same name already exists and AppendObject is called, OSS returns HTTP status code 409 and error code ObjectNotAppendable.
- If you perform a PutObject operation on an existing append object, the append object is overwritten by a new normal object.
- After HeadObject is performed, OSS returns x-oss-object-type to indicate the type of the object. If HeadObject is performed on an append object, the object type is Appendable. If HeadObject is performed on an append object, OSS also returns x-oss-next-append-position and x-oss-hash-crc64ecma.
- In the response in XML to the GetBucket (ListObjects) request, the type of an append object is set to Appendable.
- You can neither use CopyObject to copy an append object nor change the server-side encryption attribute of this object. You can use CopyObject to modify the user metadata of this object.
- The append and position parameters belong to CanonicalizedResource that must be included in signatures.

Request structure

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

Request headers

Header	Type	Required	Example	Description
append	String	Yes	N/A	The request that is sent to perform an AppendObject operation. Each time an AppendObject operation is performed, the last modified time of this object is updated.

Header	Type	Required	Example	Description
position	String	Yes	0	The position from which the AppendObject operation starts. Each time an AppendObject operation succeeds, x-oss-next-append-position in the response header specifies the position of the next AppendObject request. The value of position in the first AppendObject operation must be 0. The value of position in the subsequent operation is the current object length. For example, if the value of position specified in the first AppendObject request is 0, and the value of content-length is 65536, the value of position specified in the second AppendObject request must be set to 65536. • If the value of position is 0 and an object with the same name does not exist, you can set headers, such as x-oss-server-side-encryption in the AppendObject request in the same way as you set in a PutObject request. If you add a correct x-oss-server-side-encryption header in an AppendObject request in which the value of position is 0, the x-oss-server-side-encryption header is also included in the response header. You can initiate a CopyObject request to modify the metadata of the object in subsequent operations. • If the position value is correct and the content with a length of 0 is appended to an existing append object, the status of the object does not change.
Cache-control	String	No	no-cache	The web page caching behavior for the object. For more information, see RFC 2616. Default value: null
Content-Disposition	String	No	attachment;filename=oss_download.jpg	The name of the object when the object is downloaded. For more information, see RFC 2616. Default value: null
Content-Encoding	String	No	utf-8	The object encoding format of the object. For more information, see RFC 2616. Default value: null
Content-MD5	String	No	ohhnqLBJFiKkPSBO1eNaUA==	The Content-MD5 header value is a string calculated by using the MD5 algorithm. This header is used to check whether the content of the received message is the same as that of the sent message. To obtain the value of the Content-MD5 header, calculate a 128-bit byte array based on the message content, rather than the header, and then encode the byte array in Base64. Default value: null Limits: none
Expires	GMT	No	Wed, 08 Jul 2015 16:57:01 GMT	The time period after which the response is considered expired. For more information, see RFC 2616. Default value: null
x-oss-server-side-encryption	String	No	AES256	The method used to encrypt objects on the OSS server. Valid values: • *AES256*: uses keys managed by OSS for encryption and decryption (SSE-OSS). • *KMS*: uses CMKs managed by KMS for encryption and decryption. • *SM4*: uses the SM4 block cipher algorithm for encryption and decryption.
x-oss-object-acl	String	No	private	The ACL for the object. Valid values: • *public-read*: Only the bucket owner can perform write operations on objects in the bucket. Other users, including anonymous users, can perform only read operations on the objects in the bucket. • *private*: Only the bucket owner or authorized users can read from and write to the objects in the bucket. Other users, including anonymous users, cannot access objects in the bucket. • *public-read-write*: All users, including anonymous users, can perform read and write operations on objects in the bucket. Fees incurred by these operations are paid by the bucket owner. Use this ACL policy only when necessary.
x-oss-meta-*	String	No	x-oss-meta-location	You can add parameters prefixed with x-oss-meta-* when you create an append object by calling the AppendObject operation. However, you cannot add these parameters when you append objects to an existing append object. Parameters prefixed with x-oss-meta-* are considered metadata. You can configure multiple parameters prefixed with x-oss-meta-* for an object. However, the total size of metadata cannot exceed 8 KB. The name of parameters prefixed with x-oss-meta-* can contain hyphens (-), digits, and letters. Uppercase letters are converted to lowercase letters. Other characters such as underscores (_) are not supported.

For more information about the common request headers included in AppendObject requests, see Common request headers.

Response headers

Header	Type	Example	Description
x-oss-next-append-position	64-bit integer	1717	The position that must be provided in the next request, which is the current object length. This header is returned when a successful message is returned for an AppendObject request, or when the 409 HTTP status code is returned because the position and the object length do not match.
x-oss-hash-crc64ecma	64-bit integer	3231342946509354535	The 64-bit CRC value of the object. This value is calculated based on the ECMA-182 standard.

For more information about other common response headers involved in the AppendObject operation, see Common response headers.

Method for calculating the 64-bit CRC value

The CRC-64 value of an append object is calculated based on the ECMA-182 standard. You can calculate the CRC-64 by using the following methods:

- Use the Boost CRC module

```
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();
}
```

- Use the crcmod function in Python

```
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=0xffffffffffffffffL, rev=True)
print do_crc64("123456789")
```

Examples

Sample requests

```
POST /oss.jpg? append&position=0 HTTP/1.1
Host: BucketName.oss.example.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 q***:k***=
[1717 bytes of object data]
```

Sample responses

```
HTTP/1.1 200 OK
Date: Wed, 08 Jul 2015 06:57:01 GMT
ETag: "0***0"
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
x-oss-hash-crc64ecma: 14741617095266562575
x-oss-next-append-position: 1717
x-oss-request-id: 5***1
```

2.6.5. DeleteObject

You can call this operation to delete an object.

Usage notes

When you call the DeleteObject operation, take note of the following items:

- You must have write permissions on the object that you want to delete.
- HTTP status code 204 is returned when the DeleteObject operation succeeds, regardless of whether the object exists.
- If the object you want to delete is a symbolic link, the DeleteObject operation deletes only the symbolic link but not the content that the link points to.

Request structure

```
DELETE /ObjectName HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

The request headers involved in the DeleteObject operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the DeleteObject operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
DELETE /copy_oss.jpg HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 07:45:28 GMT
Authorization: OSS q***:z***=
```

Sample responses

```
HTTP/1.1 204 NoContent
x-oss-request-id: 5***1
Date: Fri, 24 Feb 2012 07:45:28 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.6.6. DeleteMultipleObjects

You can call this operation to delete multiple objects in the same bucket by using one HTTP request.

A maximum of 1,000 objects can be deleted when you call the DeleteMultipleObjects operation. This operation supports the following return modes:

- Verbose mode: The message body returned by OSS contains the result of each deleted object.
- Quiet mode: The message body returned by OSS contains only the results for objects that fail to be deleted. If all objects are deleted, no message body is returned.

Detail analysis

- You must specify Content-Length and Content-MD5 for the DeleteMultipleObjects request. OSS verifies that the received message body is correct based on the two header values before the delete operation is performed.
- The Content-MD5 value of the message content (excluding headers) is obtained by calculating the 128-bit hash value and then encoding the value in Base64 based on RFC 1864.
- If DeleteMultipleObjects is used to delete an object that does not exist, the operation is considered to be successful.
- The message body for DeleteMultipleObjects can be up to 2 MB. If the size of the message body exceeds 2 MB, OSS returns the MalformedXML error code.
- A maximum of 1,000 objects can be deleted when you call the DeleteMultipleObjects operation. If more than 1,000 objects are requested, OSS returns the MalformedXML error code.
- If the Content-MD5 request header is specified, OSS calculates the MD5 hash of the received content and compares the calculated MD5 hash with the Content-MD5 header value in the request. If the two MD5 hashes are different, OSS returns InvalidDigest.

Request structure

```
POST /? delete HTTP/1.1
Host: BucketName.regionid.example.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>
```

Request headers

When you call the DeleteMultipleObjects operation, you can use the encoding-type parameter to encode the object names in the response.

Header	Type	Required	Example	Description
Encoding-type	String	No	url	The Key parameter is UTF-8 encoded. If the Key parameter includes control characters that are not supported by the XML 1.0 standard, you can specify this header to encode the Key parameter value in the response. Valid values: *url*
Content-Length	String	Yes	151	The length of the HTTP message body. OSS verifies the received message body based on this header, and deletes the object only when the length of the message body is the same as this header.

Header	Type	Required	Example	Description
Content-MD5	String	Yes	lVaXz0QX0UTCXraY9Z61mQ==	The Content-MD5 header value is a string calculated by using the MD5 algorithm. This header is used to check whether the content of the received message is same as that of the sent message. After the Content-MD5 request header is uploaded, OSS calculates the MD5 hash of the received object and checks whether the calculated MD5 hash is the same as the MD5 hash provided in the request. **Note** To obtain the value of the Content-MD5 header, encrypt the message body of the DeleteMultipleObjects request by using the MD5 algorithm to obtain a 128-bit byte array, and then encode the byte array in Base64.

Request elements

Element	Type	Required	Example	Description
Delete	Container	Yes	N/A	The container that stores the DeleteMultipleObjects request.
Object	Container	Yes	N/A	The container that stores the information about the object.
Key	String	Yes	test.jpg	The name of the object to be deleted.
Quiet	Boolean	Yes	true	Enables the Quiet response mode. DeleteMultipleObjects provides the following two response modes: • Quiet: The message body of the response returned by OSS includes only objects that fail to be deleted. If all objects are deleted, the response does not include a message body. • Verbose: The message body of the response returned by OSS includes the results of all deleted objects. By default, this mode is used. Valid values: *true* (enables the Quiet mode) and *false* (enables the Verbose mode) Default value: *false*

Response headers

Response headers involved in the DeleteMultipleObjects operation contain only common response headers. For more information, see Common response headers.

Examples

Sample Request 1

```
POST /? delete HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 29 Feb 2012 12:26:16 GMT
Content-Length:151
Content-MD5: o***==
Authorization: OSS q***:+z***=
<? 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>
```

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 7***7
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>
```

Sample Request 2

```
POST /? delete HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length:151
Content-MD5: o***A==
Authorization: OSS q***c:W***=
<? 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>
```

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
Date: Wed, 29 Feb 2012 12:33:45 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.6.7. HeadObject

You can call this operation to return only the metadata of a specified object without returning the object content.

Detail analysis

- After the HeadObject request is sent, no message body is returned, regardless of whether OSS returns the 200 OK message or an error message.
- If you specify user metadata prefixed with x-oss-meta- when you send a PutObject request, such as x-oss-meta-location, the user metadata is returned.
- If the object on which you want to perform the HeadObject operation does not exist, the system returns 404 Not Found.
- If server-side encryption is configured for the object, OSS returns the x-oss-server-side-encryption header in the response. The value of the x-oss-server-side-encryption header indicates the server-side encryption algorithm of the object.

Request syntax

```
HEAD /ObjectName HTTP/1.1
Host: BucketName/regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

Header	Type	Required	Example	Description
If-Modified-Since	String	No	Fri, 24 Feb 2012 07:32:52 GMT	The information return condition. If the specified time is earlier than the last modified time of the object, OSS returns the metadata of the object and 200 OK. Otherwise, OSS returns 304 Not Modified. This parameter is empty by default.

Header	Type	Required	Example	Description
If-Unmodified-Since	String	No	Fri, 24 Feb 2012 07:32:52 GMT	The information return condition. If the specified time is equal to or later than the last modified time of the object, OSS returns the metadata of the object and 200 OK. Otherwise, OSS returns 412 Precondition Failed. This parameter is empty by default.
If-Match	String	No	215697CF4417D144C25EB698F5******	The information return condition. If the specified ETag value matches the ETag value of the object, OSS returns the metadata of the object and 200 OK. Otherwise, OSS returns 412 Precondition Failed. This parameter is empty by default.
If-None-Match	String	No	215697CF4417D144C25EB698F5******	The information return condition. If the specified ETag value does not match the ETag value of the object, OSS returns the metadata of the object and 200 OK. Otherwise, OSS returns 304 Not Modified. This parameter is empty by default.

Response headers

If the requested object is a symbolic link, take of the following headers included in the response:

- The Content-Length, ETag, and Content-Md5 headers are the metadata of the object to which the symbolic link points.
- The returned value of Last-Modified uses the last modified time of the symbolic link or the last modified time of the object to which the symbolic link points, whichever one is later.
- Other response headers are the metadata of the symbolic link.

Header	Type	Example	Description
x-oss-meta-*	String	x-oss-meta-test	The user metadata that is prefixed with x-oss-meta-. If you specify headers prefixed with x-oss-meta- in the PutObject request, the user metadata of the object is included in the response.
Custom headers that are not prefixed with x-oss-meta-	String	x-oss-persistent-headers	If you specify headers that are not prefixed with x-oss-meta in the PutObject request to configure the user metadata of the object, the headers are included in the response. Example: x-oss-persistent-headers: *key1:base64_encode(value1),key2:base64_encode(value2).....*
x-oss-server-side-encryption	String	KMS	This header is included in the response if the object is encrypted by using the server-side encryption algorithm based on entropy encoding. The value of this header is the algorithm used to encrypt the object on the server side.
x-oss-server-side-encryption-key-id	String	3dea0cae-5fd6-4bf8-8574-87c557******	This header is included in the response if the object is encrypted by using KMS at the server side. The value of this header is the CMK ID used to encrypt the object.
x-oss-object-type	String	Normal	The type of the object. Valid values: • Normal: The object is uploaded by using PutObject. • Appendable: The object is uploaded by using AppendObject. • Multipart: The object is uploaded by using MultipartUpload.
x-oss-next-append-position	String	100	The position for the next append operation. If the type of the object is Appendable, this header is included in the response.
x-oss-hash-crc64ecma	String	154294152181602747 58	The 64-bit CRC value of the object. This value is calculated based on the ECMA-182 standard. If an object is created before OSS supports CRC-64, this header may not be included in the response when you call HeadObject to query the metadata of the object.
Content-Md5	String	lVaXz0QX0UTCXraY9Z61mQ==	• This header is included in the response if the object type is Normal. The value of this header is calculated based on the following process: 1. Calculate the MD5 hash of the message content based on RFC 1864 to obtain a 128-bit value. Do not include headers to calculate the MD5 hash. 2. Encode the value by using Base64. • If the object type is Multipart or Appendable, this header is not included in the response.
Last-Modified	String	Fri, 24 Feb 2012 06:07:48 GMT	The time when the object was last modified. The time is in GMT specified in HTTP/1.1.

Examples

Sample request

```
HEAD /oss.jpg HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 07:32:52 GMT
Authorization: OSS q***:J***=
```

Sample response

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
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: "f***8"
Content-Length: 344606
Content-Type: image/jpg
Connection: keep-alive
Server: AliyunOSS
```

2.6.8. GetObjectMeta

You can call this operation to query the basic metadata of an object in a bucket. The basic metadata includes the ETag, Size (object size), and LastModified. The object content is not returned.

Usage notes

- After the GetObjectMeta request is sent, no message body is contained in the response.
- You must include the Object meta parameter in the request when you call the GetObjectMeta operation. Otherwise, the request is used for GetObject.
- If the object does not exist, OSS returns 404 Not Found.
- GetObjectMeta is more lightweight than HeadObject. Only some basic metadata of an object in a bucket is returned for GetObjectMeta. The basic metadata includes the ETag, Size (object size), and LastModified. The value of the Content-Length response header indicates the object size.

Request structure

```
GET /ObjectName? objectMeta HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

Request headers involved in the GetObjectMeta operation contain only common request headers. For more information, see Common request headers.

Response headers

Header	Type	Example	Description
Content-Length	String	344606	The size of the object.
ETag	String	215697CF4417D144C25EB698F59EB599	The ETag that is created to identify the content of the object when the object is created. If an object created by using a PutObject request, the ETag of the object is the MD5 hash of the object content. If an object is created by using other methods, the ETag value is the UUID of the object content. The ETag value of the object can be used to check whether the object content is modified. However, we recommend that you do not use the ETag of an object as the MD5 hash of the object to check data integrity. Default value: null
Last-Modified	String	Fri, 24 Feb 2012 06:07:48 GMT	The last time when the object is modified. The time is in GMT specified in HTTP/1.1.

Examples

Sample requests

```
GET /oss.jpg? objectMeta HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS q***c:C***=
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
Date: Wed, 29 Apr 2015 05:21:12 GMT
ETag: "5***E"
Last-Modified: Fri, 24 Feb 2012 06:07:48 GMT
Content-Length: 344606
Connection: keep-alive
Server: AliyunOSS
```

2.6.9. PutObjectACL

You can call this operation to modify the ACL of an object.

When you call PutObjectACL, you can set the x-oss-object-acl header in the PUT request to configure ACL for an object. Only bucket owners have permissions to configure object ACL. If the operation succeeds, 200 is returned. Otherwise, the corresponding error code and message are returned. The following object ACLs are available: private, public-read, public-read-write, and default.

Usage notes

- You can call PutObjectACL to configure ACL for an object. In addition, when you write an object, you can also include the x-oss-object-acl header in the request to configure ACL for the object. For example, you can include the x-oss-object-acl header in the request to configure the ACL for an object when you call PutObject.
- The object ACL takes precedence over its bucket ACL. For example, the bucket ACL is private, but the object ACL is public read/write. When the object is accessed, the object ACL takes precedence over the bucket ACL. If an object has no ACL configured, its bucket ACL applies.

Request structure

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

Request headers

Header	Type	Required	Example	Description
x-oss-object-acl	String	Yes	private	The ACL of the object when the object was created. Valid values: • private: Only the owner or authorized users of the bucket can read and write the object. • public-read: Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. • public-read-write: All users, including anonymous users, can read and write the object. • default: The ACL of an object is the same as that of the bucket in which the object is stored.

For more information about the other common request headers involved in the PutObjectACL operation, see Common request headers.

Response headers

The response headers involved in the PutObjectACL operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT /test-object? acl HTTP/1.1
x-oss-object-acl: public-read
Host: ***.regionid.example.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS q***c:K***=
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.6.10. GetObjectACL

You can call this operation to query the ACL of an object.

Request structure

```
GET /ObjectName? acl HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

The request headers involved in the GetObjectACL operation contain only common request headers. For more information, see Common request headers.

Response elements

Element	Type	Example	Description
AccessControlList	Container	N/A	The container that contains the ACL information.
AccessControlPolicy	Container	N/A	The container used to store results of the GetObjectACL request.
DisplayName	String	1746495857******	The name of the bucket owner, which is the same as the user ID.

Element	Type	Example	Description
Grant	Enumerated string	private	The ACL of the object. Valid values: private, public-read, and public-read-write
ID	String	1746495857******	The user ID of the bucket owner.
Owner	Container	N/A	The container that stores the information about the bucket owner.

Examples

Sample requests

```
GET /test-object? acl HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 29 Apr 2015 05:21:12 GMT
Authorization: OSS q***c:C***=
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***1
Date: Wed, 29 Apr 2015 05:21:12 GMT
Content-Length: 253
Content-Type: application/xml
Connection: keep-alive
Server: AliyunOSS
<? xml version="1.0" ? >
<AccessControlPolicy>
    <Owner>
        <ID>0***2</ID>
        <DisplayName>0***2</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>public-read </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

2.6.11. PostObject

You can call this operation to upload an object to a specific bucket by using HTML forms.

Usage notes

When you call the PostObject operation, take note of the following items:

- To perform the PostObject operation on a bucket, you must have write permissions on the bucket. If the ACL of the bucket is public read/write, you do not need to upload the signature information. If the ACL of the bucket is not public read/write, signature verification is required for this operation.
- Unlike PutObject, PostObject uses an AccessKey secret to calculate the signature for a policy. The calculated signature string is used as the value of the Signature form field. OSS checks this value to verify the validity of the signature.
- The URL of the submitted form can be the domain name of the bucket. You do not need to specify the object in the URL. In other words, the request line is in the format of `POST / HTTP/1.1` instead of `POST /ObjectName HTTP/1.1`.
- OSS does not check the signature included in headers or URLs in PostObject requests.

Request structure

```
POST / HTTP/1.1
Host: BucketName.regionid.example.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--
```

Request headers

Note The message body of a PostObject request is encoded in the multipart/form-data format. In PostObject operations, parameters are passed as form fields in the request message body, whereas parameters are passed as HTTP request headers in PutObject operations.

Header	Type	Required	Example	Description
OSSAccessKeyId	String	Conditional	LTAI4Fw2NbDUCV8zYU******	The AccessKey ID of the bucket owner. Default value: null Constraint: This form field is required when the bucket ACL is not public read/write or when the policy or Signature form field is provided.
policy	String	Conditional	`{ "expiration": "2014-12-01T12:00:00.000Z", "conditions": [ {"bucket": "johnsmith" }, ["starts-with", "$key", "user/eric/"] ] }`	Specifies the validity of the form fields in the request. A request that does not contain the policy form field is considered to be an anonymous request, and can be used only to access buckets whose ACLs are public read/write. For more information about the policy example, see Post policy. Default value: null Constraint: This form field is required when the bucket ACL is not public read/write or when the OSSAccessKeyId or Signature form field is provided. **Note** The form and the policy form field must be encoded in UTF-8.
Signature	String	Conditional	None	Specifies the signature information that is calculated based on the AccessKey secret and the policy form field. OSS checks the signature information to verify the validity of the PostObject request. Default value: null Constraint: This form field is required when the bucket ACL is not public read/write or when the OSSAccessKeyId or policy form field is provided. **Note** Form fields are case-insensitive, but their values are case-sensitive.

Header	Type	Required	Example	Description
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, and Expires	String	No	None	The HTTP request headers. For more information, see PutObject. Default value: null The form submitted by the PostObject operation must be encoded in the `multipart/form-data` format. For example, the Content-Type header must be in the `multipart/form-data;boundary=xxxxxx` format, where boundary is a boundary string.
x-oss-content-type	String	No	text/h323	You can add the x-oss-content-type header in the message body of a PostObject request to specify the Content-Type of the object to upload. The Content-Type value specified by the x-oss-content-type header has the highest priority. The content type of the object is determined by three headers in the following priority in descending order: x-oss-content-type > Content-Type in the form field > Content-Type. Default value: null
key	String	Yes	/user/a/objectName.txt	The name of the object to upload. If the object name includes a path such as a/b/c/b.jpg, OSS automatically creates corresponding folders. Default value: null
success_action_redirect	String	No	None	The URL to which the client is redirected after the object is uploaded. If this form field is not specified, the returned result is specified by success_action_status. If the upload fails, OSS returns an error code, and the client is not redirected to a URL. Default value: null
success_action_status	String	No	200	The status code returned to the client when the success_action_redirect form field is not specified and the object is uploaded. Default value: 204 Valid values: 200, 201, and 204. • If the value of this form field is set to 200 or 204, OSS returns an empty file and the 200 or 204 status code. • If the value of this form field is set to 201, OSS returns an XML file and the 201 status code. • If the value of this form field is not specified or set to an invalid value, OSS returns an empty file and the 204 status code.
x-oss-meta-*	String	No	x-oss-meta-location	The metadata customized by the user. Default value: null If the request contains a form field prefixed with x-oss-meta-, the form field is considered to be the user metadata. Example: x-oss-meta-location. **Note** An object may have multiple similar parameters prefixed with x-oss-meta-, but the total size of the user metadata cannot exceed 8 KB.
x-oss-server-side-encryption	String	No	KMS	The server-side encryption algorithm that is used when OSS creates the object. Valid values: *AES256* and *KMS* If you specify this parameter, it is returned in the response header and the uploaded object is encrypted and stored. When you download the encrypted object, the x-oss-server-side-encryption field is included in the response header and the field value is set to the algorithm used to encrypt the object.
x-oss-server-side-encryption-key-id	String	No	3dea0cae-5fd6-4bf8-8574-87c557******	The ID of the customer master key (CMK) hosted in KMS. This parameter is valid only when the value of x-oss-server-side-encryption is set to KMS.
x-oss-object-acl	String	No	private	The ACL of the object when the object is created. Valid values: *public-read*, *private*, and *public-read-write*

Header	Type	Required	Example	Description
x-oss-security-token	String	No	None	The security token for temporary authorization. If an STS temporary security credential is used for this access, you must set this field to the SecurityToken value and set OSSAccessKeyId to the value of the paired temporary AccessKey ID. The method to calculate a signature based on a temporary AccessKey ID is the same as that based on a typical AccessKey ID. Default value: null
file	String	Yes	MyFilename.txt	The file or text content. Browsers automatically set the Content-Type header based on the file type and overwrite the user setting. Only one file can be uploaded to OSS at a time. Default value: null Notice This header must be the last field in the form.

Response headers

Header	Type	Example	Description
x-oss-server-side-encryption	String	KMS	If x-oss-server-side-encryption is specified in the request, the response contains this header to indicate the encryption algorithm used to encrypt the object at the server side.

Response elements

Element	Type	Example	Description
PostResponse	Container	N/A	The container that stores the result of the Post request.
Bucket	String	examplebucket	The bucket name.
ETag	String	E64478CDF7F0BFAF602F8D676D5B04E1	The entity tag (ETag) that is created when an object is generated. If an object is created by calling the PostObject operation, the ETag value is the object UUID, which can be used to check whether the object content is changed.
Location	String	https://examplebucket.endpoint/user/a/objectName.txt	The URL of the new object that is created.

Examples

Sample requests

```
POST / HTTP/1.1
Host: ***.regionid.example.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"
eyJleHBpcmF0aW9uIjoiMjAxMy0xMi0wMVQxMjowMDowMFoiLCJjb25kaXRpb25zIjpbWyJjb250ZW50LWxlbmd0aC1yYW5nZSIsIDAsIDEwNDg1NzYwXSx7ImJ1Y2tldCI6ImFoYWhhIn0sIHsiQSI6ICJhIn0seyJrZXkiOiAiQUJDIn1dfQ==
--9431149156168
Content-Disposition: form-data; name="Signature"
k***0+d***k=
--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--
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 6***e
Date: Fri, 24 Feb 2014 06:03:28 GMT
ETag: 5***E
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

Post policy

The policy form field in a PostObject request is used to verify the validity of the request. The policy value is a JSON text encoded in UTF-8 and Base64. This value declares the conditions that a PostObject request must meet. The policy form field is optional for uploading objects to a bucket whose ACL is public read/write. However, to limit PostObject requests, we recommend that you use this field.

- Policy example

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

In the PostObject request, the policy must contain the expiration and conditions attributes.

- Expiration

Expiration specifies the expiration time of the policy, which is expressed in ISO 8601 GMT. For example, "2014-12-01T12:00:00.000Z" indicates that the PostObject request must be sent before 12:00:00 on December 1, 2014.

- Conditions

Conditions is a list that specifies the valid values of form fields in the PostObject request. The value of a form field is extended after OSS checks the policy. Therefore, the valid value of the form field set in policy is equivalent to the value of the form field before extension. For example, if the key form field is set to user/user1/${filename} and the object name before the upload is a.txt, the key form field in policy of the PostObject request must be set to ["eq", "$key", "user/user1/\${filename}"] instead of ["eq", "$key","$key", "user/user1/a.txt"]. The following table lists the conditions supported by the policy.

Conditions

Condition	Description
content-length-range	The acceptable maximum and minimum sizes of the uploaded file. This condition supports the content-length-range match mode.
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	HTTP request headers. This condition supports the exact match and starts-with match modes.
key	The name of the object after the upload. This condition supports the exact match and starts-with match modes.
success_action_redirect	The URL to which the client is redirected after the object is uploaded. This condition supports the exact match and starts-with match modes.
success_action_status	The status code returned after the object is uploaded if success_action_redirect is not specified. This condition supports the exact match and starts-with match modes.
x-oss-meta-*	The metadata customized by the user. This condition supports the exact match and starts-with match modes.

If the PostObject request contains other form fields, these extra form fields can be added to conditions of the policy. OSS does not check validity of the form fields that are not included in conditions.

- Condition match modes

Match mode	Description
Exact match	The value of a form field must be exactly the same as the value that is declared in conditions. For example, if the value of the key form field must be a, the condition can be {"key": "a"} or ["eq", "$key", "a"].
Starts With	The value of a form field must start with the specified value. For example, if the value of the key form field must start with /user/user1, the condition is ["starts-with", "$key", "/user/user1"].
Specified object size	The maximum and minimum sizes of the object that can be uploaded. For example, if the acceptable object size is 1 to 10 bytes, the condition is ["content-length-range", 1, 10].

- Escape characters

In the policy of a PostObject request, a dollar sign ($) is used to indicate a variable. Therefore, to describe the dollar sign ($), you must use escape characters (\$). In addition, some characters in JSON are escaped. The following table describes escape characters for the JSON string of the policy in the PostObject request.

Escape characters

Escape character	Description
\/	Slash
\	Backslash
\"	Double quotation marks

Escape character	Description
\$	Dollar sign
\b	Backspace
\f	Form feed
\n	Line feed
\r	Carriage return
\t	Horizontal tab
\uxxxx	Unicode character

Post Signature

For signature authentication, the HTML form must contain policy and Signature. policy specifies the values that are valid in the request. The procedure to calculate the Signature field value:

1. Create a UTF-8 encoded policy.
2. Encode the policy in Base64. The encoding result is the value of the policy form field, and this value is used as the string-to-sign.
3. Use the AccessKey secret to sign the string. The signing method is the same as the method to calculate the signature in the header, which replaces the string-to-sign with the policy form field value.

2.6.12. Callback

To enable OSS to return callback information of an object to an application server after the object is uploaded to OSS, you need only to include a callback parameter in the upload request that is sent to OSS. This topic describes how to implement upload callback in detail.

> **Note** Operations that support callbacks include PutObject, PostObject, and CompleteMultipartUpload.

Step 1: Create parameters

- Callback parameter

  A callback parameter is a Base64-encoded string (field) in the JSON format. To create a callback parameter, you must specify the URL (callbackUrl) of the server to which the callback information is sent and the content (callbackBody) of the callback information.

  The following table describes the JSON fields that are included in a callback parameter.

Field	Description	Required
callbackUrl	Field descriptions and usage notes: ○ After an object is uploaded, OSS sends a callback request by using the POST method to this URL. The body of the request is the content specified in callbackBody. In most cases, the URL returns an `HTTP/1.1 200 OK` response. The response body must be in the JSON format, and the Content-Length response header value is smaller than 3 MB. ○ You can configure up to five URLs that are separated with semicolons (;) for a request. OSS sends requests to each URL until the first success response is returned. ○ If no URLs are configured or the value of this field is null, OSS determines that the callback feature is not configured. ○ HTTPS-based URLs are supported. ○ To ensure that Chinese characters can be correctly processed, the callback URL must be encoded.	Yes
callbackHost	Field descriptions and usage notes: ○ The value of the Host header in the callback request. This field is valid only when callbackUrl is specified. ○ If callbackHost is not specified, the host values are resolved from the URLs of the callbackUrl field and are specified as the value of callbackHost.	No
callbackBody	Field descriptions and usage notes: ○ The value of the callback request body. Example: key=$(key)&etag=$(etag)&my_var=$(x:my_var). ○ OSS system variables as well as custom variables and constants are supported. The following table lists supported system variables. Custom variables are passed in by using the callback-var parameter in the PutObject and CompleteMultipart operations and by using form fields in the PostObject operation.	Yes
callbackBodyType	Field descriptions and usage notes: ○ The Content-Type header in the callback request. Valid values: application/x-www-form-urlencoded and application/json. Default value: application/x-www-form-urlencoded. ○ If callbackBodyType is set to application/x-www-form-urlencoded, the variables in callbackBodyType are replaced with URL-encoded values. If callbackBodyType is set to application/json, the variables in callbackBodyType are replaced with JSON-formatted values.	No

Examples of the JSON fields:

```
{
"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}"
}
```

The following table lists the system parameters you can configure for callbackBody.

System parameter	Description
bucket	The name of the bucket.
object	The name of the object.
etag	The ETag field configured for the object and returned to the requester.
size	The size of the object to request, which is the total size of the entire object in CompleteMultipartUpload operations.
mimeType	The type of the resource. For example, the resource type of JPEG images is image/jpeg.
imageInfo.height	The height of the image.
imageInfo.width	The width of the image.
imageInfo.format	The format of the image. Examples: JPG and PNG.

Note This parameter applies only to image objects. If the object is not an image, the values of imageInfo.height, imageInfo.width, and imageInfo.format are null.

- Create custom parameters by using callback-var

  You can configure custom parameters by using the callback-var parameter. Custom parameters are key-value pairs in map. You can add the required parameters to the map. When a POST callback request is initiated, OSS adds these custom parameters and the system parameters described in the preceding section to the body of the POST request. This method allows these parameters to be obtained by the requester.

  You can create a custom parameter the way you create a callback parameter, which must be passed in the JSON format. The JSON string is a map that consists of key-value pairs of all custom parameters.

  Note The key of a custom parameter must start with x: and be lowercase letters. Otherwise, OSS returns an error.

  Assume that you must configure two custom parameters x:var1 and x:var2. The value of x:var1 is value1. The value of x:var2 is value2. Created JSON string:

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

Note If the input callback parameter or callback-var parameter is invalid, HTTP status code 400 and error code InvalidArgument are returned. This error occurs in the following scenarios:

- URLs and headers are passed in at the same time to the callback parameter (x-oss-callback) or the callback-var parameter (x-oss-callback-var) in PutObject and CompleteMultipartUpload operations.
- The size of the callback or callback-var parameter exceeds 5 KB. This does not occur in PostObject operations because the callback-var parameter is unavailable in PostObject operations.
- The callback or callback-var parameter is not Base64-encoded or is not in a valid JSON format after the callback or callback-var parameter is decoded.
- The callbackUrl field that is decoded from the callback parameter includes more than five URLs, or the port in the URL is invalid. Example:

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

- The callbackBody field that is decoded from the callback parameter is null.
- The value of the callbackBodyType field decoded from the callback parameter is not `application/x-www-form-urlencoded` or `application/json`.
- The variables in the callbackBody field decoded from the callback parameter are not in the format of ${var}.
- The variables in the callbackBody field decoded from the callback-var parameter are not in the expected JSON format of `{"x:var1":"value1","x:var2":"value2"...}`.

Step 2: Create a callback request

After you create the callback and callback-var parameters, you must add the parameters to the callback request that is sent to OSS.

You can use the following methods to add parameters:

- Add the parameters to a URL.
- Add the parameters to the header.

- Add the parameters to the form fields in the body of a POST request.

  > **Note** You can use only this method to specify callback parameters when you upload objects by using POST requests.

You can use only one of the preceding three methods. If you use more than one method, OSS returns the InvalidArgument error code.

To add the parameters to a request that is sent to OSS, you must use Base64 to encode the JSON string created in the preceding section, and then add the parameters:

- To add the parameters to a URL, add `callback=[CallBack]` or `callback-var=[CallBackVar]` to the request as a URL parameter. When the signature for the CanonicalizedResource field is calculated, callback or callback-var is used as a subresource.
- To add the parameters to the header, add `x-oss-callback=[CallBack]` or `x-oss-callback-var=[CallBackVar]` to the request as a header. When the signature for the CanonicalizedOSSHeaders field is calculated, include x-oss-callback-var and x-oss-callback. Examples:

```
PUT /test.txt HTTP/1.1
Host: callback-test.oss-test.aliyun-inc.com
Accept-Encoding: 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: eyJjYWxsYmFja1VybCI6IjEyMS40My4xMTMuODoyMzQ1Ni9pbmRleC5odG1sIiwgICJjYWxsYmFja0JvZHkiOiJidWNrZXQ9JHtidWNrZXR9Jm9iamVjdD0ke29iamVjdH0mZXRhZz0ke2V0YWd9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pbWVUeXBlfSZpbWFnZUluZm8uaGVpZ2h0PSR7aW1hZ2VJbmZvLmhlaWdodH0maW1hZ2VJbmZvLndpZHRoPSR7aW1hZ2VJbmZvLndpZHRofSZpbWFnZUluZm8uZm9ybWF0PSR7aW1hZ2VJbmZvLmZvcm1hdH0mbXlfdmFyPSR7eDpteV92YXJ9In0=
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 mlepou3zr4u7b14:5a74vhd4UXpmyuudV14Kaen5****
Test
```

- Add the parameters to the form fields in the body of a POST request.
  - It is complex to add the callback parameter when the POST method is used to upload an object because the callback parameter must be added by using a separate form field. Example:

    ```
    --9431149156168
    Content-Disposition: form-data; name="callback"
    eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVuY29kZWQifQ==
    ```

  - Each custom parameter uses a separate form field. You cannot add the callback-var parameter to existing fields. Examples of JSON fields for customer parameters:

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

    The form fields in the POST request:

    ```
    --9431149156168
    Content-Disposition: form-data; name="callback"
    eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkmdGFibGU9JHt4OnRhYmxlfSIsImNhbGxiYWNrQm9keVR5cGUiOiJhcHBsaWNhdGlvbi94LXd3dy1mb3JtLXVybGVuY29kZWQifQ==
    --9431149156168
    Content-Disposition: form-data; name="x:var1"
    value1
    --9431149156168
    Content-Disposition: form-data; name="x:var2"
    value2
    ```

    You can also add callback conditions to the policy (if callback parameters are not added, upload verification is not performed on the policy). Example:

    ```
    { "expiration": "2014-12-01T12:00:00.000Z",
      "conditions": [
        {"bucket": "johnsmith" },
        {"callback": "eyJjYWxsYmFja1VybCI6IjEwLjEwMS4xNjYuMzA6ODA4My9jYWxsYmFjay5waHAiLCJjYWxsYmFja0hvc3QiOiIxMC4xMDEuMTY2LjMwIiwiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JChmaWxlbmFtZSkiLCJjYWxsYmFja0JvZHlUeXBlIjoiYXBwbGljYXRpb24veC13d3ctZm9ybS1lcmxlbmNvZGVkIn0="},
        ["starts-with", "$key", "user/eric/"],
      ]
    }
    ```

Step 3: Initiate a callback request

If an object is uploaded, OSS sends the content specified by the callback and callback-var parameters in the request to the application server by using the POST method. Example:

```
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: http-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
```

Step 4: (Optional) Sign the callback request

If the callback parameter is configured in the request, OSS uses POST to send a callback request to the application server based on the specified callback URL. To verify whether the callback request received by the application server is initiated by OSS, you can sign the callback request.

- Generate a signature

  A callback request is signed by OSS by using the RSA asymmetric algorithm.

  To generate a signature, encrypt the callback string by using a private key. Example:

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

  > **Note** In the preceding code, private_key is a private key known only by OSS, path is the resource path included in the callback request, query_string is the query string, and body is the message body specified in the callback request.

  A callback request is signed in the following steps:

  i. Obtain the callback string to sign, which consists of the resource path that is obtained by decoding the URL, the original query string, a carriage return, and the callback message body.

  ii. Sign the callback string by using the RSA encryption algorithm. Use the private key to encrypt the signature string. The hash function used for the signature is MD5.

  iii. Use Base64 to encode the signed result to obtain the final signature and add the signature to the Authorization header in the callback request.

  Examples:

  ```
  POST /index.php? id=1&index=2 HTTP/1.0
  Host: 121.43.113.8
  Connection: close
  Content-Length: 18
  authorization: kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2t****
  Content-Type: application/x-www-form-urlencoded
  User-Agent: http-client/0.0.1
  x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==
  bucket=yonghu-test
  ```

  In the preceding code, path is set to `/index.php`, and query_string is set to `? id=1&index=2`, and the body is set to `bucket=yonghu-test`. The final signature is `kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==`.

- Verify the signature

  Signature verification is an inverse process of signing a request. The signature is verified by the application server. Process:

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

  The fields in the preceding code and the fields that are used to sign the request have the same meanings. public_key specifies the public key and authorization specifies the signature in the callback request header. Process of verifying the signature:

  i. The x-oss-pub-key-url header in the callback request stores the Base64-encoded URL of the public key. Therefore, you must decode the Base64-encoded URL to obtain the public key.

  ```
  public_key = urlopen(base64_decode(x-oss-pub-key-url header value))
  ```

  > **Note** To ensure that the public key is issued by OSS, you must verify whether the value of the `x-oss-pub-key-url` header starts with `http://gosspublic.alicdn.com/` or `https://gosspublic.alicdn.com/`.

  ii. Obtain the signature decoded in Base64.

  ```
  signature = base64_decode(authorization header value)
  ```

  iii. Obtain the string to sign the way described in the process of signing the callback request.

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

  iv. Verify the signature.

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

  Complete process of verifying the signature:

  i. Obtain the URL of the public key by decoding `aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3YxLnBlbQ==` in Base64. The decoded URL is `http://gosspublic.alicdn.com/callback_pub_key_v1.pem`.

  ii. Decode `kKQeGTRccDKyHB3H9vF+xYMSrmhMZjzzl2/kdD1ktNVgbWEfYTQG0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==` in Base64. The decoded result cannot be displayed because it is a nonprintable string.

  iii. Obtain the string to sign, which is url_decode("index.php") + "? id=1&index=2" + "\n" + "bucket=yonghu-test", and perform MD5 verification on the string.

  iv. Verify the signature.

- Example for an application server to verify a signature:

  The following Python code provides an example of how an application server verifies a signature. Before you run the code, the M2Crypto library must be installed.

```
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 according 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()
```

The code for the server in other programming languages:

Java:

- Download link: Java
- Running method: Decompress the package, and run `java -jar oss-callback-server-demo.jar 9000`. 9000 is the port number and can be customized.

PHP:

- Download link: PHP
- Running method: Deploy the code to an Apache environment because some headers in the PHP code depend on the environment. You can modify the example code based on the environment.

Python:

- Download link: Python
- Running method: Decompress the package, and run `python callback_app_server.py`. Before you run the code, RSA dependencies must be installed.

.NET:

- Download link: .NET
- Running method: Decompress the package, and follow `README.md`.

Go:

- Download link: Go
- Running method: Decompress the package, and follow `README.md`.

Ruby:

- Download link: Ruby
- Running method: Run the ruby aliyun_oss_callback_server.rb command.

Step 5: Return the callback result

The application server returns the response to OSS.

The response to the callback request:

```
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"}
```

Note The response returned by the application server to OSS must contain the Content-Length header. The size of the response body cannot exceed 1 MB.

Step 6: Return the upload result

OSS returns the information that is returned by the application server to the user.

An example of the returned response:

```
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"}
```

Note

- The body of responses to some requests such as CompleteMultipartUpload contains content such as information in the XML format. If you use the upload callback feature, the original body content such as `{"a":"b"}` is overwritten. Exercise caution when you implement upload callback.
- If the upload callback fails, HTTP status code 203 and error code CallbackFailed are returned. This response indicates that the object is uploaded to OSS, but the callback fails. A callback failure only indicates that OSS does not receive the expected callback response. It does not indicate that the application server does not receive a callback request. For example, a callback failure occurs if the response returned by the application server is not in the JSON format.

2.6.13. PutObjectTagging

You can call this operation to configure object tagging.

Usage notes

The object tagging feature uses a key-value pair to tag an object. When you call this operation, take note of the following items:

- Limits on tags
  - You can add up to 10 tags to an object. The tags added to an object must have unique tag keys.
  - A tag key can be up to 128 bytes in length. A tag value can be up to 256 bytes in length.
  - Tag keys and values are case-sensitive.

- The key and value of a tag can contain letters, digits, spaces, and the following special characters:

  + - =. _:/

  If you configure the tags in the HTTP header and the tags contain any of the preceding characters, you can use URL encoding to encode the keys and values of the tags.

- Last-Modified

  The Last-Modified value of an object is not updated when the object tags are changed.

Request structure

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

Request elements

Element	Type	Required	Example	Description
Tagging	Container	Yes	N/A	The collection of tags.
TagSet	Container	Yes	N/A	The collection of tags.
Tag	Container	No	N/A	The collection of tags.
Key	String	No	xx	The key of the tag.
Value	String	No	1	The value of the object tag.

Response headers

Response headers involved in the PutObjectTagging operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

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

Sample responses

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

2.6.14. GetObjectTagging

You can call this operation to query the tag information of an object.

Request structure

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

Request headers

The request headers involved in the PutObjectTagging operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the PutObjectTagging operation contain only common response headers. For more information, see Common response headers.

Response elements

Element	Type	Example	Description
Tagging	Container	N/A	The collection of tags.
TagSet	Container	N/A	The collection of tags.
Tag	Container	N/A	The collection of tags.
Key	String	a	The key of the tag.
Value	String	1	The value of the tag.

Examples

Sample requests

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

Sample responses

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

2.6.15. DeleteObjectTagging

You can call this operation to delete tags of a specific object.

Request structure

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

Request headers

The request headers involved in the DeleteObjectTagging operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the DeleteObjectTagging operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

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

Sample responses

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

2.7. Multipart upload-relevant operations

2.7.1. Overview

OSS provides multipart upload in addition to operations such as PutObject and PostObject.

Multipart upload can be used in the following scenarios but is not limited to:

- Resumable upload of the object is required.
- The size of the object you want to upload is larger than 5 GB.
- The OSS server is frequently disconnected due to poor network conditions.
- The size of the object you want to upload is unknown.

2.7.2. InitiateMultipartUpload

Before you use multipart upload to upload data, you must call the InitiateMultipartUpload operation to notify OSS to initiate a multipart upload task.

The OSS server returns a globally unique upload ID for the InitiateMultipartUpload operation. The upload ID uniquely identifies the multipart upload task. You can initiate operations such as stopping or querying the multipart upload task based on this upload ID.

Usage notes

- When you perform this operation to calculate the signature for authentication, you must add `? uploads` to CanonicalizedResource.
- If you set the x-oss-server-side-encryption header in the InitiateMultipartUpload request, the response contains this header. When each part is uploaded, the server encrypts and stores them. OSS server-side encryption supports the AES-256 and SM4 algorithms. If other algorithms are specified, OSS returns HTTP status code 400 and InvalidEncryptionAlgorithmError. When you upload each part, you do not need to add the x-oss-server-side-encryption request header. If this header is specified, OSS returns HTTP status code 400 and InvalidArgument.

Request structure

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

Request parameters

The encoding-type parameter can be specified in the InitiateMultipartUpload request. OSS uses the specified encoding type to encode the object name in the response.

Parameter	Type	Required	Example	Description
encoding-type	String	No	url	The encoding type of the object names in the response. The object name can contain characters that are encoded in UTF-8. However, the XML 1.0 standard cannot be used to parse certain control characters such as characters with an ASCII value from 0 to 10. You can set the encoding-type parameter to encode the object names in the response. Set the value to url. Default value: null Valid value: url

Request headers

Header	Type	Required	Example	Description
Cache-Control	String	No	no-cache	The web page caching behavior that is specified when the object is downloaded. For more information, visit RFC 2616. Type: string. Default value: null
Content-Disposition	String	No	attachment;filename=oss_download.jpg	The presentational information of the object during the download. For more information, visit RFC 2616. Type: string. Default value: null

Header	Type	Required	Example	Description
Content-Encoding	String	No	utf-8	The content encoding type of the object during the download. For more information, visit RFC 2616. Type: string. Default value: null
Expires	Integer	No	None	The expiration time in milliseconds. For more information, visit RFC 2616. Default value: null
x-oss-server-side-encryption	String	No	AES256	The encryption algorithm that OSS uses to encrypt each uploaded part of an object before storage. Valid values: AES256 and SM4

Response elements

Element	Type	Example	Description
Bucket	String	multipart_upload	The name of the bucket to which the object is uploaded by the multipart upload task.
InitiateMultipartUploadResult	Container	N/A	The container that is used to store the result of the InitiateMultipartUpload request.
Key	String	multipart.data	The name of the object that is uploaded by the multipart upload task.
UploadId	String	0***D	The unique ID of the multipart upload task.
EncodingType	String	url	The encoding type of the object name in the response. If the encoding-type parameter is specified in the request, the object name in the response is encoded.

Examples

Sample requests

```
POST /multipart.data? uploads HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS q***4=
```

Sample responses

```
HTTP/1.1 200 OK
Content-Length: 230
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 4***8
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>0***D</UploadId>
</InitiateMultipartUploadResult>
```

2.7.3. UploadPart

After initializing a MultipartUpload task, you can call this operation to upload data in parts based on the specified object name and upload ID.

Usage notes

- Before you call UploadPart to upload parts, you must call InitiateMultipartUpload to obtain an upload ID issued by the OSS server.
- The part number ranges from 1 to 10000. If a part number is not within this range, OSS returns the InvalidArgument error code.
- In the multipart upload mode, the minimum size of each part except the last part is 100 KB. The UploadPart operation does not immediately verify the sizes of the uploaded parts because this operation cannot immediately determine which part is the last part. The size of the uploaded part is verified only when multipart upload is complete.
- OSS includes the MD5 hash for each part received by the server in the ETag header and returns the ETag header to the user.
- If the x-oss-server-side-encryption request header is specified when you call InitiateMultipartUpload, the uploaded parts are encoded. The x-oss-server-side-encryption header is included in the response to the UploadPart request, which indicates the server-side encryption method of the parts. For more information, see InitiateMultipartUpload.

Request structure

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

Request headers

The request headers involved in the UploadPart operation contain only common request headers. For more information, see Common request headers.

Response headers

The response headers involved in the UploadPart operation contains only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT /multipart.data? partNumber=1&uploadId=0***6 HTTP/1.1
Host: ***.regionid.example.com
Content-Length:6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS q***I=
[6291456 bytes data]
```

Sample responses

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
ETag: 7***9
x-oss-request-id: 3***a
Date: Wed, 22 Feb 2012 08:32:21 GMT
```

2.7.4. UploadPartCopy

You can call this operation to copy data from an existing object to upload a part.

Usage notes

To copy an object larger than 1 GB, use UploadPartCopy. For more information about how to upload an object smaller than 1 GB, see CopyObject.

When you call UploadPartCopy, take note of the following items:

- You cannot call AppendObject to copy data from append objects.
- When you call UploadPartCopy, the source and destination buckets must be located within the same region.
- Before you call UploadPart to upload a part, you must call InitiateMultipartUpload to obtain an upload ID issued by the OSS server.
- If the x-oss-server-side-encryption request header is specified when you call InitiateMultipartUpload, the uploaded part is encoded. The x-oss-server-side-encryption header is included in the response header of UploadPart, which indicates the server-side encryption method of the part.
- In multipart upload mode, each part except the last part must be larger than 100 KB in size. The size of each part is not verified when you call UploadPart because not all parts are uploaded and the system does not know which part is the last part. The size of each part is verified only when you call CompleteMultipartUpload.

Request structure

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

Request headers

The following table describes the x-oss-copy-source and x-oss-copy-source-range request headers except for common request headers in the UploadPartCopy request.

Header	Type	Required	Example	Description
x-oss-copy-source	String	Yes	oss-example/src-object	The source object address. You must have permissions to read the source object. Default value: null

Header	Type	Required	Example	Description
x-oss-copy-source-range	Integer	No	bytes=100-6291756	The range of bytes to copy data from the source object. For example, if you set bytes to 0 to 9, the system transfers byte 0 to byte 9, a total of 10 bytes. Default value: null • If the x-oss-copy-source-range request header is not specified, the entire source object is copied. • If the x-oss-copy-source-range request header is specified, the response contains the length of the entire object and the range of bytes to be copied for this operation. For example, Content-Range: bytes 0~9/44 indicates that the length of the entire object is 44 bytes. The range of bytes to be copied is byte 0 to byte 9. • If the specified range does not conform to the range conventions, OSS copies the entire object and does not include Content-Range in the response.

The following table describes request headers that are used for x-oss-copy-source to specify the source object.

Header	Type	Required	Example	Description
x-oss-copy-source-if-match	String	No	5B3C1A2E053D763E1B002CC607C5****	The copy operation condition. If the ETag value of the source object is the same as the ETag value provided by the user, OSS copies data. Otherwise, OSS returns 412 Precondition Failed. Type: string Default value: null
x-oss-copy-source-if-none-match	String	No	2019-04-09T07:01:56.000Z	The object copy condition. If the ETag value of the source object is different from the ETag value specified in the request, OSS copies the object. Otherwise, OSS returns 412 Precondition Failed. Type: string Default value: null
x-oss-copy-source-if-unmodified-since	String	No	2019-04-09T07:01:56.000Z	The object copy condition. If the specified time is later than or equal to the actual modified time of the object, OSS transfers the object and returns 200 OK. Otherwise, OSS returns 412 Precondition Failed. Type: string Default value: null
x-oss-copy-source-if-modified-since	String	No	2019-04-09T07:01:56.000Z	The object copy condition. If the source object is modified after the time specified in the request, OSS copies the object. Otherwise, OSS returns 412 Precondition Failed. Type: string Default value: null

Response elements

Element	Type	Example	Description
CopyObjectResult	Container	N/A	The result of the CopyObject operation.
LastModified	String	2019-04-09T07:01:56.000Z	The time when the destination object was last modified.
ETag	String	5B3C1A2E053D763E1B002CC607C5****	The ETag value of the destination object.

Examples

Sample requests

```
PUT /multipart.data? partNumber=1&uploadId=0***6 HTTP/1.1
Host: ***.regionid.example.com
Content-Length: 6291456
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS q***I=
x-oss-copy-source: /oss-example/ src-object
x-oss-copy-source-range:bytes=100-6291756
```

Sample responses

```
HTTP/1.1 200 OK
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 3***a
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>"5***E"</ETag>
</CopyPartResult>
```

2.7.5. CompleteMultipartUpload

You can call this operation to complete multipart upload of an object after all parts are uploaded.

Usage notes

When you call the CompleteMultipartUpload operation, you must provide a complete list of all valid parts, which includes the part number and ETag of each part. After OSS receives the list of parts, OSS verifies the validity of each part one by one. After all these parts are validated, OSS combines these parts into a complete object.

- Confirm the size of each part

  When you perform the CompleteMultipartUpload operation, OSS checks whether each part except the last part is larger than or equal to 100 KB in size and verifies the part number and ETag of each part that are provided in the part list. Therefore, when a part is uploaded, the client must record not only the part number of the part but also the ETag value of the part returned from the server.

- Handle requests

  It may take a while for OSS to process the CompleteMultipartUpload request. If the client is disconnected from OSS during this period, OSS continues to process the request.

- PartNumber

  OSS verifies the value of PartNumber when you call the CompleteMultipartUpload operation.

  The value of PartNumber ranges from 1 to 10000. The part numbers listed in the request do not have to be consecutive but must be sorted in ascending order. For example, if the part number of the first part is 1, the part number of the second part can be 5.

- UploadId

  An object can be uploaded by multiple upload tasks that have independent upload IDs. When one upload task is complete, the corresponding upload ID becomes invalid and the upload IDs of other upload tasks are not affected.

- x-oss-server-side-encryption request header

  If the x-oss-server-side-encryption header is specified in an InitiateMultipartUpload request, this header is returned in the response to the request. The value of the x-oss-server-side-encryption header in the response indicates the method used to encrypt the object on the OSS server.

Request structure

```
POST /ObjectName? uploadId=UploadId HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Content-Length: Size
Authorization: Signature
<CompleteMultipartUpload>
<Part>
<PartNumber>PartNumber</PartNumber>
<ETag>ETag</ETag>
</Part>
...
</CompleteMultipartUpload>
```

Request headers

Header	Type	Required	Description
x-oss-complete-all:yes	String	No	Specifies whether to list all parts uploaded by using the current upload ID. • If x-oss-complete-all:yes is specified in the request, OSS lists all parts that are uploaded by using the current upload ID, sorts the parts by their part numbers, and then performs the CompleteMultipartUpload operation. OSS cannot detect parts that are not uploaded or being uploaded in the CompleteMultipartUpload operation. Therefore, you must ensure that all parts are uploaded before you perform the CompleteMultipartUpload operation. • If x-oss-complete-all:yes is specified in the request, the request body cannot be specified. Otherwise, an error occurs. • The format of the response remains unchanged if x-oss-complete-all:yes is specified in the request.

For more information about other common request headers such as Host and Date, see Common request headers.

Request elements

Element	Type	Required	Example	Description
CompleteMultipartUpload	Container	Yes	N/A	The container that stores the content of the CompleteMultipartUpload request.

Element	Type	Required	Example	Description
ETag	String	No	3***9	The ETag values that are returned by OSS after parts are uploaded.
Part	Container	Yes	N/A	The container that stores the uploaded parts.
PartNumber	Integer	Yes	1	The number of parts.

Response elements

Element	Type	Example	Description
Bucket	String	oss-example	The name of the bucket.
CompleteMultipartUploadResult	Container	N/A	The container that stores the response to the CompleteMultipartUpload request.
ETag	String	B***D-3	The UUID of the object content. The ETag is created to identify the content of an object when the object is created by using the CompleteMultipartUpload request. The ETag value of an object can be used to check whether the object content is changed.
Location	String	http://oss-example.endpoint /multipart.data	The URL of the new object that is created.
Key	String	multipart.data	The name of the created object.
EncodingType	String	utf-8	The encoding type of the object name in the response. If the encoding-type parameter is specified in the request, the object name in the response is encoded.

Examples

Sample requests

```
POST /multipart.data? uploadId=0***4  HTTP/1.1
Host: ***.regionid.example.com
Content-Length: 1056
Date: Fri, 24 Feb 2012 10:19:18 GMT
Authorization: OSS q***0=
<CompleteMultipartUpload>
    <Part>
        <PartNumber>1</PartNumber>
        <ETag>"3***9"</ETag>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <ETag>"8***A"</ETag>
    </Part>
    <Part>
        <PartNumber>8</PartNumber>
        <ETag>"8***2"</ETag>
    </Part>
</CompleteMultipartUpload>
```

Sample responses

```
HTTP/1.1 200 OK
Server: AliyunOSS
Content-Length: 329
Content-Type: Application/xml
Connection: keep-alive
x-oss-request-id: 5***e
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.endpoint /multipart.data</Location>
    <Bucket>oss-example</Bucket>
    <Key>multipart.data</Key>
    <ETag>B***D-3</ETag>
</CompleteMultipartUploadResult>
```

2.7.6. AbortMultipartUpload

You can call this operation to cancel a multipart upload task based on the specified upload ID.

Usage notes

When you call AbortMultipartUpload, take note of the following items:

- You must provide the upload ID of the multipart upload task that you want to cancel.
- When you cancel a multipart upload task, parts that are being uploaded cannot be deleted.
- After a multipart upload task is canceled, the upload ID of the task cannot be used to perform any operations and the parts uploaded by the task are deleted.
- We recommend that you complete or cancel multipart upload tasks that are not complete in a timely manner because parts generated by these tasks consume storage space and incur storage fees.

Request structure

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

Request elements

Element	Type	Required	Example	Description
uploadId	String	Yes	F18A92392DFD4B3FA897C26782******	The ID of the multipart upload task.

For more information about other common request headers such as Host and Date, see Common request headers.

Response headers

Response headers involved in the AbortMultipartUpload operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
Delete /multipart.data? &uploadId=0***E  HTTP/1.1
Host: ***.regionid.example.com
Date: Wed, 22 Feb 2012 08:32:21 GMT
Authorization: OSS q***I=
```

Sample responses

```
HTTP/1.1 204
Server: AliyunOSS
Connection: keep-alive
x-oss-request-id: 0***d
Date: Wed, 22 Feb 2012 08:32:21 GMT
```

2.7.7. ListMultipartUploads

You can call this operation to list all ongoing multipart upload tasks that have been initiated but not complete or canceled.

The list returned by OSS contains a maximum of 1,000 multipart upload tasks. To specify the number of tasks that are returned in the response from OSS, you can add the max-uploads parameter to the request. In the response returned by OSS, the IsTruncated element indicates whether the response contains all the ongoing multipart upload tasks.

Request structure

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

Request parameters

Parameter	Type	Required	Example	Description
delimiter	String	No	None	The delimiter used to group objects by name. If you specify the Delimiter parameter in the request, the response contains the CommonPrefixes element. Objects whose names contain the same string from the prefix to the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes.
max-uploads	String	No	1000	The maximum number of multipart upload tasks to return for one request. If this parameter is not specified, default value 1000 applies. The max-uploads value cannot exceed 1000. Type: string

Parameter	Type	Required	Example	Description
key-marker	String	No	None	The name of the object after which the list begins. This parameter is used together with the upload-id-marker parameter. • If the upload-id-marker parameter is not set, OSS returns multipart upload tasks in which object names are listed after the key-marker value. Multipart upload tasks are listed in alphabetical order of their object names. • If the upload-id-marker parameter is set, OSS returns the following tasks:Multipart upload tasks in which object names are listed after the key-marker value. Multipart upload tasks are listed in alphabetical order of their object names.Multipart upload tasks in which object names are the key-marker value but upload IDs are greater than the upload-id-marker value.
prefix	String	No	None	The prefix that the returned object names must contain. Note that if you use a prefix to make a query, the returned object names still contain the prefix.
upload-id-marker	String	No	None	The upload ID of the multipart upload task after which the list begins. This parameter is used together with the key-marker parameter. • If the key-marker parameter is not set, OSS ignores the upload-id-marker parameter. • If the key-marker is set, the response includes the following tasks:Multipart upload tasks in which object names are listed after the key-marker value. Multipart upload tasks are listed in alphabetical order of their object names.Multipart upload tasks in which object names are equal to the key-marker parameter value but upload IDs are greater than the upload-id-marker parameter value.
encoding-type	String	No	utf-8	The encoding type of the object names in the response. Values of Delimiter, KeyMarker, Prefix, NextKeyMarker, and Key can be encoded in UTF-8. However, the XML 1.0 standard cannot be used to parse control characters such as characters with an ASCII value 0 to 10. You can set the encoding-type parameter to encode values of Delimiter, KeyMarker, Prefix, NextKeyMarker, and Key in the response. Default value: null

Response elements

Element	Type	Example	Description
ListMultipartUploadsResult	Container	N/A	The container that stores the response of the ListMultipartUpload request.
Bucket	String	oss-example	The name of the bucket.
EncodingType	String	UTF-8	The encoding type of the object name in the response. If encoding-type is specified in the request, values of those elements including Delimiter, KeyMarker, Prefix, NextKeyMarker, and Key are encoded in the returned result.
KeyMarker	String	5	The name of the object after which the list begins.
UploadIdMarker	String	None	The upload ID of the multipart upload task after which the list begins.
NextKeyMarker	String	oss.avi	The object name marker in the response for the next request to return the remaining results.
NextUploadMarker	String	0004B99B8E707874FC2D692FA5D77D3F	The NextUploadMarker value that is used for the UploadMarker value in a subsequent request when the response does not contain all required results.
MaxUploads	Integer	1000	The maximum number of upload tasks returned by OSS.
IsTruncated	Enumerated string	false	Indicates whether the multipart upload task list returned in this response is truncated. A value of true indicates that the response does not contain all required results. A value of false indicates that the response contains all required results. Valid values: false and true Default value: false
Upload	Container	N/A	The container that stores the information about multipart upload tasks.
Key	String	multipart.data	The object name in an initiated multipart upload task.
UploadId	String	0004B999EF518A1FE585B0C9360DC4C8	The ID of the multipart upload task.
Initiated	Release date	2012-02-23T04:18:23.000Z	The time when the multipart upload task was initiated.

Detail analysis

- The maximum value of the max-uploads parameter is 1000.
- The results returned by OSS are listed based on the alphabetical order of object names. If multipart upload tasks involve the same object, the results are listed in

ascending order of time.

- You can use prefixes to group and manage objects in a bucket in the same way you manage a folder in a file system.
- ListMultipartUploads supports the following request parameters: prefix, marker, delimiter, upload-id-marker, and max-uploads. You can use one or more of the preceding parameters to configure rules to query multipart upload tasks so that results that meet conditions are returned.

Examples

Sample requests

```
Get /? uploads  HTTP/1.1
Host:***.regionid.example.com
Date: Thu, 23 Feb 2012 06:14:27 GMT
Authorization: OSS q***Y=
```

Sample responses

```
HTTP/1.1 200
Server: AliyunOSS
Connection: keep-alive
Content-length: 1839
Content-type: application/xml
x-oss-request-id: 5***a
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>0***F</NextUploadIdMarker>
    <Delimiter></Delimiter>
    <Prefix></Prefix>
    <MaxUploads>1000</MaxUploads>
    <IsTruncated>false</IsTruncated>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0***8</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>multipart.data</Key>
        <UploadId>0***5</UploadId>
        <Initiated>2012-02-23T04:18:23.000Z</Initiated>
    </Upload>
    <Upload>
        <Key>oss.avi</Key>
        <UploadId>0***F</UploadId>
        <Initiated>2012-02-23T06:14:27.000Z</Initiated>
    </Upload>
</ListMultipartUploadsResult>
```

2.7.8. ListParts

You can call this operation to list all uploaded parts that are mapped to a specific upload ID.

Usage notes

- ListParts supports two request parameters: max-parts and part-number-marker.
- The maximum value of the max-parts parameter is 1000. The default value is 1000.
- The results returned by OSS are listed based on the ascending order of part numbers.
- Errors may occur during network transmission. We recommend that you do not use the results (part numbers and ETag values) of ListParts to generate the final part list of CompleteMultipart.

Request structure

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

Request parameters

Parameter	Type	Required	Example	Description
uploadId	String	No	0004B999EF5A239BB9138C6227D69F95	The ID of the multipart upload task. Default value: null

Parameter	Type	Required	Example	Description
max-parts	Integer	No	1000	The maximum number of parts that can be returned by OSS. Type: integer Default value: 1000
part-number-marker	Integer	No	5	The number of the part after which the list begins. All parts whose part numbers are greater than this parameter value are listed. Type: integer Default value: null
encoding-type	String	No	url	The encoding type of the object name in the response. The object name can contain any characters that are encoded in UTF-8. However, the XML 1.0 standard cannot be used to parse control characters such as characters with an ASCII value from 0 to 10. You can set the encoding-type parameter to encode the object names in the response. Set the value to url.

Response elements

Element	Type	Example	Description
ListPartsResult	Container	N/A	The container that stores the response to the ListParts request.
Bucket	String	multipart_upload	The name of the bucket.
EncodingType	String	url	The encoding type of the object name in the response. If the encoding-type parameter is specified in the request, the object name in the response is encoded.
Key	String	multipart.data	The name of the object.
UploadId	String	0004B999EF5A239BB9138C6227D69F95	The ID of an upload task.
PartNumberMarker	Integer	5	The number of the part after which the list begins. All parts whose part numbers are greater than this parameter value are listed.
NextPartNumberMarker	Integer	5	The NextPartNumberMarker value that is used for the PartNumberMarker value in a subsequent request when the response does not contain all required results.
MaxParts	Integer	1000	The maximum number of parts in this response.
IsTruncated	Enumerated string	false	Indicates whether the part list returned in this response is truncated. A value of true indicates that the response does not contain all required results. A value of false indicates that the response contains all required results. Valid values: true and false
Part	Container	N/A	The container that stores part information.
PartNumber	Integer	1	The number that identifies a part.
LastModified	Release date	2012-02-23T07:01:34.000Z	The time when the part was uploaded.
ETag	String	3349DC700140D7F86A0784842780****	The ETag value of the content of the uploaded part. Type: Parent nodes: ListPartsResult and Part
Size	Integer	6291456	The size of the uploaded part.

Examples

Sample requests

```
Get  /multipart.data? uploadId=0***5  HTTP/1.1
Host: ***.regionid.example.com
Date: Thu, 23 Feb 2012 07:13:28 GMT
Authorization: OSS q***8=
```

Sample responses

```
HTTP/1.1 200
Server: AliyunOSS
Connection: keep-alive
Content-length: 1221
Content-type: application/xml
x-oss-request-id: 1***c
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>0***5</UploadId>
    <NextPartNumberMarker>5</NextPartNumberMarker>
    <MaxParts>1000</MaxParts>
    <IsTruncated>false</IsTruncated>
    <Part>
        <PartNumber>1</PartNumber>
        <LastModified>2012-02-23T07:01:34.000Z</LastModified>
        <ETag>3***9</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>2</PartNumber>
        <LastModified>2012-02-23T07:01:12.000Z</LastModified>
        <ETag>3***9</ETag>
        <Size>6291456</Size>
    </Part>
    <Part>
        <PartNumber>5</PartNumber>
        <LastModified>2012-02-23T07:02:03.000Z</LastModified>
        <ETag>7***9</ETag>
        <Size>1024</Size>
    </Part>
</ListPartsResult>
```

2.8. CORS

2.8.1. Overview

Cross-origin resource sharing (CORS) allows web applications to access resources across origins. OSS allows you to use CORS to develop more flexible web applications. OSS provides CORS-related API operations for developers to control cross-origin access.

2.8.2. PutBucketcors

You can call this operation to configure a cross-origin resource sharing (CORS) rule for a specific bucket. If a CORS rule is configured for the bucket, PutBucketCORS overwrites the existing rule. By default, CORS is disabled for buckets. All cross-origin requests are not allowed.

Usage notes

When you call PutBucketCors, take note of the following items:

- Disabled-by-default design

  By default, CORS is disabled for buckets. All cross-origin requests are not allowed.

- Overwriting mechanism

  When you call PutBucketCors to configure a new CORS rule for a bucket for which a CORS rule is already configured, the existing rule is overwritten.

- Usage in applications

  When you use CORS in applications, you must call PutBucketCors to configure a CORS rule to enable CORS.

  For example, if you want to use the `XMLHttpRequest` feature of your browser to access OSS from the domain name `www.a.com`, you must call PutBucketCORS to configure a CORS rule that is described by an XML message body.

- CORS rule matching

  When OSS receives a cross-origin request or an OPTIONS request sent for a bucket, OSS reads the CORS rules configured for the bucket and checks the permission of the request. OSS matches the request with the rules one by one. When OSS finds the first match, OSS returns a corresponding header in the response. If no match is found, OSS does not include any CORS header in the response.

  The following conditions must be met before OSS determines that a CORS rule matches the request:

  - The origin from which the request is initiated must match the value of one `AllowedOrigin` element in the CORS rule.
  - The method of the request such as GET or PUT or the method corresponding to the `Access-Control-Request-Method` header in an OPTIONS request must match the value of one `AllowedMethod` element in the CORS rule.
  - Each header included in `Access-Control-Request-Headers` in an OPTIONS request must match the value of one `AllowedHeader` element in the CORS rule.

Request structure

```
PUT /? cors HTTP/1.1
Date: GMT Date
Content-Length: ContentLength
Content-Type: application/xml
Host: BucketName.regionid.example.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 >
```

Request elements

Element	Type	Required	Example	Description
CORSRule	Container	Yes	N/A	The container for CORS rules. A maximum of 10 rules can be configured for each bucket.
AllowedOrigin	String	Yes	*	The origin from which a request is allowed. You can use multiple elements to specify multiple origins. Each rule can contain only one asterisk (*) as the wildcard. If AllowedOrigin is set to an asterisk (*), all cross-origin requests are allowed.
AllowedMethod	Enumerated string	Yes	GET	The cross-origin request methods that are allowed. Valid values: GET, PUT, DELETE, POST, and HEAD
AllowedHeader	String	No	Authorization	Specifies whether the headers specified by Access-Control-Request-Headers in the preflight (OPTIONS) request are allowed. Each header specified by Access-Control-Reqluest-Headers must match the value of an AllowedHeader element. Each rule can contain only one asterisk (*) as the wildcard.
ExposeHeader	String	No	x-oss-test	The list of headers that can be exposed to the browser. The headers are the response headers that allow access from an application such as XMLHttpRequest in JavaScript. Asterisks (*) cannot be included as wildcards in the value of this parameter.
CORSConfiguration	Container	Yes	N/A	The container that stores the CORS rules configured for a bucket.
MaxAgeSeconds	Integer	No	100	The time the browser can cache the response to a preflight (OPTIONS) request to a specific resource. Unit: seconds. Only one MaxAgeSeconds element can be configured for each CORS rule.

Response headers

The response headers involved in the PutBucketcors operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
PUT /? cors HTTP/1.1
Host: ***.regionid.example.com
Content-Length: 186
Date: Fri, 04 May 2012 03:21:12 GMT
Authorization: OSS q***A=
<? 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 >
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***F
Date: Fri, 04 May 2012 03:21:12 GMT
Content-Length: 0
Connection: keep-alive
Server: AliyunOSS
```

2.8.3. GetBucketcors

You can call this operation to query the existing CORS rules of a specified bucket.

Request structure

```
GET /? cors HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

The request headers involved in the GetBucketcors operation contain only common request headers. For more information, see Common request headers.

Response elements

Element	Type	Example	Description
CORSRule	Container	N/A	The container that stores CORS rules. Up to 10 rules can be configured for a bucket.
AllowedOrigin	String	*	The sources from which cross-origin requests are allowed. If AllowedOrigin is set to an asterisk (*), cross-origin requests from all sources are allowed.
AllowedMethod	Enumerated string	GET	The cross-origin request methods that are allowed. Valid values: GET, PUT, DELETE, POST, and HEAD
AllowedHeader	String	*	Indicates whether the headers specified by Access-Control-Request-Headers in the OPTIONS preflight request are allowed. Each header specified by Access-Control-Request-Headers must match the value of an AllowedHeader element.
ExposeHeader	String	x-oss-test	The response headers for allowed access requests from applications, such as an XMLHttpRequest object in JavaScript.
MaxAgeSeconds	Integer	100	The period of time within which the browser can cache the response for an OPTIONS preflight request to specific resources. A CORS rule can contain only one MaxAgeSeconds parameter.
CORSConfiguration	Container	N/A	The container that stores the CORS rules configured for a bucket.

Examples

Sample requests

```
Get /? cors HTTP/1.1
Host: ***.regionid.example.com
Date: Thu, 13 Sep 2012 07:51:28 GMT
Authorization: OSS q***w=
```

Sample responses if CORS rules exist

```
HTTP/1.1 200
x-oss-request-id: 5***F
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>
```

2.8.4. DeleteBucketcors

You can call this operation to delete CORS rules.

Request structure

```
DELETE /? cors HTTP/1.1
Host: BucketName.regionid.example.com
Date: GMT Date
Authorization: SignatureValue
```

Request headers

Request headers involved in the DeleteBucketcors operation contain only common request headers. For more information, see Common request headers.

Response headers

Response headers involved in the DeleteBucketcors operation contain only common response headers. For more information, see Common response headers.

Examples

Sample requests

```
DELETE /? cors HTTP/1.1
Host: ***.regionid.example.com
Date: Fri, 24 Feb 2012 05:45:34 GMT
Authorization: OSS q***g=
```

Sample responses

```
HTTP/1.1 204 No Content
x-oss-request-id: 5***C
Date: Fri, 24 Feb 2012 05:45:34 GMT
Connection: keep-alive
Content-Length: 0
Server: AliyunOSS
```

2.8.5. OptionObject

Before a cross-origin request is sent, the browser sends a preflight (OPTIONS) request that includes a specific origin, HTTP method, and header information to OSS to determine whether to send the cross-origin request.

OSS allows you to call the PutBucketcors operation to enable CORS for buckets. OSS determines whether to allow the preflight request of the browser based on the specified rules after CORS is enabled. If OSS rejects this request or CORS is disabled, 403 Forbidden is returned.

Request structure

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.regionid.example.com
Origin:Origin
Access-Control-Request-Method:HTTP method
Access-Control-Request-Headers:Request Headers
```

Request headers

Header	Type	Required	Example	Description
Origin	String	Yes	http://www.example.com	The origin of the request, used to identify a cross-origin request. Default value: null

Header	Type	Required	Example	Description
Access-Control-Request-Method	String	Yes	PUT	The method to use in the cross-origin request. Default value: null
Access-Control-Request-Headers	String	Yes	x-oss-test	The request headers, except for standard headers, to use in the cross-origin request. Default value: null

Response headers

Header	Type	Example	Description
Access-Control-Allow-Origin	String	http://www.example.com	The origin that is included in the request. If the request is denied, the response does not contain the header.
Access-Control-Allow-Methods	String	PUT	The HTTP method of the request. If the request is denied, the response does not contain the header.
Access-Control-Allow-Headers	String	x-oss-test	The list of headers included in the request. If the request includes headers that are not allowed, the response does not contain the headers and the request is denied.
Access-Control-Expose-Headers	String	Content-Length	The list of headers that are accessible to JavaScript applications on a client.
Access-Control-Max-Age	Integer	100	The maximum duration for the browser to cache preflight results. Unit: seconds.

Examples

Sample requests

```
OPTIONS /testobject HTTP/1.1
Host: ***.regionid.example.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
```

Sample responses

```
HTTP/1.1 200 OK
x-oss-request-id: 5***C
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
```

2.9. Error responses

If an error occurs when you access OSS, OSS returns the error code and error message so that you can locate and rectify the error.

OSS error response format

If an error occurs when you access OSS, OSS returns HTTP status code 3xx, 4xx, or 5xx and a message body in the application/xml format.

Example of an error response body:

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

All error message bodies include the following elements:

- Code: the error code that OSS returns to the user.
- Message: the detailed error message provided by OSS.
- RequestId: the UUID that uniquely identifies a request. If a problem persists, submit the request ID to OSS technical support personnel to seek help.

- HostId: the ID of the host in the accessed OSS cluster, which is the same as the host ID specified in the request.

For more information about the elements in error messages, see specific request descriptions.

Error codes

The following table lists the OSS error codes.

Error codes

Error code	Description	HTTP status code
AccessDenied	The error message returned because the access is denied.	403
BucketAlreadyExists	The error message returned because the bucket already exists.	409
BucketNotEmpty	The error message returned because the bucket is not empty.	409
EntityTooLarge	The error message returned because the entity is too large.	400
EntityTooSmall	The error message returned because the entity is too small.	400
FieldItemTooLong	The error message returned because the total length of the form fields in the POST request exceeds the limit.	400
FilePartInterity	The error message returned because the part has changed.	400
FilePartNotExist	The error message returned because the part does not exist.	400
FilePartStale	The error message returned because the part has expired.	400
IncorrectNumberOfFilesInPOSTRequest	The error message returned because the number of objects in the POST request is invalid.	400
InvalidArgument	The error message returned because the parameter format is invalid.	400
InvalidAccessKeyId	The error message returned because the AccessKey ID does not exist.	403
InvalidBucketName	The error message returned because the bucket name is invalid.	400
InvalidDigest	The error message returned because Content-MD5 you specified is invalid.	400
InvalidEncryptionAlgorithmError	The error message returned because the specified encryption algorithm based on entropy encoding is invalid.	400
InvalidObjectName	The error message returned because the object name is invalid.	400
InvalidPart	The error message returned because the part is invalid.	400
InvalidPartOrder	The error message returned because the part sequence is invalid.	400
InvalidPolicyDocument	The error message returned because the content of the form does not meet the conditions specified in the policy document.	400
InvalidTargetBucketForLogging	The error message returned because the destination bucket specified for logging is invalid.	400
InternalError	The error message returned because an internal OSS error occurs.	500
MalformedXML	The error message returned because the XML format is invalid.	400
MalformedPOSTRequest	The error message returned because the format of the POST request body is invalid.	400
MaxPOSTPreDataLengthExceededError	The error message returned because the POST request body, excluding the object content to upload, is too large.	400
MethodNotAllowed	The error message returned because the method is not supported.	405

Error code	Description	HTTP status code
MissingArgument	The error message returned because the required parameters are not set.	411
MissingContentLength	The error message returned because the content length is not specified.	411
NoSuchBucket	The error message returned because the specified bucket does not exist.	404
NoSuchKey	The error message returned because the specified object does not exist.	404
NoSuchUpload	The error message returned because the specified multipart upload ID does not exist.	404
NotImplemented	The error message returned because the method cannot be implemented.	400
PreconditionFailed	The error message returned because the precondition you specified does not hold.	412
RequestTimeTooSkewed	The error message returned because the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes.	403
RequestTimeout	The error message returned because the request times out.	400
RequestIsNotMultiPartContent	The error message returned because the content-type value specified in the POST request is invalid.	400
SignatureDoesNotMatch	The error message returned because the request signature provided does not match the signature calculated by OSS.	403
TooManyBuckets	The error message returned because the number of the buckets in your account has exceeded the limit.	400

Operations not supported by OSS

If an operation not supported by OSS is performed to access a resource, OSS returns HTTP status code 405 and error code MethodNotAllowed.

Sample error requests

```
ABC /1.txt HTTP/1.1
Host: bucketname.regionid.example.com
Date: Thu, 11 Aug 2016 03:53:40 GMT
Authorization: signatureValue
```

Sample responses

```
HTTP/1.1 405 Method Not Allowed
Server: AliyunOSS
Date: Thu, 11 Aug 2016 03:53:44 GMT
Content-Type: application/xml
Content-Length: 338
Connection: keep-alive
x-oss-request-id: 57ABF6C8BC4D25D86CBA5ADE
Allow: GET DELETE HEAD PUT POST OPTIONS
<? xml version="1.0" encoding="UTF-8"? >
<Error>
 <Code>MethodNotAllowed</Code>
 <Message>The specified method is not allowed against this resource. </Message>
 <RequestId>57ABF6C8BC4D25D86CBA5ADE</RequestId>
 <HostId>bucketname.regionid.example.com</HostId>
 <Method>abc</Method>
 <ResourceType>Bucket</ResourceType>
</Error>
```

Note If the accessed resource is /bucket/, ResourceType must be set to Bucket. If the accessed resource is /bucket/object, ResourceType must be set to Object.

Operations supported by OSS but not supported by parameters

If parameters not supported by OSS are added to the request for the object supported by OSS (for example, the If-Modified-Since parameter is added to the PUT operation request), OSS returns 400 Bad Request.

Sample error requests

```
PUT /abc.zip HTTP/1.1
Host: bucketname.regionid.example.com
Accept: */*
Date: Thu, 11 Aug 2016 01:44:50 GMT
If-Modified-Since: Thu, 11 Aug 2016 01:43:51 GMT
Content-Length: 363
```

Sample responses

```
HTTP/1.1 400 Bad Request
Server: AliyunOSS
Date: Thu, 11 Aug 2016 01:44:54 GMT
Content-Type: application/xml
Content-Length: 322
Connection: keep-alive
x-oss-request-id: 57ABD896CCB80C366955187E
x-oss-server-time: 0
<? xml version="1.0" encoding="UTF-8"? >
<Error>
 <Code>NotImplemented</Code>
 <Message>A header you provided implies functionality that is not implemented. </Message>
 <RequestId>57ABD896CCB80C366955187E</RequestId>
 <HostId>bucketname.regionid.example.com</HostId>
 <Header>If-Modified-Since</Header>
</Error>
```

3.ASAPI Developer Guide

3.1. Preparations

3.1.1. Obtain API information

To call API operations by using an SDK, you must first obtain information about the service name, API operation name, and API version.

Context

The information on the API Directory page of the Apsara Uni-manager Management Console is for reference only. For more information about how to call API operations, see the developer guides of cloud services. If the information on the API Directory page is different from that in the developer guide of a cloud service, the information in the developer guide prevails.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Products** and choose **Others** > **API Tool**.
3. In the left-side navigation pane, click **API Directory**.
4. On the **API Directory** page, select the required cloud service from the Product Name drop-down list, enter a keyword of an operation name in the search box, and then click Search.
5. In the search result, you can view the **cloud service name**, **API operation name**, and **API version**.

3.1.2. Obtain an AccessKey pair

AccessKey pairs are classified into AccessKey pairs assigned by Resource Access Management (RAM) and those assigned by Security Token Service (STS). You can use either type of AccessKey pair for authentication when you call operations. This topic describes how to obtain an AccessKey pair that is assigned by RAM.

Context

An AccessKey pair is used to verify the identity of a request sender based on symmetric encryption. An AccessKey pair consists of an AccessKey ID and an AccessKey secret. The AccessKey ID is used to identify a user. The AccessKey secret is used to encrypt the signature string.

AccessKey credentials can be assigned by RAM or STS:

- Use RAM to assign an AccessKey ID and an AccessKey secret to a third party.
- Use STS to assign an AccessKey ID, an AccessKey secret, and a security token to a third party. For more information, see Obtain an STS AccessKey pair.

The Apsara Uni-manager Management Console provides AccessKey pairs for personal accounts and organizations. Only operations administrators and the administrators of level-1 organizations can obtain the AccessKey pairs of organizations. We recommend that you use the AccessKey pairs of personal accounts to call the operations provided by the Apsara Uni-manager Management Console and cloud services. If you use the AccessKey pair of a personal account, configure the request headers that are described in the following table for access control. Otherwise, an error that indicates insufficient permissions may occur.

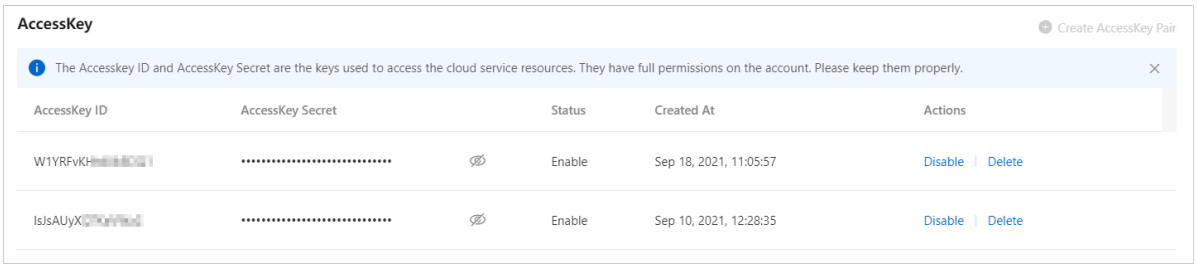
Warning The AccessKey pair of a personal account is managed and controlled by the permission system of the Apsara Uni-manager Management Console. If you need to obtain the AccessKey pair of an organization, make sure that your operation is allowed by an administrator. This ensures security.

Parameter	Description
x-acs-regionid	The region where the bucket resides. Example: cn-hangzhou-*.
x-acs-organizationid	The ID of the organization in the Apsara Uni-manager Management Console. For more information about how to obtain organization IDs, see the "GetOrganizationList" section of *Apsara Uni-manager Management Console Developer Guide*.
x-acs-resourcegroupid	The ID of the resource set in the Apsara Uni-manager Management Console. For more information about how to obtain resource set IDs, see the "ListResourceGroup" section of *Apsara Uni-manager Management Console Developer Guide*.
x-acs-instanceid	The name of the bucket that you want to operate. Example: examplebucket. You can find the OSS bucket name in the bucket list. For more information, see the "Obtain a bucket name" section of the Obtain the values of common header parameters topic.

Obtain the AccessKey pair of a personal account

To obtain the AccessKey pair of a personal account, perform the following steps:

1. Log on to the Apsara Uni-manager Management Console.
2. In the upper-right corner of the homepage, click the profile picture and then click **User Information**.
3. In the **AccessKey Pair** section, view the AccessKey pair of your account.



> **Note** The AccessKey pair consists of an AccessKey ID and an AccessKey secret. These credentials provide you with full permissions on Apsara Stack resources. You must keep the AccessKey pair confidential.

Obtain the AccessKey pair of an organization

To obtain the AccessKey pair of an organization, perform the following steps:

1. Log on to the Apsara Uni-manager Management Console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane, choose **Resources > Organizations**.
4. In the organization list, click the name of the level-1 organization whose AccessKey pair you want to obtain.
5. Click **Management Accesskey**.
6. In the dialog box that appears, view the AccessKey pair of the organization.

3.1.3. Obtain the values of common header parameters

You must provide multiple header parameters when you call API operations. This topic describes how to obtain the values of common header parameters. This topic also explains some response parameters.

Obtain a region ID

1. Log on to the Apsara Uni-manager Management Console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane, choose **Resources** > **Regions**.
4. Click the name of the organization whose region ID you want to view.
5. View the region ID in the **Regions** section.

Obtain an organization ID

You can call the GetOrganizationList operation to obtain the organization ID. For more information, see *GetOrganizationList* in *Apsara Uni-manager Management Console Developer Guide*.

Obtain a resource set ID

You can call the ListResourceGroup operation to obtain the resource set ID. For more information, see *ListResourceGroup* in *Apsara Uni-manager Management Console Developer Guide*.

Obtain a bucket name

You can obtain the name of an OSS bucket in the Apsara Uni-manager Management Console. A bucket name must be in the **x-acs-instanceid** format. To obtain the name of a bucket, perform the following steps:

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, choose **Products** > **Storage** > **Object Storage Service**.
3. In the left-side navigation pane, click **Buckets**.
4. On the **Buckets** page, view the name in the **Bucket Name** column.

Response parameters

The following table describes the additional response parameters returned by default when you call operations to query the instances of cloud services.

Parameter	Description
ResourceGroupName	The name of the resource set.
ResourceGroup	The ID of the resource set.
DepartmentName	The name of the organization.
Department	The ID of the organization.

3.1.4. Obtain an STS AccessKey pair

This topic describes how to use Security Token Service (STS) to call API operations in Apsara Stack.

Background information

STS allows you to manage temporary credentials to your Apsara Stack resources. You can use STS to call API operations in Apsara Stack.

An STS AccessKey pair consists of an AccessKey ID, an AccessKey secret, and a security token. When you use STS to call API operations, you must use an STS AccessKey p

Procedure

1. Obtain the `RAM role`.
    i. Log on to the Apsara Uni-manager Management Console.

ii. Move the pointer over the profile picture in the upper-right corner of the page and click **Personal Information**.

iii. Click **View Current Role Policy**.

In the **View Policies of Current Role** dialog box, view the ARN of your `RAM role`.

2. Obtain the STS AccessKey pair.

> **Note**
>
> In the following code, replace the variables with actual values. Delete the angle brackets <> in the final code.
>
> The following section describes the variables:
>
> - <yourRegionID>: Replace this variable with the actual region ID.
> - <yourAccessKeyID>: Replace this variable with your AccessKey ID.
> - <yourAccessKeySecret>: Replace this variable with your AccessKey secret.
> - <yourRoleSessionName>: Replace this variable with your current session name.
> - <yourRoleArn>: Replace this variable with the ARN of your RAM role.
> - "sts.aliyuns.com": Replace this variable with the endpoint of STS.

```
public static void main(String[] args) {
    // Create and initialize a DefaultAcsClient instance.
    // Specify the region ID, AccessKey ID, and AccessKey secret.
    DefaultProfile profile = DefaultProfile.getProfile("<yourRegionID>", "<yourAccessKeyID>", "<yourAccessKeySecret>");
    HttpClientConfig clientConfig = HttpClientConfig.getDefault();
    clientConfig.setIgnoreSSLCerts(true);
    clientConfig.setProtocolType(ProtocolType.HTTPS);
    profile.setHttpClientConfig(clientConfig);
    IAcsClient client = new DefaultAcsClient(profile);
    String assumeRole = callAssumeRole(client);
    if (StringUtils.isEmpty(assumeRole)) return;
    JSONObject assumeRoleJson = JSONObject.parseObject(assumeRole);
    JSONObject credentials = assumeRoleJson.getJSONObject("Credentials");
    // Obtain the STS AccessKey pair.
    String stsAK = credentials.getString("AccessKeyId");
    String stsSK = credentials.getString("AccessKeySecret");
    String stsToken = credentials.getString("SecurityToken");
}

private static String callAssumeRole(IAcsClient client) {
    CommonRequest request = new CommonRequest();
    // Replace sts.aliyuns.com with the endpoint of STS. The endpoint of STS  is in the sts-vpc.${global:region}.${global:internet-domain} format
.
    request.setSysDomain("sts.aliyuns.com");
    request.setSysProduct("Sts");
    request.setSysAction("AssumeRole");
    request.setSysVersion("2015-04-01");
    request.setSysMethod(MethodType.POST);
    request.setHttpContentType(FormatType.FORM);
    // Replace <yourRoleSessionName> with the name of the current session.
    request.putQueryParameter("RoleSessionName", "<yourRoleSessionName>");
    // Replace "<yourRoleArn>" with the actual ARN of the RAM role that you obtained in the preceding step.
    request.putQueryParameter("RoleArn", "<yourRoleArn>");
    try {
        CommonResponse response = client.getCommonResponse(request);
        return response.getData();
    } catch (ServerException e) {
        System.out.println(e.getErrCode());
        System.out.println(e.getErrMsg());
    } catch (ClientException e) {
        System.out.println(e.getErrCode());
        System.out.println(e.getErrMsg());
    }
    return null;
}
```

3. Use the STS AccessKey pair to call an operation of a service. The following sample code provides an example on how to use the STS AccessKey pair to call the `PutBucket` operation of Object Storage Service (OSS).

> **Note**
>
> In the following code, replace the variables with actual values. Delete the angle brackets <> in the final code.

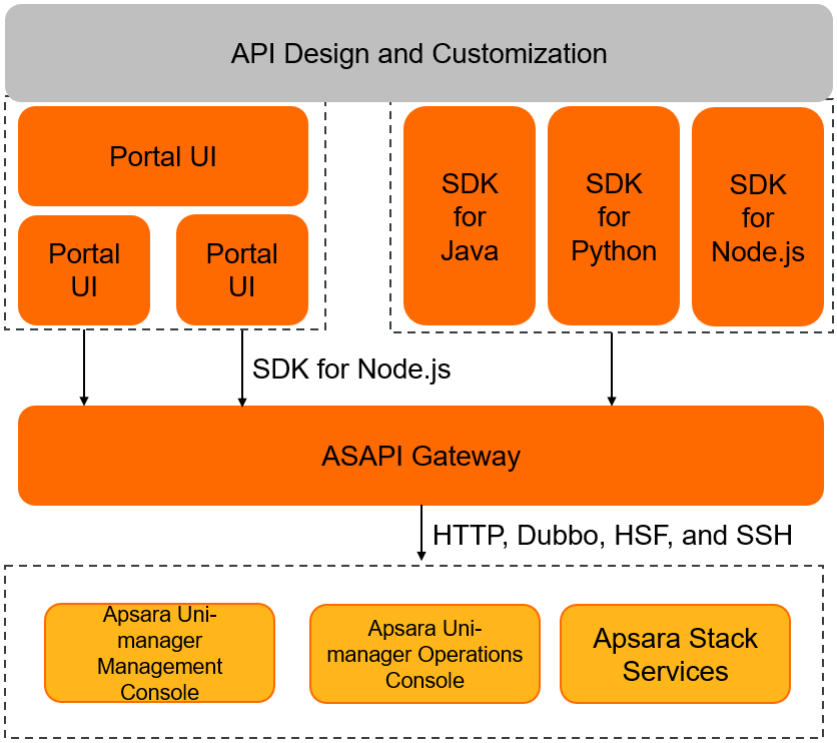
```
import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.aliyun.asapi.ASClient;

import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

public class asapitestdemo {
    public static void main(String[] args) {
        ASClient client = new ASClient();
        // Specify the identifier of the caller. The identifier is used only to record logs.
        client.setSdkSource("asapi@asapi-inc.com");

        // Prepare for initialization and specify the request parameters.
        Map<String, Object> parameters = new HashMap<String, Object>();
        parameters.put(ASClient.PRODUCT, "OneRouter");
        parameters.put(ASClient.ACTION, "DoOpenApi");
        parameters.put(ASClient.VERSION, "2018-12-12");
        String uuId = UUID.randomUUID().toString();
        // Enter your AccessKey ID and AccessKey secret.
        parameters.put(ASClient.ACCESSKEY_ID, "xxx");
        parameters.put(ASClient.ACCESSKEY_SECRET, "xxx");
        // When you use an STS AccessKey pair to call the rpc operation, you must add the SecurityToken field to the request parameters.
        parameters.put("SecurityToken", "<your-securitytoken>");
        // When you use an STS AccessKey pair to call the roa operation, you must add the x-acs-security-token field to the headers.
        // headers.put("x-acs-security-token", "<your-securitytoken>");
        // Enter your actual region ID.
        parameters.put(ASClient.REGIONID, "xxx");
        // Enter the service name.
        parameters.put("ProductName", "oss");
        // Enter the name of the API operation. It must be the same as the name of the API operation in API Reference.
        parameters.put("OpenApiAction", "PutBucket");
        // Enter the user account information.
        parameters.put("AccountInfo", uuId);

        JSONObject params = new JSONObject();
        // Enter the edition type of the service. Set this parameter to enterprise for the enterprise edition.
        params.put("asVersion", "enterprise");
        // Enter the architecture type of CPU.
        params.put("asArchitechture", "arm");
        params.put("SecurityToken", "<yourStsToken>");

        // Enter the parameters in API Reference. The parameters are case-sensitive.

        params.put("BucketName", "asapibucket");
        parameters.put("Params", params.toJSONString());

        Map<String, String> headers = new HashMap<String, String>();
        // Enter the ID of the organization in the Apsara Uni-manager Management Console.
        headers.put(ASClient.X_ACS_ORGANIZATION_ID, "3");
        // Enter the ID of the resource set in the Apsara Uni-manager Management Console.
        headers.put(ASClient.X_ACS_RESOUECEGROUP_ID, "<your-resourcegroupid>");

        // Enter the endpoint of Apsara Stack API (ASAPI).
        String endpoint = "https://asapi.xxx.xxx/asapi/v3";

        // Initiate a request.
        String result = client.doPost(endpoint, headers, parameters);

        // Display the results.
        System.out.println(JSONObject.toJSONString(JSONObject.parseObject(result), SerializerFeature.PrettyFormat));
    }
}
```

3.2. Use ASAPI to make API calls

3.2.1. ASAPI overview

This topic describes the architecture and features of the Apsara Stack API (ASAPI).

The ASAPI is a unified ingress gateway for Apsara Stack services and a main component of the Apsara Uni-manager Management Console. The ASAPI provides a unified method for you to manage and make API calls. You can use Apsara Stack SDK to centrally make API calls to Apsara Stack services, including Elastic Compute Service (ECS) and Virtual Private Cloud (VPC).



The ASAPI allows you to make API calls to manage and maintain Apsara Stack services.

- Make API calls to implement enterprise-grade management. For example, you can manage organizations, users, roles, resource sets, and logon policies. These operations are provided by the Apsara Uni-manager Management Console.
- Make API calls to manage operations. For example, you can manage quotas, usage statistics, specifications, and password policies. These operations are provided by the Apsara Uni-manager Management Console.
- Make API calls to manage resources. For example, you can create, delete, query, or modify cloud resources. These operations are provided by Apsara Stack services. If you call an API operation to create an ECS instance or a VPC, the API operation is provided by ECS or VPC.
- Make API calls to query monitoring metrics of Apsara Stack services. These operations are provided by the Apsara Uni-manager Management Console.
- Make API calls to perform O&M. For example, you can manage monitoring and alerting, inventories, tasks, and physical servers. These operations are provided by the Apsara Uni-manager Operations Console.

For more information, see the developer guide of the corresponding Apsara Stack service. For information about API operations used to manage organization users, see *Apsara Uni-manager Management Console Developer Guide*. For information about the API operation used to create an ECS instance, see *ECS Developer Guide*.

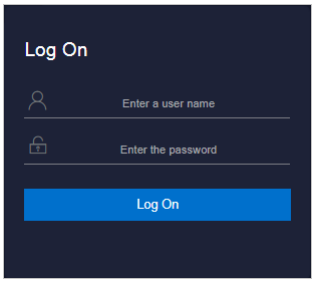
> **Note** If the "Use Apsara Stack SDK to make API calls" section is included in the developer guide of an Apsara Stack service, the ASAPI is supported by the Apsara Stack service. In this case, you can make API calls based on the developer guide.

3.2.2. Obtain the endpoint of ASAPI

This topic describes how to obtain the endpoint of Apsara Stack API (ASAPI) in the Apsara Uni-manager Operations Console.

Procedure

1. In the address bar, enter the URL *region-id*.aso.*intranet-domain-id*.com and press the Enter key.



> **Note** You can select a language from the drop-down list in the upper-right corner of the page.

2. Enter your username and password.

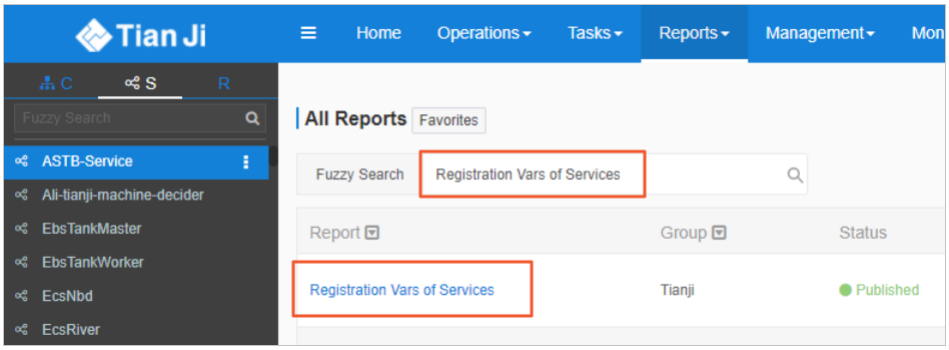
> **Note** Obtain the username and password for logging on to the ASO console from the deployment personnel or an administrator.

When you log on to the ASO console for the first time, you must change the password of your username as prompted.

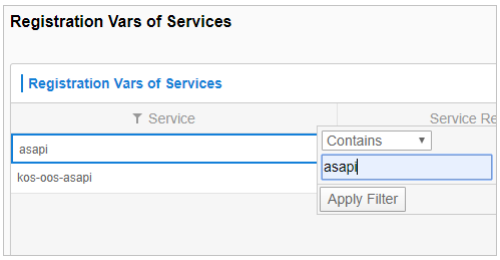
To enhance security, a password must meet the following requirements:

- It must contain uppercase and lowercase letters.
- It must contain digits.
- It must contain special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).
- It must be 10 to 20 characters in length.

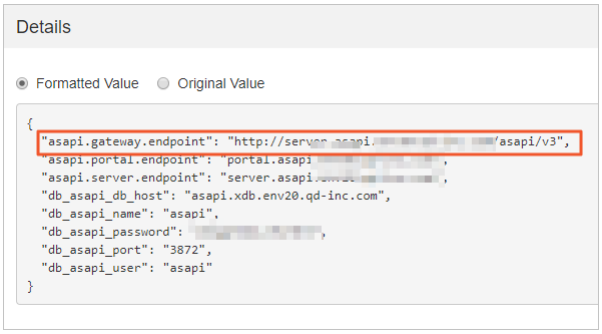
3. Click **Log On** to go to the **ASO** console.
4. In the left-side navigation pane, choose **Product Management>Product List>Apsara Infrastructure Management Framework** to go to the Apsara Infrastructure Manage
5. In the left-side navigation pane, choose **Reports**.
6. On the **All Reports** page, search for **Registration Vars of Services**. Click **Registration Vars of Services**.



7. On the **Registration Vars of Services** page, click the icon next to **Service**. In the dialog box that appears, search for **asapi**.



8. Right-click the **Service Registration** column that corresponds to asapi and select **Show More** from the shortcut menu.
9. In the **Details** dialog box, view the value of **asapi.gateway.endpoint**. The value is the endpoint of ASAPI.



3.2.3. Obtain Apsara Stack SDKs

Apsara Stack API (ASAPI) is a unified portal provided by Apsara Stack. You can use the ASAPI to connect to all Apsara Stack services. Apsara Stack provides SDKs for API calls.

You can go to Apsara Stack documentation and download the Apsara Stack SDKs in the **Other Resources** section at the lower part of the page.

3.2.4. Common parameters

Common request parameters must be included in all API requests.

Common request parameters

The following table describes the common request parameters that are used when you call API operations supported by Apsara Stack API (ASAPI).

Parameter	Type	Required	Description

Parameter	Type	Required	Description
Action	String	Yes	The operation mode. Set this parameter to DoOpenApi.
Product	String	Yes	The method to access the service. Set this parameter to OneRouter.
Version	String	Yes	The API version of the service registered on ASAPI. The current API version is 2018-12-12.
ProductName	String	Yes	The name of the service. Set this parameter to oss.
OpenApiAction	String	Yes	The name of the API operation. It must be the same as the name of the API operation in API Reference.
RegionId	String	Yes	The ID of the region. For more information, see Obtain a region ID.
AccessKeyId	String	Yes	The AccessKey ID that is used to access the service. For information about how to obtain an AccessKey ID, see Obtain an AccessKey pair.
Signature	String	Yes	The signature string. For more information about how signatures are calculated, see Request signatures.
SignatureMethod	String	Yes	The encryption algorithm of the signature string. Valid values: HMAC-SHA1 and HMAC-SHA256.
SignatureVersion	String	Yes	The version of the signature encryption algorithm. Valid values: 1.0 and 2.0.
SignatureNonce	String	Yes	A unique, random number used to prevent replay attacks. You must use different numbers for different requests.
Timestamp	String	Yes	The timestamp of the request. Specify the time in the ISO 8601 standard in the yyyy-MM-ddTHH:mm:ssZ format. The time must be in UTC. Example: 2018-01-01T12:00:00Z.
Format	String	Yes	The format of the response parameters. Set the value to JSON.

Note

The `Signature`, `SignatureMethod`, `SignatureVersion`, `SignatureNonce`, `Timestamp`, and `Format` parameters are encapsulated into Apsara Stack SDK. You do not need to specify these parameters.

3.2.5. SDK description

3.2.5.1. ASAPI SDK for Java

This topic describes how to obtain and use ASAPI SDK for Java to call API operations.

Prerequisites

- An authorized account is created, and an AccessKey pair is obtained. An AccessKey pair consists of an AccessKey ID and an AccessKey secret You can obtain and view your AccessKey pair in the Apsara Uni-manager Management Console. For more information, see Obtain an AccessKey pair.
- The endpoint of ASAPI is obtained. For more information, see Obtain the endpoint of ASAPI.
- OSS SDK for Java 1.8 or later is installed.

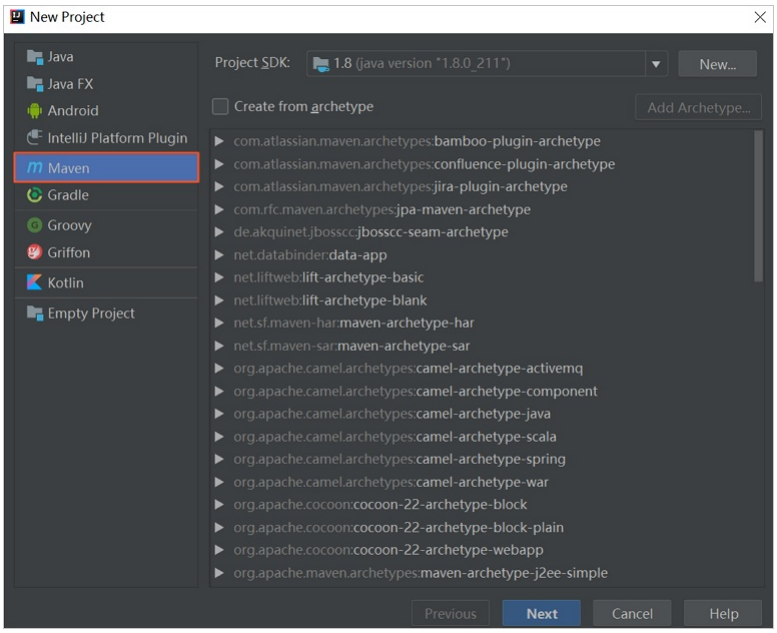
Background information

The help center of Apsara Stack provides Maven dependencies and JAR files for Apsara Stack SDK for Java. You can write code to use Apsara Stack SDK for Java to call API operations.

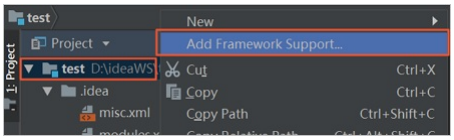
In this example, the Windows 10 64-bit OS and IntelliJ IDEA are used.

Procedure

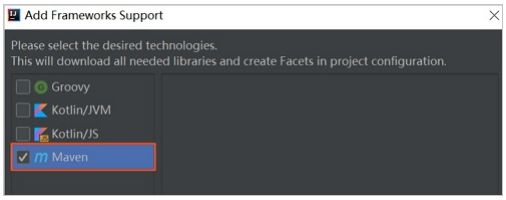
1. Use one of the following methods to configure Maven in IntelliJ IDEA:
   - Use Maven that is integrated in IntelliJ IDEA.
   - Download Maven based on your OS from the official website and configure Maven. For more information, see Downloading Apache Maven 3.8.6.
2. Obtain the Maven dependencies for Apsara Stack SDK for Java from the official website. For more information about how to obtain the dependencies, see Obtain the core package for ASAPI.
3. Use one of the following methods to create a Maven project:
   - Method 1: Create a Maven project in IntelliJ IDEA.



   - Method 2: Convert an existing project to a Maven project.
     a. Right-click the project that you want to convert and select **Add Framework Support...**.



     b. Select **Maven** and click **OK**.



4. Create a folder named libs in the root directory of the project. Then, add the JAR file of Apsara Stack SDK for Java to the libs folder, as shown in the following figure.

5. Add the dependencies of Apsara Stack SDK for Java to the pom.xml file in the project directory.

```
<dependencies>
    <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>asapi</artifactId>
        <version>1.0.6.4-RELEASE</version>
        <scope>system</scope>
        <systemPath>${project.basedir}/libs/asapi-1.0.6.4-RELEASE.jar</systemPath>
    </dependency>
  <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>alicrypto-java-aliyun</artifactId>
    <version>1.0.4</version>
    <systemPath>${project.basedir}/libs/alicrypto-java-aliyun-1.0.4.jar</systemPath>
  </dependency>
  <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>gmsse</artifactId>
    <version>1.1.0</version>
    <systemPath>${project.basedir}/libs/gmsse-1.1.0.jar</systemPath>
  </dependency>
    <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.7</version>
    <systemPath>${project.basedir}/libs/httpclient-4.5.7.jar</systemPath>
  </dependency>
</dependencies>
```

Note Replace `1.0.6.4-RELEASE` with the actual version of Apsara Stack SDK for Java.

6. On the right side of IntelliJ IDEA, view the information in the Maven section.

   If the information in the following figure appears, the dependencies are added.

Use OSS SDK for Java

Currently, you can use ASAPI to call OSS APIs by using OneRouter. The following sample code provides an example on how to call OSS API operations:

```
// 1. Create a connection to an ASClient object.
ASClient client = new ASClient();
// 2. Configure the identifier of the requester. This parameter is required and used only for identification. You can set this parameter to a random value.
client.setSdkSource("<The identifier of the requester>");
Map<String, Object> parameters = new HashMap<String, Object>();
// 3. Configure the parameters for authorization. The parameter values vary based on your environment.
parameters.put(ASClient.PRODUCT, "OneRouter");
parameters.put(ASClient.ACTION, "DoOpenApi");
parameters.put(ASClient.VERSION, "2018-12-12");
String uuId = UUID.randomUUID().toString();
// Specify the AccessKey ID of the authorized account.
parameters.put("AccessKeyId","<AccessKey ID>");
// Specify the AccessKey secret of the authorized account.
parameters.put("AccessKeySecret", "<AccessKey Secret>");
// Specify the region ID.
parameters.put("RegionId", "<your RegionID>");
// 4. Specify the information about the API operation that you want to call.
// Specify the name of the service.
parameters.put("ProductName", "oss");
// Specify the name of the API operation. It must be the same as the name of the API operation in API Reference.
parameters.put("OpenApiAction", "PutBucket");
// Specify the user account information.
parameters.put("AccountInfo", uuId);
// When you use an STS AccessKey pair to call the rpc operation, you must add the SecurityToken field to the request parameters.
// parameters.put("SecurityToken", "<your-securitytoken>");
// When you use an STS AccessKey pair to call the roa operation, you must add the x-acs-security-token field to the headers.
// headers.put("x-acs-security-token", "<your-securitytoken>");
// 5. Configure operation-specific parameters.
JSONObject params = new JSONObject();
parameters.put("<The name of the operation-specific parameter>", "<The value of the operation-specific parameter>");
parameters.put("Params", params.toJSONString());
// 6. Configure request headers and the encryption algorithm for the signature string. Valid values of the SignatureMethod parameter: HMAC-SHA1 and HMAC-SHA256. By default, HMAC-SHA256 is used.
Map<String, String> headers = new HashMap<String, String>();
//headers.put("SignatureMethod", "HMAC-SHA1");
headers.put("x-acs-regionid", "<your-regionid>");
headers.put("x-acs-resourcegroupid", "<your-resourcegroupid>");
headers.put("x-acs-organizationid", "<your-organizationid>");
//headers.put("x-acs-instanceid", "<your-instanceid>");
// 7. Call the API operation. We recommend that you use the doPost() method.
// Configure the connection timeout period.
//client.setConnectTimeout(10000);
// Configure the request timeout period.
//client.setSocketTimeout(10000);
// Enter the endpoint of ASAPI.
String endpoint = "https://public.asapi.xxx.xxx/asapi/v3";
// Initiate a request.
String result = client.doPost(endpoint, headers, parameters);
// Display the results.
System.out.println(result);
```

Configure a proxy

If you need to use a proxy to call API operations, use the following sample code to configure the proxy:

```
Cloud cloud = new Cloud();
// Configure the IP address.
cloud.setConfig("proxy_host", "<IP>");
// Configure the port.
cloud.setConfig("proxy_port", "<Port>");
// Create a connection to an ASClient object.
ASClient client = new ASClient(cloud);
```

Headers

Header	Description
x-acs-regionid	The ID of the region. For more information, see Obtain a region ID.
x-acs-organizationid	The ID of the organization in the Apsara Uni-manager Management Console. For more information, see Query an organization ID.
x-acs-resourcegroupid	The ID of the resource set in the Apsara Uni-manager Management Console. For more information, see Query a resource set ID.
x-acs-instanceid	The name of the bucket on which the operation is performed. For more information, see Query a bucket name.

3.2.5.2. ASAPI SDK for Python

This topic describes how to obtain and use Apsara Stack SDK for Python to call API operations.

Prerequisites

- An authorized account is created, and an AccessKey pair is obtained. An AccessKey pair consists of an AccessKey ID and an AccessKey secret You can obtain and view y manager Management Console. For more information, see Obtain an AccessKey pair.
- The endpoint of the Apsara Stack API (ASAPI) is obtained. For more information, see Generate an endpoint for ASAPI.

Install Apsara Stack SDK for Python

1. Download the required version of Apsara Stack SDK for Python. For more information, see Obtain Apsara Stack SDKs.
2. Upload the compressed file of Apsara Stack SDK for Python to the host from which you want to send API requests and extract the file. Then, run the following code

```
# Install Apsara Stack SDK for Python 2.x.
pip install aliyun-python-sdk-asapi-2.4.7-py2.tar.gz
# Install Apsara Stack SDK for Python 3.x.
pip3 install aliyun-python-sdk-asapi-2.4.7-py3.tar.gz
```

> **Note**
>
> Replace `2.4.7` with the actual version of Apsara Stack SDK for Python.

Call examples

You can use the GET or POST method to call API operations provided by ASAPI.

```
# 1. Add dependencies.
from aliyunsdkasapi.ASClient import ASClient
from aliyunsdkasapi.AsapiRequest import AsapiRequest

if __name__ == '__main__':

    #2. Create a request.
    # Set the access mode to OneRouter, the ASAPI version to 2018-12-12, and the operation mode to DoOpenApi, and enter the ASAPI endpoint based on y
our business requirements.
    req = AsapiRequest("OneRouter", "2018-12-12", "DoOpenApi", "<asapi-endpoint>")

    #3. Specify business information.
    # Set the service name to oss.
    req.add_body_params("ProductName", "oss")
    # Specify the name of the API operation. It must be the same as the name of the API operation in API Reference.
    req.add_body_params("OpenApiAction", "PutBucket")
    req.add_body_params("AccountInfo", "xxx")
    # Specify the API-related parameters. In this example, set BucketName to examplebucket. Modify the parameters based on your business requirements
.
    req.add_body_params("Params", "{\"BucketName\":\"examplebucket\"}")

    # When you use STS AccessKey credentials to call the rpc operation, you must add the SecurityToken field to the request parameters.
    # req.add_body_params("SecurityToken", "<your-securitytoken>");
    # When you use STS AccessKey credentials to call the roa operation, you must add the x-acs-security-token field to the headers.
    # req.add_header("x-acs-security-token", "<your-securitytoken>");

    #4. Specify the header information.
    req.add_header("x-acs-resourcegroupid", "<your-resourcegroupid>");
    req.add_header("x-acs-organizationid", "<your-organizationid>");
    req.add_header("x-acs-regionid", "<your-regionid>");
    req.add_header("x-acs-instanceid", "<your-instanceid>");

    #5. Specify the request mode.
    req.set_method("POST")

    # 6. Use the environment information to initialize an ASClient object.
    # The timeout parameter specifies the timeout period for a request. The cert_file parameter specifies the certificate. The verify parameter speci
fies whether to verify the certificate.
    as_client = ASClient("<accesskeyid>", "<accesskeysecret>", "<regionid>",timeout=1000,
            cert_file=None,
            verify=False)
    # Specify the identifier of the requester. This parameter is required and used only for identification. You can set this parameter to a random va
lue.
    as_client.setSdkSource("<The identifier of the requester>")
    response = as_client.do_request(req)
    print(response)
```

Headers

Header	Description
x-acs-resourcegroupid	The ID of the resource set in the Apsara Uni-manager Management Console. For more information, see Query a resource set ID.
x-acs-organizationid	The ID of the organization in the Apsara Uni-manager Management Console. For more information, see Query an organization ID.

x-acs-regionid	The ID of the region. For more information about how to obtain a region ID, see Obtain a region ID.
x-acs-instanceid	The name of the bucket on which the operation is performed. For more information, see Query a bucket name.

Configure a proxy

Perform the following steps to configure a proxy:

1. Import OS dependencies.

```
import os
```

2. Configure one or more proxies.

```
# Before you call an API operation, configure one or more proxies.
os.environ['http_proxy'] = "http://x.x.x.x:xxxx"
os.environ['https_proxy'] = "https://x.x.x.x:xxxx"
```

3.2.5.3. ASAPI SDK for Node.js

This topic describes how to obtain and use ASAPI SDK for Node.js to call API operations.

Prerequisites

- An authorized account is created, and an AccessKey pair is obtained. An AccessKey pair consists of an AccessKey ID and an AccessKey secret You can obtain and view your AccessKey pair in the Apsara Uni-manager Management Console. For more information, see Obtain an AccessKey pair.
- The endpoint of ASAPI is generated. For more information, see Generate an endpoint for ASAPI.
- tnpm is installed. If tnpm is not installed, run the following command to install tnpm:

```
npm i -g npminstall --registry=http://registry.npm.alibaba-inc.com
npm install -g tnpm --registry=http://registry.npm.alibaba-inc.com
```

Install ASAPI SDK for Node.js

Go to the directory in which you want to install ASAPI SDK for Node.js and run the following command to install ASAPI SDK for Node.js:

```
tnpm i @ali/asapi-nodejs-sdk@0.0.33 --save
```

Use ASAPI SDK for Node.js

The following code provides an example on how to use ASAPI SDK for Node.js to call API operations:

```
// 1. Import the dependencies of ASAPI SDK for Node.js.
const Controller = require('egg').Controller;
const ASAPIClient = require('@ali/asapi-nodejs-sdk');
// 2. Configure the ASAPI client.
const config = {
    API_GATEWAY: '<The endpoint of ASAPI>',
    regionId: '<The region in which the Apsara Stack service is located>',
    accessKeyId: '<The AccessKey ID of the authorized Apsara Stack tenant account>',
    accessKeySecret: '<The AccessKey secret of the authorized Apsara Stack tenant account>'
};
// 3. Initiate the ASAPI client.
const AsapiClient = new ASAPIClient(config, this.ctx);
// 4. Specify the request parameters.
const params = {
    // Set the access mode to OneRouter, the ASAPI version to 2018-12-12, and the operation mode to DoOpenApi, and enter the ID of the region in whic
h the Apsara Stack cloud service is located based on your business requirements.
    "Action": "DoOpenAPI",
    "Product": "OneRouter",
    "Version": "2018-12-12",
    regionId: '<The region in which the Apsara Stack service is located>',
    // When you use an STS AccessKey pair to call the rpc operation, you must specify the SecurityToken parameter.
    // "SecurityToken": "<your-securitytoken>",
    // Specify business parameters in the following format: "<Business parameters required by the parameter>": "<Parameter value>".
    // Set the Apsara Stack service name to oss.
    "ProductName":"oss",
    // Specify the name of the API operation. It must be the same as the name of the API operation in API Reference. Example: PutBucket.
   "OpenApiAction":"PutBucket",
    // Specify the API-related parameters. In this example, BucketName is set to asapibucket. Specify the parameters based on your business requireme
nts.
    "Params":"{\"BucketName\":\"asapibucket\"}"
}
// 5. Add headers.
// When you use an STS AccessKey pair to call the roa operation, you must specify the x-acs-security-token header.
//AsapiClient.addHeader("x-acs-security-token", "<your-securitytoken>")
AsapiClient.addHeader('x-acs-caller-sdk-source', '<The identifier of the requester>')
AsapiClient.addHeader('x-acs-roleid', '<The ID of the role>')
AsapiClient.addHeader('x-acs-userid', '<The ID of the user>')
AsapiClient.addHeader('x-acs-organizationid', '<your-organizationid>')
AsapiClient.addHeadert("x-acs-regionid", "<your-regionid>");
AsapiClient.addHeader("x-acs-resourcegroupid", "<your-resourcegroupid>");
AsapiClient.addHeader("x-acs-instanceid", "<your-instanceid>");
// 6. Initiate a call and obtain the response.
try {
  const res = await AsapiClient.doRequest(config.API_GATEWAY, params, {
    method: 'POST', // Specify the request method.
    formatParams: false, // Specify whether to format parameters and capitalize the first letter of each parameter. If this parameter is set to false
, parameters are not formatted.
    enableProxy: true, // Specify whether to configure a proxy. If this parameter is set to true, a proxy is configured and you must specify the prox
y parameters.
    proxy: 'http://x.x.x.x:xxxx',
    timeout: 10000, // Specify the timeout period.
    // ca: , // Specify the CA certificate. For more information, see https://nodejs.org/api/https.html#https_https_request_options_callback.
  })
  console.log(res)
} catch (e) {
  console.log(e)
}
```

Disable certificate validation

The following code provides an example on how to disable certificate validation:

```
// Specify environment variables.
NODE_TLS_REJECT_UNAUTHORIZED=0
```

Headers

Header	Description
x-acs-regionid	The ID of the region. For more information, see Obtain a region ID.
x-acs-organizationid	The ID of the organization in the Apsara Uni-manager Management Console. For more information, see Obtain an organization ID.
x-acs-resourcegroupid	The ID of the resource set in the Apsara Uni-manager Management Console. For more information, see Obtain a resource set ID.
x-acs-instanceid	The name of the bucket on which the operation is performed. For more information, see Obtain a bucket name.
x-acs-caller-sdk-source	The custom name of the requester or the system. This parameter is used to identify the source of the API request.
x-acs-roleid	The ID of the role.

Header	Description
x-acs-userid	The ID of the user.

3.3. HTTP requests

3.3.1. Overview

This topic describes how to make API requests and is intended for users who use API URLs to initiate HTTP GET requests.

If you are using an SDK to initiate requests, you can skip this topic.

The request URL consists of different parameters and has a fixed syntax. The URL contains common request parameters, your signature, and operation-specific parameters. Sample request URLs are provided for each API operation. These URLs are not encoded to make them easy to read. Before you make a request, you must encode the request URL. After the authentication process is complete based on your signature, the results are returned to you. Response parameters are returned if the call is successful, whereas an error message is returned if the call fails. You can troubleshoot issues based on the error message.

3.3.2. Request syntax

ASAPI allows you to initiate HTTP GET requests based on URLs. Each URL must contain request parameters. This topic describes the syntax of HTTP GET requests and provides the endpoint of ASAPI.

Sample request

The following example shows an unencoded URL that is used to call the PutBucket operation:

```
https://public.asapi.aliyuns.com/asapi/v3?Action=DoOpenApi&Product=OneRouter&Version=2018-12-12&ProductName=oss&OpenApiAction=PutBucket&Id=<Organization ID>&<Common request parameters>
```

- `https` specifies the protocol that is used to send the request.
- `public.asapi.aliyuns.com/asapi/v3` specifies the endpoint of ASAPI.
- `Action=DoOpenApi&Product=OneRouter&Version=2018-12-12&ProductName=oss&OpenApiAction=PutBucket` specifies information about the API operation that you want to call. OpenApiAction specifies the name of the OSS API operation.
- `Id` specifies the ID of the organization.
- `<Common request parameters>` specifies the common parameters specified in the system.

Protocols

ASAPI supports only HTTPS.

Endpoints

For more information about how to generate the endpoint of ASAPI, see Generate an endpoint for ASAPI.

You must specify parameters, such as the Action, Product, Version, ProductName, and OpenApiAction parameters, to specify the API operation that you want to call, such as `Action=DoOpenApi&Product=OneRouter&Version=2018-12-12&ProductName=oss&OpenApiAction=PutBucket`. You must also specify other operation-specific request parameters and common request parameters. For more information, see Common request parameters.

3.3.3. Request signatures

This topic describes how to sign HTTP API requests. Alibaba Cloud uses the request signature to verify the identity of the requester. Specifically, Alibaba Cloud uses an AccessKey pair to perform symmetric encryption and verify the identity of the requester.

> **Note**
> - An AccessKey pair consists of an AccessKey ID and an AccessKey secret. The AccessKey pair is issued to each user or organization in the Apsara Uni-manager Management Console.
> - The AccessKey ID is used to verify the identities of users, and the AccessKey secret is used to encrypt and verify signature strings. You must store the AccessKey secret in a secure location.
> - ASAPI provides SDKs for Node.js, Python, and Java to facilitate complex signature calculation. For more information, see Obtain Apsara Stack SDKs.

Step 1: Construct a canonicalized query string

1. Sort the request parameters in alphabetical order, including all common and operation-specific parameters except for `Signature`.

   > **Note**
   > If you use the GET method to send a request, the request parameters are included as part of the request URL. The parameters follow the question mark (?) and are connected by ampersands (&) in the URL.

2. Encode the names and values of the request parameters in UTF-8 based on RFC 3986.
   - Letters, digits, and special characters such as hyphens (`-`), underscores (`_`), and periods (`.`) do not need to be encoded.

- Other characters must be percent-encoded in the `%XY` format. `XY` represents the ASCII code of the characters in hexadecimal notation. For example, double quotation marks (`"`) are encoded as `%22` .
- Extended UTF-8 characters are encoded in the `%XY%ZA…` format.
- Spaces must be encoded as `%20` . Do not encode spaces as plus signs (`+`).

  The preceding encoding method is similar to but slightly different from the `application/x-www-form-urlencoded` MIME encoding algorithm.

  If you use `java.net.URLEncoder` in the Java standard library, use `percentEncode` to encode request parameters and their values. In the encoded query string, replace plus signs (`+`) with `%20` , asterisks (`*`) with `%2A` , and `%7E` with tildes (`~`). This way, you can obtain an encoded string that matches the preceding encoding rules.

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws UnsupportedEncodingException {
    return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*", "%2A").replace("%7E", "~") : null;
}
```

3. Use equal signs (`=`) to connect the name and value of an encoded parameter.
4. Use ampersands (`&`) to connect the encoded request parameters. These parameters must be arranged in the same order as those in Step 1.

Then, a canonicalized query string that follows the request structure is obtained. For more information, see Request structure.

Step 2: Create a string-to-sign

1. Create a string-to-sign (`StringToSign`). You can also use `percentEncode` to encode the canonicalized query string that is generated in the Step 1. When you create a string-to-sign, take note of the following rules:

```
StringToSign=
    HTTPMethod + "&" + //HTTPMethod: The HTTP method that is used to send the request, such as GET.
    percentEncode("/") + "&" + //percentEncode("/"): Encode backslashes (/) as %2F by using UTF-8.
    percentEncode(CanonicalizedQueryString) // Encode the canonicalized query string that is created in Step 1.
```

2. Calculate the HMAC-SHA1 or HMAC-256 value of `StringToSign` based on RFC 2104. In this example, the Java Base64 encoding method is used.

```
String signature = Base64.getEncoder().encodeToString(signData);
```

> **Note**
>
> When you calculate the signature, the key value that is specified by RFC 2104 is your `AccessKey secret` plus an ampersand (&), which has an ASCII value of 38. For more information, see Obtain an AccessKey pair.

3. Encode the `Signature` parameter based on RFC 3986 and add the parameter to the canonicalized query string.

Example 1: Concatenate parameters

The following example shows how to call the `PutBucket` operation to create a bucket. In this example, the value of AccessKeyID is `testid` and the value of AccessKeySecret is `testsecret` .

1. Construct a canonicalized query string.

```
https://public.asapi.aliyuns.com/asapi/v3?AccessKeyId=testid&Action=DoOpenApi&BucketName=examplebucket&&Format=JSON&Id=3&OpenApiAction=PutBucket&Product=OneRouter&ProductName=oss&RegionId=cn-neimeng-env141-d01&SignatureMethod=HMAC-SHA1&SignatureNonce=43967c6e-e3ad-44fa-b06b-fcf3db5f39d2&SignatureVersion=1.0&Timestamp=2022-01-11T08%3A50%3A40Z&Version=2018-12-12
```

2. Create a string-to-sign (`StringToSign`).

```
POST&%2F&AccessKeyId%3Dtestid%26Action%3DDoOpenApi%26BucketName%3Dexamplebucket%26%26Format%3DJSON%26Id%3D3%26OpenApiAction%3DPutBucket%26Product%3DOneRouter%26ProductName%3Doss%26RegionId%3Dcn-neimeng-env141-d01%26SignatureMethod%3DHMAC-SHA1%26SignatureNonce%3D43967c6e-e3ad-44fa-b06b-fcf3db5f39d2%26SignatureVersion%3D1.0%26Timestamp%3D2022-01-11T08%253A50%253A40Z%26Version%3D2018-12-12
```

3. Calculate the signature.

   Calculate the signature. As `AccessKeySecret` is set to testsecret, `testsecret&` is used as the key value to calculate the signature. The calculated signature is `OLeaidS1JvxuMvnyHOwuJ+uX5qY=` . In this example, the Java Base64 encoding method is used.

   `String signature = Base64.getEncoder().encodeToString(signData);` Encode `Signature=KankeY6w8gp/yWQfoeaD+LZZ25Q=` based on RFC 3986 and add it to the canonicalized query string.

```
https://public.asapi.aliyuns.com/asapi/v3?AccessKeyId=testid&Action=DoOpenApi&BucketName=examplebucket&Format=JSON&Id=3&OpenApiAction=PutBucket&Product=OneRouter&ProductName=oss&RegionId=cn-neimeng-env141-d01&SignatureMethod=HMAC-SHA1&SignatureNonce=43967c6e-e3ad-44fa-b06b-fcf3db5f39d2&SignatureVersion=1.0&Timestamp=2022-01-11T08%3A50%3A40Z&Version=2018-12-12&Signature=KankeY6w8gp/yWQfoeaD+LZZ25Q=
```

You can use the new URL in a browser, in cURL, or in Wget to initiate an HTTP request that calls the PutBucket operation to create a bucket named examplebucket.

Example 2: Use the standard library of the programming language

The following example shows how to call the `PutBucket` operation to create a bucket. In this example, `AccessKeyID` is set to testid, `AccessKeySecret` is set to testsecret, and all request parameters are stored in a Java `Map<String, String>` object.

1. Predefine the encoding method.

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws UnsupportedEncodingException {
        return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*", "%2A").replace("%7E", "~") : null;
}
```

2. Predefine the time format for the `Timestamp` parameter. Specify the `Timestamp` parameter in the ISO 8601 format. The time must be in UTC+0.

```
private static final String ISO8601_DATE_FORMAT = "yyyy-MM-dd'T'HH:mm:ss'Z'";
private static String formatIso8601Date(Date date) {
    SimpleDateFormat df = new SimpleDateFormat(ISO8601_DATE_FORMAT);
    df.setTimeZone(new SimpleTimeZone(0, "GMT"));
    return df.format(date);
}
```

3. Create a query string.

```
final String HTTP_METHOD = "POST";
Map parameters = new HashMap();
// Specify request parameters.
parameters.put("Product", "OneRouter");
parameters.put("Action", "DoOpenApi");
parameters.put("Version", "2018-12-12");
parameters.put("AccessKeyId", "testid");
parameters.put("RegionId", "cn-region-test");
parameters.put("Timestamp", formatIso8601Date(new Date()));
parameters.put("SignatureMethod", "HMAC-SHA1");
parameters.put("SignatureVersion", "1.0");
parameters.put("SignatureNonce", UUID.randomUUID().toString());
parameters.put("Format", "JSON");
parameters.put("Id", "1");
JSONObject params = new JSONObject();
// Specify the following parameters for this API call. The parameters are case-sensitive.

params.put("BucketName", "asapibucket");
parameters.put("Params", params.toJSONString());
// Sort the request parameters.
String[] sortedKeys = parameters.keySet().toArray(new String[]{});
Arrays.sort(sortedKeys);
final String SEPARATOR = "&";
// Create a string-to-sign.
StringBuilder stringToSign = new StringBuilder();
stringToSign.append(HTTP_METHOD).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
StringBuilder canonicalizedQueryString = new StringBuilder();
for(String key : sortedKeys) {
    // Encode the keys and values of the parameters.
    canonicalizedQueryString.append("&")
    .append(percentEncode(key)).append("=")
    .append(percentEncode(parameters.get(key)));
}
// Encode the canonicalized query string.
stringToSign.append(percentEncode(
canonicalizedQueryString.toString().substring(1)));
```

4. Calculate the signature. As `AccessKeySecret` is set to testsecret, `testsecret&` is used as the key value to calculate the signature. The calculated signature is `OLeaidS1JvxuMvnyHOwuJ%2BuX5qY%3D`.

```
// The following code provides an example on how to calculate the signature.
final String ALGORITHM = "HmacSHA1";
final String ENCODING = "UTF-8";
key = "testsecret&";
Mac mac = Mac.getInstance(ALGORITHM);
mac.init(new SecretKeySpec(key.getBytes(ENCODING), ALGORITHM));
byte[] signData = mac.doFinal(stringToSign.getBytes(ENCODING));
String signature = Base64.getEncoder().encodeToString(signData);
```

   Encode the Signature parameter based on RFC 3986 and add the parameter to the URL. Then, you can obtain the following URL:

5. Use a browser or a tool such as cURL or Wget to send the HTTP request.

3.3.4. Responses

Responses are returned only in the JSON format. If a return code is greater than or equal to 200 and less than 300, a success response is returned. Otherwise, an error response is returned.

Sample success responses

If a request is successful, the request ID and operation-specific parameter values are returned. For more information, see the developer guide of the corresponding Apsara Stack service.

```
  "successResponse":true,
  "eagleEyeTraceId":"xxxxxxxxxxxxx",
  "asapiSuccess":true,
  "code":"200",
  "cost":12,
  "asapiRequestId":"xxxxxxxxxxxxx",
  "data":[
    {
      "muserId":"aliyuntest",
      "internal":false,
      "level":"0",
      "supportRegions":"cn-*",
      "personNum":100,
      "mtime":1553247962000,
      "uuid":"1",
      "parentId":0,
      "supportRegionList":[
        "cn-*"
      ],
      "name":"root",
      "alias":"root",
      "ctime":1553247962000,
      "id":1,
      "resourceGroupNum":100,
      "cuserId":"aliyuntest"
    }
  ],
  "requestId":"xxxxxxxxxxxxx",
  "success":true,
  "message":"success"
}
```


Sample error responses

If a request fails, the following information is returned. For more information, see the developer guide of the corresponding Apsara Stack service.

```
{
    "code": "asapi.server.api.notfound",// The error code.
    "eagleEyeTraceId": "xxx-xxx-xxx-xxx", // The trace ID provided by EagleEye.
    "asapiSuccess": false, // Indicates whether the request is successful.
    "hostId": "xxxx", // The ID of the host.
    "message": "Failed to ....", // The message returned.
    "asapiErrorMessage": "Failed to ....", // The error message provided by ASAPI.
    "asapiErrorCode": "asapi.server.api.notfound" // The error code provided by ASAPI.
}
```

4.SDK reference

4.1. Java SDK

4.1.1. Preface

This topic is written for OSS SDK for Java version 3.8.0.

Compatibility

- For SDKs version 3.×.×:
  - API operation: compatible
  - Namespace: compatible
- For SDKs version 2.×.×:
  - API operation: compatible
  - Namespace: compatible
- For SDKs version 1.0.×:
  - API operation: compatible
  - Namespace: incompatible. The Tablestore code in version 1.0.x is removed from version 2.0.0. Package names `com.aliyun.openservices.*` and `com.aliyun.openservices.oss.*` are replaced with `com.aliyun.oss.*`.

SDK source code and API documents

For the SDK source code, visit GitHub. For more information, visit Aliyun OSS SDK for Java V3.8.0 API.

Sample code

OSS SDK for Java provides a variety of sample code for you to reference or use. The following table describes the sample code.

Sample file	Description
GetStartedSample.java	Shows you how to perform basic upload and download operations.
SimpleGetObjectSample.java	Shows you how to download objects.
ListObjectsSample.java	Shows you how to list objects.
DeleteObjectsSample.java	Shows you how to delete multiple objects at a time.
AppendObjectSample.java	Shows you how to use append upload.
ObjectMetaSample.java	Shows you how to use object metadata.
CreateFolderSample.java	Shows you how to create a folder. For more information about OSS folders, see the topic about folders.
UploadSample.java	Shows you how to use resumable upload.
DownloadSample.java	Shows you how to use resumable download.
ImageSample.java	Shows you how to use IMG.
PostObjectSample.java	Shows you how to use PostObject. The implementation is independent of OSS SDK for Java.
GetProgressSample.java	Shows you how to use progress bars in object upload and download.
CallbackSample.java	Shows you how to perform upload callback.
CRCSample.java	Shows you how to perform CRC for object upload and download.
BucketOperationsSample.java	Shows you how to configure a bucket, including configuring ACL, lifecycle, logging, hotlink protection, and CORS.
MultipartUploadSample.java	We recommend that you use the uploadFile operation to perform resumable upload for concurrent uploads implemented by the multipart upload API operation.
ConcurrentGetObjectSample.java	We recommend that you use the downloadFile operation to perform resumable download for concurrent downloads implemented by range download.
UploadPartCopySample.java	Shows you how to perform multipart copy for large objects.

4.1.2. Installation

This topic describes how to install OSS SDK for Java.

Prerequisites

- Environment

  Use Java 1.7 or later.
- Compatibility

  Run the `java -version` command to check the Java version.

Download the SDK

- Download directly
- Download from GitHub
- Download previous versions

Install the SDK

- Method 1: (Recommended) Add dependencies to your Maven project

  To use OSS SDK for Java in Maven, you need only to add the corresponding dependencies to the pom.xml file. For example, when you use OSS SDK for Java version 3.8.0, add the following content to <dependencies>:

```
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>3.8.0</version>
</dependency>
```

- Method 2: Import the JAR package to your Eclipse project

  For example, when you use OSS SDK for Java version 3.8.0, take the following steps to install the SDK:

  i. Download the OSS SDK for Java package.
  ii. Decompress the package.
  iii. Copy the *aliyun-sdk-oss-3.8.0.jar* file and all files in the lib directory to your project.
  iv. Right-click your project name in Eclipse. Choose **Properties > Java Build Path > Add JARs**.
  v. Select all JAR files that you have copied in Step 3 and import them to Libraries.

- Method 3: Import the JAR package to your IntelliJ IDEA project

  For example, when you use OSS SDK for Java version 3.8.0, take the following steps to install the SDK:

  i. Download the OSS SDK for Java package.
  ii. Decompress the package.
  iii. Copy the *aliyun-sdk-oss-3.8.0.jar* file and all JAR files in the lib directory to your project.
  iv. In IntelliJ IDEA, select your project and choose **File > Project Structure > Modules > Dependencies > + > JARs or directories**.
  v. Select all JAR files that you have copied in Step 3 and import them to External Libraries.

4.1.3. Initialization

OSSClient serves as the Java client to manage OSS resources such as buckets and objects. To use OSS SDK for Java to initiate a request, you must initialize an OSSClient instance and modify the default configuration items of ClientConfiguration.

Create an OSSClient instance

To create an OSSClient instance, you must specify an endpoint. For more information about endpoints, see Obtain an endpoint.

To create an OSSClient instance, you must specify an AccessKey pair. For more information about how to obtain an AccessKey pair, see Obtain an AccessKey pair.

- Create an OSSClient instance based on an OSS endpoint

  The following code provides an example on how to create an OSSClient instance based on an OSS endpoint:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  You can use one or more OSSClient instances for your project. In other words, you can use multiple OSSClient instances simultaneously.

- Create an OSSClient instance based on Apsara Stack or a private region

  The following code provides an example on how to create an OSSClient instance based on Apsara Stack or a private region:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create a ClientConfiguration instance and modify the default parameters based on your requirements.
ClientBuilderConfiguration conf = new ClientBuilderConfiguration();
// Disable CNAME.
conf.setSupportCname(false);
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, conf);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Create an OSSClient instance based on an IP address

  The following code provides an example on how to create an OSSClient instance based on an IP address:

```
// In some special cases such as a private region, the IP address is used as the endpoint. In such cases, specify the actual IP address based on yo
ur requirements.
String endpoint = "http://10.10.10.10";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create a ClientConfiguration instance.
ClientBuilderConfiguration conf = new ClientBuilderConfiguration();
// By default, access from a second-level domain is disabled. Enable OSS access from a second-level domain. You must configure this parameter for O
SS SDK for Java version 2.1.2 or earlier. The versions later than OSS SDK for Java version 2.1.2 automatically detect IP addresses.
conf.setSLDEnabled(true);
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, conf);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Create an OSSClient instance based on STS

  The following code provides an example on how to create an OSSClient instance based on Security Token Service (STS):

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Configure an OSSClient instance

ClientConfiguration is a configuration class of OSSClient. You can use ClientConfiguration to configure parameters such as host proxies, connection timeouts, and the maximum number of connections. The following table describes the parameters of this operation.

Parameter	Description	Configuration method
MaxConnections	The maximum number of HTTP connections that are allowed. Default value: 1024.	ClientConfiguration.setMaxConnections
SocketTimeout	The timeout period for data transmission at the socket layer. Unit: milliseconds. Default value: 50000.	ClientConfiguration.setSocketTimeout
ConnectionTimeout	The timeout period for establishing a connection. Unit: milliseconds. Default value: 50000.	ClientConfiguration.setConnectionTimeout
ConnectionRequestTimeout	The timeout period to obtain a connection from the connection pool. Unit: milliseconds. By default, this parameter is not configured.	ClientConfiguration.setConnectionRequestTimeout
IdleConnectionTime	The idle connection timeout period to close the connection. Unit: milliseconds. Default value: 60000.	ClientConfiguration.setIdleConnectionTime
MaxErrorRetry	The maximum number of allowed retry attempts in the case of a request error. Default value: 3.	ClientConfiguration.setMaxErrorRetry
SupportCname	Specifies whether CNAME can be used as an endpoint. By default, CNAME can be used as an endpoint.	ClientConfiguration.setSupportCname
SLDEnabled	Specifies whether access over second-level domains is enabled. By default, access over second-level domains is disabled.	ClientConfiguration.setSLDEnabled
Protocol	The protocol used to connect to OSS. Default value: HTTP. Valid values: HTTP and HTTPS.	ClientConfiguration.setProtocol
UserAgent	The user agent in the HTTP header. Default value: aliyun-sdk-java.	ClientConfiguration.setUserAgent
ProxyHost	The IP address to access the proxy host.	ClientConfiguration.setProxyHost
ProxyPort	The port for the proxy host.	ClientConfiguration.setProxyPort
ProxyUsername	The username verified by the proxy host.	ClientConfiguration.setProxyUsername
ProxyPassword	The password verified by the proxy host.	ClientConfiguration.setProxyPassword

The following code provides an example on how to configure parameters for an OSSClient instance with ClientConfiguration:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create a ClientConfiguration instance. ClientConfiguration is a configuration class of OSSClient. You can use ClientConfiguration to configure par
ameters such as host proxies, connection timeouts, and the maximum number of connections.
ClientBuilderConfiguration conf = new ClientBuilderConfiguration();
// Configure the maximum number of HTTP connections allowed by an OSSClient instance. Default value: 1024.
conf.setMaxConnections(200);
// Configure the timeout period for data transmission at the socket layer. Unit: milliseconds. Default value: 50000.
conf.setSocketTimeout(10000);
// Configure the timeout period for establishing a connection. Unit: milliseconds. Default value: 50000.
conf.setConnectionTimeout(10000);
// Configure the timeout period for obtaining a connection from the connections pool. Unit: milliseconds. By default, this parameter is not configure
d.
conf.setConnectionRequestTimeout(1000);
// Configure the timeout period for idle connections. The connection is closed when the timeout period expires. Unit: milliseconds. Default value: 60
000.
conf.setIdleConnectionTime(10000);
// Configure the maximum number of allowed retry attempts in the case of a request error. Default value: 3.
conf.setMaxErrorRetry(5);
// Configure whether CNAME can be used as an endpoint. By default, CNAME can be used as an endpoint.
conf.setSupportCname(true);
// Configure whether access from second-level domains is enabled. By default, access from second-level domains is disabled.
conf.setSLDEnabled(true);
// Configure the protocol used to connect to OSS. Default value: HTTP. Valid values: HTTP and HTTPS.
conf.setProtocol(Protocol.HTTP);
// Configure the user agent in the HTTP header. Default value: aliyun-sdk-java.
conf.setUserAgent("aliyun-sdk-java");
// Configure the port for the proxy host.
conf.setProxyHost("<yourProxyHost>");
// Configure the username verified by the proxy host.
conf.setProxyUsername("<yourProxyUserName>");
// Configure the password verified by the proxy host.
conf.setProxyPassword("<yourProxyPassword>");
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, conf);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.4. Quick start

This topic describes how to use OSS SDK for Java to complete routine operations such as create buckets, upload objects, and download objects.

Sample projects

OSS SDK for Java provides Maven- and Ant-based sample projects. You can compile and run the sample projects on local devices or develop your own applications based on the sample projects. For more information about how to compile and run a project, see README.md in the project directory.

- Maven-based sample project: aliyun-oss-java-sdk-demo-mvn.zip
- Ant-based sample project: aliyun-oss-java-sdk-demo-ant.zip

Create a bucket

A bucket is a global namespace in OSS. A bucket is a container for objects stored in OSS. The following code provides an example on how to create a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a bucket.
ossClient.createBucket(bucketName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

For more information about how to obtain an AccessKey ID and AccessKey secret, see Obtain an AccessKey pair.

For more information about how to obtain an endpoint, see Obtain an endpoint.

Upload objects

The following code provides an example on how to upload an object to OSS:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Upload content to a specified bucket and save it by using the specified object name.
String content = "Hello OSS";
ossClient.putObject(bucketName, objectName, new ByteArrayInputStream(content.getBytes()));
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Download an object

The following code provides an example on how to query the content of an object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Call ossClient.getObject to query an OSSObject instance. The OSSObject instance contains the content and metadata of the object.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Call ossObject.getObjectContent to query an input stream of the object. Read the input stream to query the content of the object.
InputStream content = ossObject.getObjectContent();
if (content ! = null) {
    BufferedReader reader = new BufferedReader(new InputStreamReader(content));
    while (true) {
        String line = reader.readLine();
        if (line == null) break;
        System.out.println("\n" + line);
    }
    // You must close the connection after each use to prevent connection leaks. Otherwise, requests may be unable to establish connections and the p
rogram cannot run properly.
    content.close();
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List objects

The following code provides an example on how to list objects in a specified bucket. By default, a maximum of 100 objects are listed.

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Call ossClient.listObjects to obtain an ObjectListing instance. The ObjectListing instance contains the response to the listObject request.
ObjectListing objectListing = ossClient.listObjects(bucketName);
// Call objectListing.getObjectSummaries to query the descriptions of all objects.
for (OSSObjectSummary objectSummary : objectListing.getObjectSummaries()) {
    System.out.println(" - " + objectSummary.getKey() + "  " +
            "(size = " + objectSummary.getSize() + ")");
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete objects

For the complete code used to delete objects, visit GitHub.

The following code provides an example on how to delete an object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Delete the object.
ossClient.deleteObject(bucketName, objectName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5. Buckets

4.1.5.1. Create buckets

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to create a bucket.

The following code provides an example on how to create a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a bucket.
ossClient.createBucket(bucketName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.2. List buckets

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to list buckets.

Buckets are listed in alphabetical order. You can list all buckets or only buckets that meet specified conditions. To list buckets of the IA or Archive storage class, use OSS SDK for Java version 2.6.0 or later.

List all buckets

The following code provides an example on how to list all buckets:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List all buckets.
List<Bucket> buckets = ossClient.listBuckets();
for (Bucket bucket : buckets) {
    System.out.println(" - " + bucket.getName());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List buckets whose names contain a specified prefix

The following code provides an example on how to list buckets whose names contain a specified prefix:

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// List buckets whose names contain a specified prefix.
listBucketsRequest.setPrefix("<yourBucketPrefix>");
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

List buckets whose names come after a specified marker

The marker parameter specifies the name of the bucket after which the list begins. The following code provides an example on how to list buckets whose names come after a specified marker:

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// List buckets whose names come after the specified marker.
listBucketsRequest.setMarker("<yourBucketMarker>");
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

List a specified number of buckets

The following code provides an example on how to specify the maxKeys parameter to list a specified number of buckets:

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// Set the maximum number of buckets to list each time to 500. The default value is 100. The maximum value is 1000.
listBucketsRequest.setMaxKeys(500);
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

4.1.5.3. Determine whether buckets exist

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to determine whether a bucket exists.

The following code provides an example on how to determine whether a specific bucket exists:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account to log on to OSS because the account has permissions on all API
operations. We recommend that you use a RAM user to call API operations or perform routine operations and maintenance. To create a RAM user, log on t
o https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
boolean exists = ossClient.doesBucketExist("<yourBucketName>");
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.4. Manage bucket ACLs

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to configure and obtain the access control list (ACL) of a bucket.

Configure ACL for a bucket

The following table describes the ACLs for buckets.

ACL	Description	Value
Private	Only the owner or authorized users of the bucket can read and write the object.	CannedAccessControlList.Private
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	CannedAccessControlList.PublicRead
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	CannedAccessControlList.PublicReadWrite

The following code provides an example on how to configure ACL for a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Set the ACL of the bucket to private.
ossClient.setBucketAcl("<yourBucketName>", CannedAccessControlList.Private);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Query the ACL of a bucket

The following code provides an example on how to query the ACL of a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Query the ACL of the bucket.
AccessControlList acl = ossClient.getBucketAcl("<yourBucketName>");
System.out.println(acl.toString());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.5. Delete buckets

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to delete a bucket.

> **Note** Before you delete a bucket, you must delete all objects and parts generated by multipart upload in the bucket.

The following code provides an example on how to delete a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Delete the bucket.
ossClient.deleteBucket("<yourBucketName>");
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.6. Manage lifecycle rules

You can configure lifecycle rules for an OSS bucket to automatically delete expired objects to save storage space. This topic describes how to manage lifecycle rules.

Background

Each lifecycle rule contains the following information:

- Policy: specifies the mode to match objects and parts.
  - Match by prefix: matches objects and parts by prefix. You can create multiple rules to configure different prefixes. Each prefix must be unique.
  - Match by bucket: matches all objects and parts contained in the bucket. You can create only one rule if you select this method.
- Object lifecycle policy: specifies the validity period or expiration data and the operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired objects are deleted.
  - Expiration date: specifies a date. Objects that are last modified before this date expire and are deleted.
- Part lifecycle policy: specifies the validity period or expiration time and the delete operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted.
  - Expiration date: specifies a date. Parts that are last modified before this date expire and are deleted.

Lifecycle rules also apply to the parts uploaded by using uploadPart. In this case, the last modified time of an object is the time the multipart upload task is initiated.

Configure lifecycle rules

The following code provides an example on how to configure lifecycle rules:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
SetBucketLifecycleRequest request = new SetBucketLifecycleRequest(bucketName);
// Set the rule ID and the prefix of object names to match the rule.
String ruleId0 = "rule0";
String matchPrefix0 = "A0/";
String ruleId1 = "rule1";
String matchPrefix1 = "A1/";
String ruleId2 = "rule2";
String matchPrefix2 = "A2/";
String ruleId3 = "rule3";
String matchPrefix3 = "A3/";
// Specify that objects expire three days after they are last modified.
request.AddLifecycleRule(new LifecycleRule(ruleId0, matchPrefix0, RuleStatus.Enabled, 3));
// Specify that the objects that are last modified before the specified date expire.
LifecycleRule rule = new LifecycleRule(ruleId1, matchPrefix1, RuleStatus.Enabled);
rule.setCreatedBeforeDate(DateUtil.parseISO8601Date("2022-10-12T00:00:00.000Z"));
request.AddLifecycleRule(rule);
// Specify that parts expire three days after they are last modified.
rule = new LifecycleRule(ruleId2, matchPrefix2, RuleStatus.Enabled);
LifecycleRule.AbortMultipartUpload abortMultipartUpload = new LifecycleRule.AbortMultipartUpload();
abortMultipartUpload.setExpirationDays(3);
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);
// Specify that the parts that are last modified before the specified date expire.
rule = new LifecycleRule(ruleId3, matchPrefix3, RuleStatus.Enabled);
abortMultipartUpload = new LifecycleRule.AbortMultipartUpload();
abortMultipartUpload.setCreatedBeforeDate(DateUtil.parseISO8601Date("2022-10-12T00:00:00.000Z"));
rule.setAbortMultipartUpload(abortMultipartUpload);
request.AddLifecycleRule(rule);
ossClient.setBucketLifecycle(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

View lifecycle rules

The following code provides an example on how to view lifecycle rules:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
List<LifecycleRule> rules = ossClient.getBucketLifecycle(bucketName);
for (LifecycleRule rule : rules) {
    System.out.println(rule.getId());
    System.out.println(rule.getPrefix());
    System.out.println(rule.getExpirationDays());
    System.out.println(rule.getCreatedBeforeDate());
    if(rule.hasAbortMultipartUpload()) {
        System.out.println(rule.getAbortMultipartUpload().getExpirationDays());
        System.out.println(rule.getAbortMultipartUpload().getCreatedBeforeDate());
    }
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete lifecycle rules

The following code provides an example on how to delete the lifecycle rules configured for a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
ossClient.deleteBucketLifecycle(bucketName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.7. Configure hotlink protection

OSS provides hotlink protection to prevent unauthorized domain names from accessing your OSS data. You can configure the Referer field in the HTTP header.

To configure hotlink protection, you must configure the following parameters:

- Referer Whitelist: Only specified domain names are allowed to access OSS resources.
- Allow Empty Referer: If you do not allow empty Refer fields, a request is allowed to access OSS resources only if the request includes the Referer field configured in the HTTP or HTTPS header.

Configure a Referer whitelist

The following code provides an example on how to configure the Referer whitelist for a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
List<String> refererList = new ArrayList<String>();
// Add Referer values to the whitelist. You can use an asterisk (*) or a question mark (?) as a wildcard to set the Referer parameter.
refererList.add("http://www.aliyun.com");
refererList.add("http://www. *.com");
refererList.add("http://www.?.aliyuncs.com");
// Configure BucketReferer. A value of true indicates that an empty Referer field is allowed.
BucketReferer br = new BucketReferer(true, refererList);
ossClient.setBucketReferer(bucketName, br);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Query the hotlink configurations of a bucket

The following code provides an example on how to query the hotlink configurations of a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Query the Referer whitelist of the bucket.
BucketReferer br = ossClient.getBucketReferer(bucketName);
List<String> refererList = br.getRefererList();
for (String referer : refererList) {
    System.out.println(referer);
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete the hotlink configurations of a bucket

The following code provides an example on how to delete the hotlink configurations of a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// You cannot delete a Referer whitelist directly. To delete a Referer whitelist, you must create a rule that allows an empty Referer field to replac
e the existing rule.
BucketReferer br = new BucketReferer();
ossClient.setBucketReferer(bucketName, br);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.8. Configure CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to different regions. OSS provides CORS operations to facilitate cross-origin access control.

Configure CORS rules

The following code provides an example on how to configure CORS rules for a specific bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
SetBucketCORSRequest request = new SetBucketCORSRequest(bucketName);
// Create a container for the CORS rules. A maximum of 10 CORS rules can be configured for each bucket.
ArrayList<CORSRule> putCorsRules = new ArrayList<CORSRule>();
CORSRule corRule = new CORSRule();
ArrayList<String> allowedOrigin = new ArrayList<String>();
// Specify the allowed origins from which CORS requests are sent.
allowedOrigin.add( "http://www.b.com");
ArrayList<String> allowedMethod = new ArrayList<String>();
// Specify the HTTP methods that CORS requests are allowed to use, including GET, PUT, DELETE, POST, and HEAD.
allowedMethod.add("GET");
ArrayList<String> allowedHeader = new ArrayList<String>();
// Determine whether to allow the header specified in Access-Control-Request-Headers of the preflight (OPTIONS) request.
allowedHeader.add("x-oss-test");
ArrayList<String> exposedHeader = new ArrayList<String>();
// Specify the response headers that you are allowed to access from applications.
exposedHeader.add("x-oss-test1");
// You can use only one asterisk (*) as the wildcard for AllowedOrigins and AllowedMethods in a CORS rule. Asterisks (*) indicate to allow all origin
s or operations.
corRule.setAllowedMethods(allowedMethod);
corRule.setAllowedOrigins(allowedOrigin);
// AllowedHeaders and ExposeHeaders do not support wildcards.
corRule.setAllowedHeaders(allowedHeader);
corRule.setExposeHeaders(exposedHeader);
// Specifies the time the browser can cache the response to a preflight (OPTIONS) request to a specific resource. Unit: seconds.
corRule.setMaxAgeSeconds(10);
// You can configure up to 10 CORS rules for the bucket.
putCorsRules.add(corRule);
// The existing rules are replaced.
request.setCorsRules(putCorsRules);
ossClient.setBucketCORS(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Query CORS rules

The following code provides an example on how to query CORS rules configured for a specific bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
ArrayList<CORSRule> corsRules;
// Query the CORS rule list.
corsRules =  (ArrayList<CORSRule>) ossClient.getBucketCORSRules(bucketName);
for (CORSRule rule : corsRules) {
    for (String allowedOrigin1 : rule.getAllowedOrigins()) {
        // Query the allowed origins from which CORS requests are sent.
        System.out.println(allowedOrigin1);
    }
    for (String allowedMethod1 : rule.getAllowedMethods()) {
        // Query the allowed cross-origin request methods.
        System.out.println(allowedMethod1);
    }
    if (rule.getAllowedHeaders().size() > 0){
        for (String allowedHeader1 : rule.getAllowedHeaders()) {
            // Query the list of headers allowed in cross-origin requests.
            System.out.println(allowedHeader1);
        }
    }
    if (rule.getExposeHeaders().size() > 0) {
        for (String exposeHeader : rule.getExposeHeaders()) {
            // Query the list of response headers that can be accessed from an application.
            System.out.println(exposeHeader);
        }
    }
    if ( null ! = rule.getMaxAgeSeconds()) {
        System.out.println(rule.getMaxAgeSeconds());
    }
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete CORS rules

The following code provides an example on how to delete the CORS rules configured for a specific bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Delete the CORS rules configured for the bucket.
ossClient.deleteBucketCORSRules(bucketName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.9. Configure logging

You can enable logging to record access to a bucket in log objects, which are stored in a specified bucket.

The log objects are in the following format: `<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString`.

Enable logging

The following code provides an example on how to enable logging for a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
SetBucketLoggingRequest request = new SetBucketLoggingRequest("<yourSourceBucketName>");
// Specify the bucket used to store log objects.
request.setTargetBucket("<yourTargetBucketName>");
// Specify the folder where log objects are stored.
request.setTargetPrefix("<yourTargetPrefix>");
ossClient.setBucketLogging(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

View logging configurations

The following code provides an example on how to view the logging configurations of a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
BucketLoggingResult result = ossClient.getBucketLogging("<yourSourceBucketName>");
System.out.println(result.getTargetBucket());
System.out.println(result.getTargetPrefix());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Disable logging

The following code provides an example on how to disable logging for a bucket:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
SetBucketLoggingRequest request = new SetBucketLoggingRequest("<yourSourceBucketName>");
request.setTargetBucket(null);
request.setTargetPrefix(null);
ossClient.setBucketLogging(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.5.10. Configure static website hosting

This topic describes how to set your bucket to the static website hosting mode. After the configurations take effect, you can use the bucket domain name to access this static website and be redirected to a specified index page or error page.

Configure static website hosting

The following code provides an example on how to configure static website hosting:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
SetBucketWebsiteRequest request = new SetBucketWebsiteRequest("<yourBucketName>");
request.setIndexDocument("index.html");
request.setErrorDocument("error.html");
ossClient.setBucketWebsite(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

View static website hosting configurations

The following code provides an example on how to view static website hosting configurations:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
BucketWebsiteResult result = ossClient.getBucketWebsite("<yourBucketName>");
System.out.println(result.getIndexDocument());
System.out.println(result.getErrorDocument());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete static website hosting configurations

The following code provides an example on how to delete static website hosting configurations:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
ossClient.deleteBucketWebsite("<yourBucketName>");
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.6. Upload objects

4.1.6.1. Overview

This topic describes a variety of methods OSS SDK for Java provides to upload objects.

Objects are the basic unit for data operations in OSS. OSS SDK for Java provides the following object upload methods:

- Simple upload: includes streaming upload and local file upload. You can use this method to upload an object up to 5 GB in size.
- Form upload: uploads an object up to 5 GB in size.
- Append upload: uploads an object up to 5 GB in size.
- Resumable upload: supports concurrent upload and resumable upload in which you can define the size of each part. This method is suitable for the upload of large objects. You can use this method to upload an object up to 48.8 TB in size.
- Multipart upload: supports the upload of an object up to 48.8 TB in size. This method is suitable for the upload of large objects.

During object upload, you can configure object metadata and view upload progress by using a progress bar. After the object is uploaded, you can perform upload callback.

4.1.6.2. Simple upload

Simple upload allows you to use the PutObject operation to upload a single object to OSS. This topic describes how to perform simple upload by using OSS SDK for Java.

Simple upload includes stream upload and file upload. Stream upload uses InputStream as the source object. File upload uses a local file as the source object.

> **Note** For the complete code used to perform simple upload, visit GitHub.

Stream upload

Use ossClient.putObject to upload a data stream to OSS.

- Upload a string

  The following code provides an example on how to upload a string:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a PutObjectRequest object.
String content = "Hello OSS";
// <yourObjectName> indicates the complete path of the object you want to upload to OSS. The path must include the extension of the object name. Ex
ample: abc/efg/123.jpg.
PutObjectRequest putObjectRequest = new PutObjectRequest("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()));
// Upload the string.
ossClient.putObject(putObjectRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Upload a byte array

  The following code provides an example on how to upload a byte array:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId,accessKeySecret);
// Upload the byte array.
byte[] content = "Hello OSS".getBytes();
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content));
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Upload a network stream

  The following code provides an example on how to upload a network stream:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Upload the network stream.
InputStream inputStream = new URL("https://www.aliyun.com/").openStream();
ossClient.putObject("<yourBucketName>", "<yourObjectName>", inputStream);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Upload a file stream

  The following code provides an example on how to upload a file stream:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Upload the file stream.
InputStream inputStream = new FileInputStream("<yourlocalFile>");
ossClient.putObject("<yourBucketName>", "<yourObjectName>", inputStream);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Local file upload

The following code provides an example on how to upload a local file to OSS:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a PutObjectRequest object.
PutObjectRequest putObjectRequest = new PutObjectRequest("<yourBucketName>", "<yourObjectName>", new File("<yourLocalFile>"));
// Upload the local file.
ossClient.putObject(putObjectRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.6.3. Form upload

Form upload uses an HTML form to upload an object to a specified bucket. The maximum size of an object to upload is 5 GB.

The following code provides an example on how to implement form upload:

```
public class PostObjectSample {
```

```
    // Upload an object.
    private String localFilePath = "<yourLocalFile>";
    // The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
    private String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    // Security risks may arise if you log on with the AccessKey pair of an Alibaba Cloud account because the account has permissions on all API oper
ations. We recommend that you use a RAM user's credentials to call API operations or perform routine operations and maintenance. To create a RAM user
, log on to https://ram.console.aliyun.com.
    private String accessKeyId = "<yourAccessKeyId>";
    private String accessKeySecret = "<yourAccessKeySecret>";
    // Specify the bucket name.
    private String bucketName = "<yourBucketName>";
    // Specify the object name.
    private String objectName = "<yourObjectName>";
    /**
     // Start form upload.
     * @throws Exception
     */
    private void PostObject() throws Exception {
        // Add the bucket name to the URL. The URL format is http://yourBucketName.oss-cn-hangzhou.aliyuncs.com.
        String urlStr = endpoint.replace("http://", "http://" + bucketName+ ".") ;
        // Create a map for the form.
        Map<String, String> formFields = new LinkedHashMap<String, String>();
        // Specify the object name.
        formFields.put("key", this.objectName);
        // Configure Content-Disposition.
        formFields.put("Content-Disposition", "attachment;filename="
                + localFilePath);
        // Configure callback parameters.
        Callback callback = new Callback();
        // Configure the URL of the server to which you want to send the callback request. Example: http://oss-demo.aliyuncs.com:23450 or http://127.
0.0.1:9090.
        callback.setCallbackUrl("<yourCallbackServerUrl>");
        // Configure the Host header in the callback request. Example: oss-cn-hangzhou.aliyuncs.com.
        callback.setCallbackHost("<yourCallbackServerHost>");
        // Configure the body field for the callback request.
        callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":${size}}");
        // Configure Content-Type for the callback request.
        callback.setCalbackBodyType(CalbackBodyType.JSON);
        // Configure the custom parameters used to send the callback request. Each custom parameter consists of a key-value pair. The key must start
with x: and be in lower case.
        callback.addCallbackVar("x:var1", "value1");
        callback.addCallbackVar("x:var2", "value2");
        // Configure the callback parameter in Map for the form.
        setCallBack(formFields, callback);
        // Set OSSAccessKeyId.
        formFields.put("OSSAccessKeyId", accessKeyId);
        String policy = "{\"expiration\": \"2120-01-01T12:00:00.000Z\",\"conditions\": [[\"content-length-range\", 0, 104857600]]}";
        String encodePolicy = new String(Base64.encodeBase64(policy.getBytes()));
        // Set the policy.
        formFields.put("policy", encodePolicy);
        // Set a signature.
        String signaturecom = com.aliyun.oss.common.auth.ServiceSignature.create().computeSignature(accessKeySecret, encodePolicy);
        // Set the signature.
        formFields.put("Signature", signaturecom);
        String ret = formUpload(urlStr, formFields, localFilePath);
        System.out.println("Post Object [" + this.objectName + "] to bucket [" + bucketName + "]");
        System.out.println("post reponse:" + ret);
    }
    private static String formUpload(String urlStr, Map<String, String> formFields, String localFile)
            throws Exception {
        String res = "";
        HttpURLConnection conn = null;
        String boundary = "9431149156168";
        try {
            URL url = new URL(urlStr);
            conn = (HttpURLConnection) url.openConnection();
            conn.setConnectTimeout(5000);
            conn.setReadTimeout(30000);
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setRequestMethod("POST");
            conn.setRequestProperty("User-Agent",
                    "Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2.6)");
            // Set the MD5 hash. The MD5 hash is calculated based on the entire body.
            conn.setRequestProperty("Content-MD5", "<yourContentMD5>");
            conn.setRequestProperty("Content-Type",
                    "multipart/form-data; boundary=" + boundary);
            OutputStream out = new DataOutputStream(conn.getOutputStream());
            // Recursively read all Map data in the form and write the data to the output stream.
            if (formFields ! = null) {
                StringBuffer strBuf = new StringBuffer();
                Iterator<Entry<String, String>> iter = formFields.entrySet().iterator();
                int i = 0;
                while (iter.hasNext()) {
                    Entry<String, String> entry = iter.next();
                    String inputName = entry.getKey();
                    String inputValue = entry.getValue();
```

```
                    if (inputValue == null) {
                        continue;
                    }
                    if (i == 0) {
                        strBuf.append("--").append(boundary).append("\r\n");
                        strBuf.append("Content-Disposition: form-data; name=\""
                                + inputName + "\"\r\n\r\n");
                        strBuf.append(inputValue);
                    } else {
                        strBuf.append("\r\n").append("--").append(boundary).append("\r\n");
                        strBuf.append("Content-Disposition: form-data; name=\""
                                + inputName + "\"\r\n\r\n");
                        strBuf.append(inputValue);
                    }
                    i++;
                }
                out.write(strBuf.toString().getBytes());
            }
            // Read the object and write it to the output stream.
            File file = new File(localFile);
            String filename = file.getName();
            String contentType = new MimetypesFileTypeMap().getContentType(file);
            if (contentType == null || contentType.equals("")) {
                contentType = "application/octet-stream";
            }
            StringBuffer strBuf = new StringBuffer();
            strBuf.append("\r\n").append("--").append(boundary)
                    .append("\r\n");
            strBuf.append("Content-Disposition: form-data; name=\"file\"; "
                    + "filename=\"" + filename + "\"\r\n");
            strBuf.append("Content-Type: " + contentType + "\r\n\r\n");
            out.write(strBuf.toString().getBytes());
            DataInputStream in = new DataInputStream(new FileInputStream(file));
            int bytes = 0;
            byte[] bufferOut = new byte[1024];
            while ((bytes = in.read(bufferOut)) != -1) {
                out.write(bufferOut, 0, bytes);
            }
            in.close();
            byte[] endData = ("\r\n--" + boundary + "--\r\n").getBytes();
            out.write(endData);
            out.flush();
            out.close();
            // Read the returned data.
            strBuf = new StringBuffer();
            BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            String line = null;
            while ((line = reader.readLine()) != null) {
                strBuf.append(line).append("\n");
            }
            res = strBuf.toString();
            reader.close();
            reader = null;
        } catch (Exception e) {
            System.err.println("Send post request exception: " + e);
            throw e;
        } finally {
            if (conn != null) {
                conn.disconnect();
                conn = null;
            }
        }
        return res;
    }
     private static void setCallBack(Map<String, String> formFields, Callback callback) {
        if (callback != null) {
            String jsonCb = OSSUtils.jsonizeCallback(callback);
            String base64Cb = BinaryUtil.toBase64String(jsonCb.getBytes());
            formFields.put("callback", base64Cb);
            if (callback.hasCallbackVar()) {
                Map<String, String> varMap = callback.getCallbackVar();
                for (Entry<String, String> entry : varMap.entrySet()) {
                    formFields.put(entry.getKey(), entry.getValue());
                }
            }
        }
    }
    public static void main(String[] args) throws Exception {
        PostObjectSample ossPostObject = new PostObjectSample();
        ossPostObject.PostObject();
    }
}
```

4.1.6.4. Append upload

OSS allows you to use append object to append content directly to the end of an object. This method applies to append objects.

You cannot perform copyObject for append objects.

The following code provides an example on how to perform append upload:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Security risks may arise if you log on with the AccessKey pair of an Alibaba Cloud account because the account has permissions on all API operatio
ns. We recommend that you use a RAM user's credentials to call API operations or perform routine operations and maintenance. To create a RAM user, lo
g on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String content1 = "Hello OSS A \n";
String content2 = "Hello OSS B \n";
String content3 = "Hello OSS C \n";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
ObjectMetadata meta = new ObjectMetadata();
// Specify the type of content you want to upload.
meta.setContentType("text/plain");
// Configure parameters by using AppendObjectRequest.
AppendObjectRequest appendObjectRequest = new AppendObjectRequest("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content1.getBytes(
)),meta);
// Configure a single parameter by using AppendObjectRequest.
// Specify the bucket name.
//appendObjectRequest.setBucketName("<yourBucketName>");
// Specify the object name.
//appendObjectRequest.setKey("<yourObjectName>");
// Specify the content you want to append. Two types of content are available: InputStream and File. Select InputStream.
//appendObjectRequest.setInputStream(new ByteArrayInputStream(content1.getBytes()));
// Specify the content you want to append. Two types of content are available: InputStream and File. Select File.
//appendObjectRequest.setFile(new File("<yourLocalFile>"));
// Specify object metadata. The specified object metadata takes effect from the first append.
//appendObjectRequest.setMetadata(meta);
// Start the first append.
// Configure the position where the object is appended based on the previous object length.
appendObjectRequest.setPosition(0L);
AppendObjectResult appendObjectResult = ossClient.appendObject(appendObjectRequest);
// Calculate the 64-bit CRC value. The value is calculated based on the ECMA-182 standard.
System.out.println(appendObjectResult.getObjectCRC());
// Start the second append.
// nextPosition indicates the position included in the next request, which is equal to the length of the current object.
appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
appendObjectRequest.setInputStream(new ByteArrayInputStream(content2.getBytes()));
appendObjectResult = ossClient.appendObject(appendObjectRequest);
// Start the third append.
appendObjectRequest.setPosition(appendObjectResult.getNextPosition());
appendObjectRequest.setInputStream(new ByteArrayInputStream(content3.getBytes()));
appendObjectResult = ossClient.appendObject(appendObjectRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.6.5. Resumable upload

If the system stops responding during multipart upload, you can resume the upload by using the ListMultipartUploads and ListParts operations to retrieve all multipart upload tasks on an object and list the uploaded parts in each task. This method allows an upload task to be resumed from the last uploaded part.

The following code provides an example on how to perform resumable upload:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
ObjectMetadata meta = new ObjectMetadata();
// Specify the type of content you want to upload.
meta.setContentType("text/plain");
// Configure parameters by using UploadFileRequest.
UploadFileRequest uploadFileRequest = new UploadFileRequest("<yourBucketName>","<yourObjectName>");
// Configure a single parameter by using UploadFileRequest.
// Specify the bucket name.
//uploadFileRequest.setBucketName("<yourBucketName>");
// Specify the object name.
//uploadFileRequest.setKey("<yourObjectName>");
// Specify the local file to upload.
uploadFileRequest.setUploadFile("<yourLocalFile>");
// Specify the number of threads for simultaneous upload. Default value: 1.
uploadFileRequest.setTaskNum(5);
// Specify the size of a part you want to upload. Valid values: 100 KB to 5 GB. The default size of each part is calculated based on the formula: Def
ault size of each part = Object size/10000.
uploadFileRequest.setPartSize(1 * 1024 * 1024);
// Enable resumable upload. By default, resumable upload is disabled.
uploadFileRequest.setEnableCheckpoint(true);
// Specify the checkpoint file that records the upload results of each part. You need to configure this parameter when you enable resumable upload. T
his file stores progress information. If a part fails to be uploaded, you can continue the upload based on the progress information recorded in the c
heckpoint file. After the object is uploaded, the checkpoint file is deleted. By default, this file is stored in uploadFile.ucp, which is the same di
rectory as that of the local file you want to upload.
uploadFileRequest.setCheckpointFile("<yourCheckpointFile>");
// Configure object metadata.
uploadFileRequest.setObjectMetadata(meta);
// Configure upload callback. The parameter type is Callback.
uploadFileRequest.setCallback("<yourCallbackEvent>");
// Start resumable upload.
ossClient.uploadFile(uploadFileRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.6.6. Multipart upload

OSS allows you to use multipart upload to split an object into several parts and upload them separately. After all parts are uploaded, you can combine the uploaded parts into a complete object to finish the upload.

To implement multipart upload, perform the following operations:

1. Initiate a multipart upload task.

   Call the ossClient.initiateMultipartUpload method to obtain a unique upload ID in OSS.

2. Upload the parts.

   Call the ossClient.uploadPart method to upload the parts.

   > **Note**
   > - Part numbers identify the relative positions of parts in an object that share the same upload ID. If you have uploaded a part and used its part number again to upload another part, the latter part overwrites the former part.
   > - OSS includes the MD5 hash of part data in the ETag header and returns the MD5 hash to the user.
   > - OSS calculates the MD5 hash of uploaded data and compares it with the MD5 hash calculated by the SDK. If the two hashes are different, the InvalidDigest error code is returned.

3. Complete the multipart upload task.

   After all parts are uploaded, call the ossClient.completeMultipartUpload method to combine these parts into a complete object.

The following code provides an example on how to perform multipart upload:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// <yourObjectName> indicates the complete path of the object you want to upload to OSS. The path must include the extension of the object name. Exam
ple: abc/efg/123.jpg.
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create an InitiateMultipartUploadRequest object.
InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest("<yourBucketName>", "<yourObjectName>");
// Optional. Specify the storage class.
// ObjectMetadata metadata = new ObjectMetadata();
// metadata.setHeader(OSSHeaders.OSS_STORAGE_CLASS, StorageClass.Standard.toString());
// request.setObjectMetadata(metadata);
// Initiate a multipart upload task.
InitiateMultipartUploadResult upresult = ossClient.initiateMultipartUpload(request);
// Obtain an upload ID. The upload ID is the unique identification of a multipart upload task. You can initiate related requests based on the upload
ID, such as canceling a multipart upload and querying a multipart upload.
String uploadId = upresult.getUploadId();
// partETags is a set of PartETags. A PartETag consists of an ETag and a part number.
List<PartETag> partETags =  new ArrayList<PartETag>();
// Calculate the total number of parts to upload.
final long partSize = 1 * 1024 * 1024L;   // 1MB
final File sampleFile = new File("<localFile>");
long fileLength = sampleFile.length();
int partCount = (int) (fileLength / partSize);
if (fileLength % partSize ! = 0) {
    partCount++;
 }
// Upload each part sequentially until all parts are uploaded.
for (int i = 0; i < partCount; i++) {
    long startPos = i * partSize;
    long curPartSize = (i + 1 == partCount) ? (fileLength - startPos) : partSize;
    InputStream instream = new FileInputStream(sampleFile);
    // Skip the uploaded parts.
    instream.skip(startPos);
    UploadPartRequest uploadPartRequest = new UploadPartRequest();
    uploadPartRequest.setBucketName(bucketName);
    uploadPartRequest.setKey(objectName);
    uploadPartRequest.setUploadId(uploadId);
    uploadPartRequest.setInputStream(instream);
    // Configure the size of each part. Each part except for the last part must be larger than 100 KB.
    uploadPartRequest.setPartSize(curPartSize);
    // Configure part numbers. Each part is configured with a part number. The number ranges from 1 to 10000. If you configure a number beyond the ra
nge, OSS returns an InvalidArgument error code.
    uploadPartRequest.setPartNumber( i + 1);
    // Parts are not necessarily uploaded in order. They may be uploaded from different OSS clients. OSS sorts the parts based on their part numbers
and combines them to obtain a complete object.
    UploadPartResult uploadPartResult = ossClient.uploadPart(uploadPartRequest);
    // After you upload a part, OSS returns a result that contains a PartETag. The PartETag is stored in partETags.
    partETags.add(uploadPartResult.getPartETag());
}
// Create a CompleteMultipartUploadRequest object.
// When the multipart upload task is complete, you must provide all valid partETags. After OSS receives the partETags, OSS verifies the validity of a
ll parts one by one. After all parts are validated, OSS combines these parts into a complete object.
CompleteMultipartUploadRequest completeMultipartUploadRequest =
        new CompleteMultipartUploadRequest(bucketName, objectName, uploadId, partETags);
// Optional. Set the ACL.
// completeMultipartUploadRequest.setObjectACL(CannedAccessControlList.PublicRead);
// Complete the multipart upload task.
CompleteMultipartUploadResult completeMultipartUploadResult = ossClient.completeMultipartUpload(completeMultipartUploadRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Cancel a multipart upload task

You can call the ossClient.abortMultipartUpload method to cancel a multipart upload task. If a multipart upload task is canceled, the upload ID can no longer be used to upload any parts. The parts uploaded by using that upload ID are also deleted.

The following code provides an example on how to cancel a multipart upload task:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Cancel the multipart upload task. The upload ID is returned by InitiateMultipartUpload.
AbortMultipartUploadRequest abortMultipartUploadRequest =
        new AbortMultipartUploadRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");
ossClient.abortMultipartUpload(abortMultipartUploadRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List uploaded parts

You can call the ossClient.listParts method to list all parts that are uploaded by using the specified upload ID.

- Simple list

  The following code provides an example for simple list:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List uploaded parts. The upload ID is returned by InitiateMultipartUpload.
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");
 // Configure an upload ID.
 //listPartsRequest.setUploadId(uploadId);
 // Set the maximum number of parts to list on each page to 100. By default, a maximum of 1,000 parts can be listed.
 listPartsRequest.setMaxParts(100);
 // Specify the start point of the list. Only the parts whose part numbers are greater than the value of this parameter are listed.
 listPartsRequest.setPartNumberMarker(2);
PartListing partListing = ossClient.listParts(listPartsRequest);
for (PartSummary part : partListing.getParts()) {
    // Query part numbers.
    part.getPartNumber();
    // Query the part size.
    part.getSize();
    // Query the ETag.
    part.getETag();
    // Query the last modified time.
    part.getLastModified();
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List all uploaded parts

  By default, listParts can list up to 1,000 parts at a time. The following code provides an example on how to list more than 1,000 parts:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List all uploaded parts.
PartListing partListing;
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");
do {
    partListing = ossClient.listParts(listPartsRequest);
    for (PartSummary part : partListing.getParts()) {
        // Query part numbers.
        part.getPartNumber();
        // Query the part size.
        part.getSize();
        // Query the ETag.
        part.getETag();
        // Query the last modified time.
        part.getLastModified();
    }
// Specify the start point of the list. Only the parts whose part numbers are greater than the value of this parameter are listed.
listPartsRequest.setPartNumberMarker(partListing.getNextPartNumberMarker());
} while (partListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown()
```

- List all uploaded parts by page

  The following code provides an example on how to specify the maximum number of parts to list on each page and list all uploaded parts by page:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List all uploaded parts by page.
PartListing partListing;
ListPartsRequest listPartsRequest = new ListPartsRequest("<yourBucketName>", "<yourObjectName>", "<uploadId>");
// Set the maximum number of parts to list on each page to 100.
listPartsRequest.setMaxParts(100);
do {
    partListing = ossClient.listParts(listPartsRequest);
    for (PartSummary part : partListing.getParts()) {
        // Query part numbers.
        part.getPartNumber();
        // Query the part size.
        part.getSize();
        // Query the ETag.
        part.getETag();
        // Query the last modified time.
        part.getLastModified();
    }
    listPartsRequest.setPartNumberMarker(partListing.getNextPartNumberMarker());
} while (partListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List multipart upload tasks

You can call the ossClient.listMultipartUploads method to list all ongoing multipart upload tasks. Ongoing multipart upload tasks are tasks that are initiated but incomplete or tasks that are canceled. The following table describes the parameters.

Parameter	Description	Configuration method
prefix	Specifies that the returned object names must contain a specified prefix. Note that if you use a prefix for a query, the returned object names contain the prefix.	ListMultipartUploadsRequest.setPrefix(String prefix)
delimiter	Groups objects by name. Objects whose names contain the same string from the prefix and the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes.	ListMultipartUploadsRequest.setDelimiter(String delimiter)
maxUploads	Specifies the maximum number of multipart upload tasks to return. The default value is the same as the maximum value, which is 1000.	ListMultipartUploadsRequest.setMaxUploads(Integer maxUploads)
keyMarker	Specifies the key-marker value from which to start the list. Multipart upload tasks are listed in alphabetical order of their object names. You can use this parameter with the uploadIdMarker parameter to specify the start point to list the returned results.	ListMultipartUploadsRequest.setKeyMarker(String keyMarker)
uploadIdMarker	UploadIdMarker is used with the keyMarker parameter to specify the start point to list the returned results. If the keyMarker parameter is not configured, the uploadIdMarker parameter is invalid. If the keyMarker parameter is configured, the query result includes: • All objects whose names come after the value of keyMarker. • All multipart upload tasks in which the object names come after the keyMarker value and the upload IDs are greater than the uploadIdMarker value.	ListMultipartUploadsRequest.setUploadIdMarker(String uploadIdMarker)

- Simple list

  The following code provides an example for simple list:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List multipart upload tasks. By default, a maximum of 1,000 parts can be listed.
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploadsRequest(bucketName);
MultipartUploadListing multipartUploadListing = ossClient.listMultipartUploads(listMultipartUploadsRequest);
for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads()) {
    // Query upload IDs.
    multipartUpload.getUploadId();
    // Query object names.
    multipartUpload.getKey();
    // Query the time the multipart upload tasks are initiated.
    multipartUpload.getInitiated();
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

If the value of the isTruncated field in the returned result is false, values of nextKeyMarker and nextUploadIdMarker are returned and used as the start point for the next object reading. If all multipart upload tasks cannot be returned at a time, list them by page.

- List all multipart upload tasks

  By default, listMultipartUploads can list up to 1,000 multipart upload tasks at a time. The following code provides an example on how to list more than 1,000 multipart upload tasks:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List multipart upload tasks.
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploadsRequest(bucketName);
do {
    multipartUploadListing = ossClient.listMultipartUploads(listMultipartUploadsRequest);
    for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads()) {
        // Query upload IDs.
        multipartUpload.getUploadId();
        // Query object names.
        multipartUpload.getKey();
        // Query the time the multipart upload tasks are initiated.
        multipartUpload.getInitiated();
    }
    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.getNextKeyMarker());
    listMultipartUploadsRequest.setUploadIdMarker(multipartUploadListing.getNextUploadIdMarker());
} while (multipartUploadListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List all multipart upload tasks by page

  The following code provides an example on how to list all multipart upload tasks by page:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List multipart upload tasks.
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploadsRequest(bucketName);
// Specify the number of multipart upload tasks to list on each page.
listMultipartUploadsRequest.setMaxUploads(50);
do {
    multipartUploadListing = ossClient.listMultipartUploads(listMultipartUploadsRequest);
    for (MultipartUpload multipartUpload : multipartUploadListing.getMultipartUploads()) {
        // Query upload IDs.
        multipartUpload.getUploadId();
        // Query object names.
        multipartUpload.getKey();
        // Query the time the multipart upload tasks are initiated.
        multipartUpload.getInitiated();
    }
    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.getNextKeyMarker());
    listMultipartUploadsRequest.setUploadIdMarker(multipartUploadListing.getNextUploadIdMarker());
} while (multipartUploadListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.6.7. Use a progress bar

Progress bars are used to indicate the progress of an upload or download.

The following code provides an example on how to use upload progress bars by using the ossClient.putObject method:

```
public class PutObjectProgressListener implements ProgressListener {
    private long bytesWritten = 0;
    private long totalBytes = -1;
    private boolean succeed = false;
    @Override
    public void progressChanged(ProgressEvent progressEvent) {
        long bytes = progressEvent.getBytes();
        ProgressEventType eventType = progressEvent.getEventType();
        switch (eventType) {
        case TRANSFER_STARTED_EVENT:
            System.out.println("Start to upload......");
            break;
        case REQUEST_CONTENT_LENGTH_EVENT:
            this.totalBytes = bytes;
            System.out.println(this.totalBytes + " bytes in total will be uploaded to OSS");
            break;
        case REQUEST_BYTE_TRANSFER_EVENT:
            this.bytesWritten += bytes;
            if (this.totalBytes ! = -1) {
                int percent = (int)(this.bytesWritten * 100.0 / this.totalBytes);
                System.out.println(bytes + " bytes have been written at this time, upload progress: " + percent + "%(" + this.bytesWritten + "/" + th
is.totalBytes + ")");
            } else {
                System.out.println(bytes + " bytes have been written at this time, upload ratio: unknown" + "(" + this.bytesWritten + "/...)");
            }
            break;
        case TRANSFER_COMPLETED_EVENT:
            this.succeed = true;
            System.out.println("Succeed to upload, " + this.bytesWritten + " bytes have been transferred in total");
            break;
        case TRANSFER_FAILED_EVENT:
            System.out.println("Failed to upload, " + this.bytesWritten + " bytes have been transferred");
            break;
        default:
            break;
        }
    }
    public boolean isSucceed() {
        return succeed;
    }
    public static void main(String[] args) {
        // The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account to log on to OSS because the account has permissions on
all API operations. We recommend that you use a RAM user to call API operations or perform routine operations and maintenance. To create a RAM user,
log on to https://ram.console.aliyun.com.
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        String objectName = "<yourObjectName>";
        // Create an OSSClient instance.
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        try {
            // View the upload progress information while you upload an object.
            ossClient.putObject(new PutObjectRequest(bucketName, objectName, new File("<yourLocalFile>")).
                    <PutObjectRequest>withProgressListener(new PutObjectProgressListener()));
        } catch (Exception e) {
            e.printStackTrace();
        }
        // Shut down the OSSClient instance.
        ossClient.shutdown();
    }
}
```

You can view progress information when you use ossClient.putObject, ossClient.getObject, or ossClient.uploadPart. However, you cannot view progress information when you use ossClient.uploadFile or ossClient.downloadFile.

4.1.6.8. Upload callback

This topic describes how to use upload callback.

The following code provides an example on how to use upload callback:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Specify the IP address of the server to which you want send the callback request. Example: http://oss-demo.aliyuncs.com:23450 or http://127.0.0.1:
9090.
String callbackUrl = "<yourCallbackServerUrl>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
String content = "Hello OSS";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, objectName,new ByteArrayInputStream(content.getBytes()));
// Configure upload callback parameters.
Callback callback = new Callback();
callback.setCallbackUrl(callbackUrl);
// Optional. Configure the value of the host field included in the callback request header. This value is the same as the value of the host field spe
cified for your server.
// callback.setCallbackHost("yourCallbackHost");
// Configure the value of the body field included in the callback request.
callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":${size}}");
// Configure Content-Type for the callback request.
callback.setCalbackBodyType(CalbackBodyType.JSON);
// Configure custom parameters for the callback request. Each custom parameter consists of a key and a value. The key must start with x:.
callback.addCallbackVar("x:var1", "value1");
callback.addCallbackVar("x:var2", "value2");
putObjectRequest.setCallback(callback);
PutObjectResult putObjectResult = ossClient.putObject(putObjectRequest);
// Read the response for the upload callback request.
byte[] buffer = new byte[1024];
putObjectResult.getResponse().getContent().read(buffer);
// You must close the stream after the object is read. Otherwise, connection leaks may occur. Consequently, no connections are available and an excep
tion occurs.
putObjectResult.getResponse().getContent().close();
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.7. Download objects

4.1.7.1. Overview

OSS provides a variety of methods for you to download objects from OSS.

OSS SDK for Java provides the following object download methods:

- Streaming download
- Download to local files
- Range download
- Resumable download
- Conditional download

You can view the download progress by using a progress bar. For more information, see Use a progress bar.

4.1.7.2. Streaming download

If you have a large object to download or it is time-consuming to download the entire object at a time, you can use streaming download. Streaming download enables you to download part of the object content each time until you download the entire object.

If you do not close the ossObject object after it is used, connection leaks may occur. As a result, no connection is available and an exception occurs. The following code provides an example on how to close an ossObject object:

```
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
ossObject.close();
```

The following code provides an example on how to download an object by using streaming download:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// <yourObjectName> indicates the complete path of the object you want to download from OSS. The path must include the file extension of the object.
For example, set <yourObjectName> to abc/efg/123.jpg.
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// ossObject includes the bucket name, object name, object metadata, and an input stream.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Read the object content.
System.out.println("Object content:");
BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.getObjectContent()));
while (true) {
    String line = reader.readLine();
    if (line == null) break;
    System.out.println("\n" + line);
}
// You must close the stream that is obtained by using ossClient.getObject after the object is read. Otherwise, connection leaks may occur. As a resu
lt, no connections are available and an exception occurs.
reader.close();
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.7.3. Download to local files

This topic describes how to download an object to a local file.

The following code provides an example on how to download a specified object to a local file:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Download the object to a local file. If the name of the object is the same as that of the local file, the object replaces the local file. If the n
ame of the object is different from that of the local file, the object is downloaded.
ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File("<yourLocalFile>"));
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.7.4. Range download

If you need only part of the data in an object, you can use range download to download data within the specified range.

Specify a valid range

The following code provides an example on how to specify a valid range to download an object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// <yourObjectName> indicates the complete path of the object you want to download from OSS. The path must include the file extension of the object.
Example: abc/efg/123.jpg.
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, objectName);
// The valid range for an object whose size is 1,000 bytes is from byte 0 to byte 999.
// Query the data that is within the range from byte 0 to byte 999, which includes a total of 1,000 bytes. If the specified range is invalid, for exa
mple, the specified range includes a negative number, or the specified value is larger than the object size, the entire object is downloaded.
getObjectRequest.setRange(0, 999);
// Start range download.
OSSObject ossObject = ossClient.getObject(getObjectRequest);
// Read the data.
byte[] buf = new byte[1024];
InputStream in = ossObject.getObjectContent();
for (int n = 0; n ! = -1; ) {
    n = in.read(buf, 0, buf.length);
}
// You must close the stream that is obtained by using ossClient.getObject after the object is read. Otherwise, connection leaks may occur. As a resu
lt, no connections are available and an exception occurs.
in.close();
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Streaming download may not read all data at a time. To read 64 KB of data from OSS in streaming mode, you can use the following method to read the data multiple times until 64 KB of data or the entire object is read. For more information, see InputStream.read.

```
byte[] buf = new byte[1024];
InputStream in = ossObject.getObjectContent();
for (int n = 0; n ! = -1; ) {
    n = in.read(buf, 0, buf.length);
}
in.close();
```

Specify an invalid range

The valid range for an object whose size is 1,000 bytes is from byte 0 to byte 999. If the specified range is not within the range, the range does not take effect. Then, OSS returns the 200 HTTP status code and the data of the entire object. The following list provides examples of invalid requests and the returned results:

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.

Compatible download

If you add x-oss-range-behavior:standard to the request header, the download behavior is modified when the specified range is not within the valid range. Example: Use range download to download an object whose size is 1,000 bytes.

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns the 206 HTTP status code and the data that is within the range from byte 500 to byte 999.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns the 416 HTTP status code and the error code InvalidRange.

The following code provides an example on how to perform compatible download:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Specify an object whose size is 1,000 bytes.
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Specify the compatible behavior.
// When the value at the end of the range is invalid, OSS returns the data that is within the range from byte 500 to byte 999. The 206 HTTP status co
de is also returned.
getObjectRequest = new GetObjectRequest(bucketName, objectName);
getObjectRequest.setRange(500, 2000);
getObjectRequest.addHeader("x-oss-range-behavior", "standard");
OSSObject ossObject = ossClient.getObject(getObjectRequest);
ossObject.close();
System.out.println("standard get " + "500~2000 "+ "statusCode:"  + ossObject.getResponse().getStatusCode());
System.out.println("standard get " + "500~2000 " + "contentLength:" + ossObject.getResponse().getContentLength());
try {
    // When the value at the start of the range is invalid, exceptions are returned for the following code. The returned HTTP status code is 416 and
the returned error code is InvalidRange.
    getObjectRequest = new GetObjectRequest(bucketName, objectName);
    getObjectRequest.setRange(1000, 2000);
    getObjectRequest.addHeader("x-oss-range-behavior", "standard");
    ossClient.getObject(getObjectRequest);
} catch (OSSException e) {
    System.out.println("standard get " + "1000~2000:" + "error code:" + e.getErrorCode());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.7.5. Resumable download

You may fail to download a large object if the network is unstable or if the program stops responding. In some cases, you may still fail to download the object even after multiple attempts. To solve this problem, OSS SDK for Java provides the resumable download feature.

To enable resumable download, you need to split the object you want to download into multiple parts and download them separately. After all parts are downloaded, these parts are combined into a complete object.

You can use ossClient.downloadFile to enable resumable download. The following table describes the parameters you can configure for DownloadFileRequest.

Parameter	Description	Required	Default value	Configuration method
bucket	The name of the bucket.	Yes	None	Constructor
key	The name of the object.	Yes	None	Constructor
downloadFile	The path to the local file. The OSS object is downloaded to this file.	No	Name of the object	Constructor or setUploadFile
partSize	The size of each part. Valid values: 1 byte to 5 GB.	No	Object size/10000	setPartSize
taskNum	The number of parts you need to download simultaneously.	No	1	setTaskNum

Parameter	Description	Required	Default value	Configuration method
enableCheckpoint	Specifies whether to enable resumable upload.	No	false	setEnableCheckpoint
checkpointFile	The file that records the results of multipart download. This parameter must be configured to enable resumable upload. This file stores information about download progress. If you fail to download a part, the next download continues based on the recorded progress. After the object is downloaded, the checkpoint file is deleted.	No	downloadFile.ucp (share the same directory with DownloadFile)	setCheckpointFile

The following code provides an example on how to perform resumable download:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Send a request to download 10 tasks simultaneously and start resumable download.
DownloadFileRequest downloadFileRequest = new DownloadFileRequest(bucketName, objectName);
downloadFileRequest.setDownloadFile("<yourDownloadFile>");
downloadFileRequest.setPartSize(1 * 1024 * 1024);
downloadFileRequest.setTaskNum(10);
downloadFileRequest.setEnableCheckpoint(true);
downloadFileRequest.setCheckpointFile("<yourCheckpointFile>");
// Download the object.
DownloadFileResult downloadRes = ossClient.downloadFile(downloadFileRequest);
// After the object is downloaded, the object metadata is returned.
downloadRes.getObjectMetadata();
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.7.6. Conditional download

This topic describes how to download objects that meet the specified conditions.

OSS allows you to configure one or more conditions for object downloads. If the specified conditions are met, the object can be downloaded. If the specified conditions are not met, an error code is returned and the object cannot be downloaded. The following table describes the download conditions.

Parameter	Description
If-Modified-Since	If the specified time is earlier than the time the object is modified, the object can be downloaded. Otherwise, 304 Not modified is returned.
If-Unmodified-Since	If the specified time is later than or equal to the time the object is modified, the object can be downloaded. Otherwise, 412 Precondition failed is returned.
If-Match	If the specified ETag matches that of an object, the object can be downloaded. Otherwise, 412 Precondition failed is returned.
If-None-Match	If the specified ETag does not match that of an object, the object can be downloaded. Otherwise, 304 Not modified is returned.

You can use If-Modified-Since and If-Unmodified-Since as object download conditions at the same time. You can also configure both If-Match and If-None-Match for object downloads.

You can obtain the ETag by using ossClient.getObjectMeta.

The following code provides an example on how to perform conditional download:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
GetObjectRequest request = new GetObjectRequest(bucketName, objectName);
// Configure the download conditions.
request.setModifiedSinceConstraint(new Date());
// Download the object to a local file.
ossClient.getObject(request, new File("<yourLocalFile>"));
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Conditional download supports ossClient.getObject and ossClient.downloadFile.

4.1.7.7. Use a progress bar

Progress bars are used to indicate the progress of an upload or download.

For the complete code of download progress bars, visit GitHub.

The following code provides an example on how to use upload progress bars in ossClient.getObject:

```
static class GetObjectProgressListener implements ProgressListener {
        private long bytesRead = 0;
        private long totalBytes = -1;
        private boolean succeed = false;
        @Override
        public void progressChanged(ProgressEvent progressEvent) {
            long bytes = progressEvent.getBytes();
            ProgressEventType eventType = progressEvent.getEventType();
            switch (eventType) {
            case TRANSFER_STARTED_EVENT:
                System.out.println("Start to download......");
                break;
            case RESPONSE_CONTENT_LENGTH_EVENT:
                this.totalBytes = bytes;
                System.out.println(this.totalBytes + " bytes in total will be downloaded to a local file");
                break;
            case RESPONSE_BYTE_TRANSFER_EVENT:
                this.bytesRead += bytes;
                if (this.totalBytes ! = -1) {
                    int percent = (int)(this.bytesRead * 100.0 / this.totalBytes);
                    System.out.println(bytes + " bytes have been read at this time, download progress: " +
                            percent + "%(" + this.bytesRead + "/" + this.totalBytes + ")");
                } else {
                    System.out.println(bytes + " bytes have been read at this time, download ratio: unknown" +
                            "(" + this.bytesRead + "/...)");
                }
                break;
            case TRANSFER_COMPLETED_EVENT:
                this.succeed = true;
                System.out.println("Succeed to download, " + this.bytesRead + " bytes have been transferred in total");
                break;
            case TRANSFER_FAILED_EVENT:
                System.out.println("Failed to download, " + this.bytesRead + " bytes have been transferred");
                break;
            default:
                break;
            }
        }
        public boolean isSucceed() {
            return succeed;
        }
    }
public static void main(String[] args) {
    // The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
    String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    String accessKeyId = "<yourAccessKeyId>";
    String accessKeySecret = "<yourAccessKeySecret>";
    String bucketName = "<yourBucketName>";
    String objectName = "<yourObjectName>";
    OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
    try {
            // View the download progress while you download an object.
            ossClient.getObject(new GetObjectRequest(bucketName, objectName).
                    <GetObjectRequest>withProgressListener(new GetObjectProgressListener()),
                    new File("<yourLocalFile>"));
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Shut down the OSSClient instance.
    ossClient.shutdown();
}
```

You can view progress information when you use ossClient.putObject, ossClient.getObject, or ossClient.uploadPart. However, you cannot view progress information when you use ossClient.uploadFile or ossClient.downloadFile.

4.1.8. Manage objects

4.1.8.1. Overview

You can manage objects in a bucket by using a series of API operations such as SetObjectAcl, GetObjectAcl, ListObjects, DeleteObject, CopyObject, and DoesObjectExist.

By using the preceding operations, you can perform the following operations:

- Determine whether an object exists
- Manage object ACLs
- Manage object metadata

- List objects
- Query objects
- Delete objects
- Copy objects

4.1.8.2. Determine whether an object exists

This topic describes how to determine whether an object exists.

The following code provides an example on how to determine whether an object exists:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Determine whether the object exists. doesObjectExist includes the isOnlyInOSS parameter. If the value of isOnlyInOSS is true, OSS does not perform
redirection back-to-origin or mirroring back-to-origin when the object does not exist.
If the value of isOnlyInOSS is false, OSS performs redirection back-to-origin or mirroring back-to-origin when the object does not exist.
boolean found = ossClient.doesObjectExist("<yourBucketName>", "<yourObjectName>");
System.out.println(found);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.8.3. Manage object ACLs

This topic describes how to manage the access control list (ACL) of an object.

The following table describes the ACLs that can be configured for objects.

ACL	Description	Value
Inherited from bucket	The ACL of an object is the same as that of the bucket in which the object is stored.	CannedAccessControlList.Default
Private	Only the owner or authorized users of the bucket can read and write the object.	CannedAccessControlList.Private
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	CannedAccessControlList.PublicRead
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	CannedAccessControlList.PublicReadWrite

The ACL of an object takes precedence over the ACL of the bucket in which the object is stored. For example, the ACL of a bucket is private, while the ACL of the object that is stored in this bucket is public. All users can read and write the object. If an object does not have a configured ACL, the object will inherit the ACL of the bucket in which the object is stored.

Configure ACL for an object

The following code provides an example on how to configure ACL for an object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Configure the ACL of the specified object to public read.
ossClient.setObjectAcl("<yourBucketName>", "<yourObjectName>", CannedAccessControlList.PublicRead);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Query the ACL of an object

The following code provides an example on how to query the ACL of a specified object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Query the ACL of an object.
ObjectAcl objectAcl = ossClient.getObjectAcl("<yourBucketName>", "<yourObjectName>");
System.out.println(objectAcl.getPermission().toString());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.8.4. Manage object metadata

Object metadata contains HTTP headers and user metadata.

Configure object metadata

- Configure HTTP headers

  The following code provides an example on how to configure HTTP headers:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String content = "Hello OSS";
// Create object metadata. You can configure HTTP headers by using object metadata.
ObjectMetadata meta = new ObjectMetadata();
String md5 = BinaryUtil.toBase64String(BinaryUtil.calculateMd5(content.getBytes()));
// Enable MD5 verification. After MD5 verification is enabled, OSS calculates the actual MD5 hash and compares this MD5 hash wih that provided in t
he request. If the two values are different, an error occurs.
meta.setContentMD5(md5);
// Specify the type of content you want to upload. The browser determines the format and encoding type used to read the object based on the content
type of the object. If the content type is not specified, a content type is generated based on the object name extension. If no extension is availa
ble, default value application/octet-stream is used as the content type.
meta.setContentType("text/plain");
// Specify the name for the object when the content is downloaded.
meta.setContentDisposition("attachment; filename=\"DownloadFilename\"");
// Specify the length of the object content to upload. If the actual object length is greater than the specified length, the object is uploaded bas
ed on the specified length. If the object is smaller than the specified length, the object is uploaded based on the actual length.
meta.setContentLength(content.length());
// Specify the caching behavior of the website when the content is downloaded.
meta.setCacheControl("Download Action");
// Specify the expiration time of the cache in the GMT.
meta.setExpirationTime(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
// Specify the content encoding format when the content is downloaded.
meta.setContentEncoding("utf-8");
// Configure the header.
meta.setHeader("<yourHeader>", "<yourHeaderValue>");
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Upload the object.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()), meta);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  For more information about HTTP headers, visit RFC 2616.

- Configure user metadata

  You can configure the user metadata of an object to describe the object.

  The following code provides an example on how to configure the user metadata of an object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account to log on to OSS because the account has permissions on all A
PI operations. We recommend that you use a RAM user to call API operations or perform routine operations and maintenance. To create a RAM user, log
on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String content = "Hello OSS";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Create object metadata.
ObjectMetadata meta = new ObjectMetadata();
// Set the value of the property parameter for the user metadata to property-value. We recommend that you use the Base64 encoding method.
meta.addUserMetadata("property", "property-value");
// Upload the object.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(content.getBytes()), meta);
// Query the object metadata.
ossClient.getObjectMetadata("<yourBucketName>", "<yourObjectName>");
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  Object metadata is downloaded with the object. An object may contain multiple pieces of object metadata. The total size of the object metadata cannot exceed 8 KB.

Modify object metadata

The following code provides an example on how to modify object metadata:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String sourceBucketName = "<yourSourceBucketName>";
String sourceObjectName = "<yourSourceObjectName>";
String destinationBucketName = "<yourDestinationBucketName>";
String destinationObjectName = "<yourDestinationObjectName>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,accessKeySecret);
// Replace the source object with the target object. Call ossClient.copyObject to modify object metadata.
CopyObjectRequest request = new CopyObjectRequest(sourceBucketName, sourceObjectName, destinationBucketName, destinationObjectName);
ObjectMetadata meta = new ObjectMetadata();
// Specify the type of content you want to upload. The browser determines the format and encoding type used to read the object based on the content t
ype of the object. If the content type is not specified, a content type is generated based on the object name extension. If no extension is available
, the default value application/octet-stream is used as the content type.
meta.setContentType("text/plain");
// Specify the name for the object when the content is downloaded.
meta.setContentDisposition("Download File Name");
// Specify the caching behavior of the website when the content is downloaded.
meta.setCacheControl("Download Action");
// Specify the expiration time of the cache in the GMT.
meta.setExpirationTime(DateUtil.parseIso8601Date("2022-10-12T00:00:00.000Z"));
// Specify the content encoding format when the content is downloaded.
meta.setContentEncoding("utf-8");
// Configure the header.
meta.setHeader("<yourHeader>", "<yourHeaderValue>");
// Set the value of the property parameter for the user metadata to property-value.
meta.addUserMetadata("property", "property-value");
request.setNewObjectMetadata(meta);
// Modified the object metadata.
ossClient.copyObject(request);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Query object metadata

The following table lists the methods you can use to query object metadata.

Configuration method	Description	Benefits
ossClient.getSimplifiedObjectMeta	Queries part of object metadata such as the ETag, Size, and LastModified (the last modified time) values of the object.	More lightweight and faster
ossClient.getObjectMetadata	Queries all object metadata	None

The following code provides an example on how to query the object metadata:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Query parts of the object metadata.
SimplifiedObjectMeta objectMeta = ossClient.getSimplifiedObjectMeta("<yourBucketName>", "<yourObjectName>");
System.out.println(objectMeta.getSize());
System.out.println(objectMeta.getETag());
System.out.println(objectMeta.getLastModified());
// Query all object metadata.
ObjectMetadata metadata = ossClient.getObjectMetadata("<yourBucketName>", "<yourObjectName>");
System.out.println(metadata.getContentType());
System.out.println(metadata.getLastModified());
System.out.println(metadata.getExpirationTime());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.8.5. List objects

Objects in a bucket are sorted alphabetically in OSS. You can use ossClient.listObjects to list objects in a bucket.

listObjects supports the following types of parameter formats:

- ObjectListing listObjects(String bucketName): lists up to 100 objects in a bucket.
- ObjectListing listObjects(String bucketName, String prefix): lists objects whose names contain a specified prefix in a bucket. A maximum of 100 objects can be listed.
- ObjectListing listObjects(ListObjectsRequest listObjectsRequest): lists objects that meet a specified condition. You can use this format to list objects in a flexible manner.

The following table describes parameters you can configure for ObjectListing.

Parameter	Description	Method
objectSummaries	The metadata of the objects that you want to list.	List<OSSObjectSummary> getObjectSummaries()
prefix	The prefix in the names of the objects that you want to list.	String getPrefix()
delimiter	The character used to group object names.	String getDelimiter()
marker	The position from which the list operation starts.	String getMarker()
maxKeys	The maximum number of objects that can be listed each time.	int getMaxKeys()
nextMarker	The position from which the next list operation starts.	String getNextMarker()
isTruncated	Indicates whether the listed objects are truncated. Valid values: • false: All objects are listed without truncation. • true: Only a part of the objects are listed.	boolean isTruncated()
commonPrefixes	A set of substrings of object names. Objects whose names end with the delimiter and contain the same prefix are grouped as a single result element in commonPrefixes.	List<String> getCommonPrefixes()
encodingType	The encoding type of the object names in the response.	String getEncodingType()

List objects by using a simple list request

The following code provides an example on how to list objects in a specified bucket. By default, a maximum of 100 objects can be listed.

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String KeyPrefix = "<yourKeyPrefix>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List objects.  If you do not set KeyPrefix, all objects in the bucket are listed. If you set KeyPrefix, objects whose names contain the specified
prefix in the bucket are listed.
ObjectListing objectListing = ossClient.listObjects(bucketName, KeyPrefix);
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List objects by using ListObjectsRequest

You can configure parameters of ListObjectsRequest to implement flexible queries. The following table describes the parameters you can configure for ListObjectsRequest.

Parameter	Description	Method
prefix	The prefix that must be included in the names of objects you want to list.	setPrefix(String prefix)
delimiter	The character used to group object names. Objects whose names contain the same string that stretches from the specified prefix to the first delimiter after the prefix are grouped as a CommonPrefixes element.	setDelimiter(String delimiter)
marker	The position from which the list operation starts. All objects whose names are alphabetically after the value of marker are listed.	setMarker(String marker)
maxKeys	The maximum number of objects that can be listed at a time. Maximum value: 1000. Default value: 100.	setMaxKeys(Integer maxKeys)
encodingType	The encoding type of the object names in the response. Only URL encoding is supported.	setEncodingType(String encodingType)

- List a specified number of objects in a bucket

  The following code provides an example on how to list a specified number of objects:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Specify the maximum number of objects that this operation can list.
final int maxKeys = 200;
// List objects.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).withMaxKeys(maxKeys));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List objects whose names contain a specified prefix in a bucket

  The following code provides an example on how to list objects whose names contain a specified prefix in a bucket:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Specify the prefix that is contained in the names of the objects that you want to list.
final String keyPrefix = "<yourkeyPrefix>";
// List objects whose names contain the specified prefix. By default, 100 objects are listed.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).withPrefix(keyPrefix));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List objects whose names are alphabetically after the value of marker

  You can configure the marker parameter to specify the name of the object after which the list operation starts. The following code provides an example on how to list objects whose names are alphabetically after the specified marker:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
final String marker = "<yourMarker>";
// List objects whose names are alphabetically after the specified marker. By default, 100 objects are listed.
ObjectListing objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).withMarker(marker));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List all objects in a bucket by page

  The following code provides an example on how to list all objects in a specified bucket by page. You can set maxKeys to specify the maximum number of objects that can be listed on each page.

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
final int maxKeys = 200;
String nextMarker = null;
ObjectListing objectListing;
do {
    objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).withMarker(nextMarker).withMaxKeys(maxKeys));
    List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
    for (OSSObjectSummary s : sums) {
        System.out.println("\t" + s.getKey());
    }
    nextMarker = objectListing.getNextMarker();
} while (objectListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List objects whose names contain a specified prefix by page

  The following code provides an example on how to list objects whose names contain a specified prefix by page:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List up to 200 objects with the specified prefix in their names on each page.
final int maxKeys = 200;
final String keyPrefix = "<yourkeyPrefix>";
String nextMarker = "<yourNextMarker>";
ObjectListing objectListing;
do {
    objectListing = ossClient.listObjects(new ListObjectsRequest(bucketName).
            withPrefix(keyPrefix).withMarker(nextMarker).withMaxKeys(maxKeys));
    List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
    for (OSSObjectSummary s : sums) {
        System.out.println("\t" + s.getKey());
    }
    nextMarker = objectListing.getNextMarker();
} while (objectListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- List objects whose names are encoded by a specific encoding type

  Currently, OSS supports only URL encoding. If the name of an object contains one or more of the following special characters, the name must be encoded before the object is transmitted:

  - Single quotation marks (')
  - Double quotation marks (")
  - Ampersands (&)
  - Angle brackets (<>)
  - Pause markers (、)
  - Chinese characters

  The following code provides an example on how to list objects whose names are encoded by a specific encoding type:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
final int maxKeys = 200;
final String keyPrefix = "<yourkeyPrefix>";
String nextMarker = "<yourNextMarker>";
ObjectListing objectListing;
do {
    ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
    listObjectsRequest.setPrefix(keyPrefix);
    listObjectsRequest.setMaxKeys(maxKeys);
    listObjectsRequest.setMarker(nextMarker);
    // Specify the encoding type of the object names.
    listObjectsRequest.setEncodingType("url");
    objectListing = ossClient.listObjects(listObjectsRequest);
    // Decode the value of Key in the response.
    for (OSSObjectSummary objectSummary: objectListing.getObjectSummaries()) {
        System.out.println("Key:" + URLDecoder.decode(objectSummary.getKey(), "UTF-8"));
    }
    // Decode the value of commonPrefixes in the response.
    for (String commonPrefixes: objectListing.getCommonPrefixes()) {
        System.out.println("CommonPrefixes:" + URLDecoder.decode(commonPrefixes, "UTF-8"));
    }
    // Decode the value of nextMarker in the response.
    if (objectListing.getNextMarker() != null) {
        nextMarker = URLDecoder.decode(objectListing.getNextMarker(), "UTF-8");
    }
} while (objectListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

List objects in a bucket by directory

OSS uses a flat structure for objects instead of a hierarchical structure to store objects. A directory is a zero-byte object whose name ends with a forward slash (/). You can upload and download this object. By default, objects whose names end with a forward slash (/) are displayed as directories in the OSS console.

You can specify the delimiter and prefix parameters to list objects by directory.

- If prefix is set to the name of a directory, objects and subdirectories whose names contain the prefix are listed.
- If you also set delimiter to a forward slash (/) in the request, the objects and subdirectories whose names start with the specified prefix in the directory are listed. Each subdirectory is listed as a single result element in commonPrefixes. The objects and directories in these subdirectories are not listed.

For the complete sample code that is used to create a directory, visit GitHub.

Consider a bucket that contains the following objects: oss.jpg, fun/test.jpg, fun/movie/001.avi, and fun/movie/007.avi. The forward slash (/) is used as the directory delimiter. The following examples show how to list objects in the bucket.

- List all objects in the bucket

  The following code provides an example on how to list all objects in the bucket:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
// List all objects in the bucket.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);
// Traverse all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
    System.out.println(objectSummary.getKey());
}
// Traverse all commonPrefix elements.
System.out.println("CommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  The following objects are returned:

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
oss.jpg
CommonPrefixes:
```

- List all objects in the specified directory

  The following code provides an example on how to list all objects in the specified directory of the bucket:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
// Set Prefix to fun/ to list all objects in the fun/ directory.
listObjectsRequest.setPrefix("fun/");
// Lists all objects in the fun/ directory.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);
// Traverse all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
    System.out.println(objectSummary.getKey());
}
// Traverse all commonPrefix elements.
System.out.println("\nCommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  The following objects are returned:

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
CommonPrefixes:
```

- List objects and subdirectories in a specified directory

  The following code provides an example on how to list the objects and subdirectories in a specified directory:

```
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName);
// Set the delimiter to a forward slash (/).
listObjectsRequest.setDelimiter("/");
// List all objects and subdirectories in the fun/ directory.
listObjectsRequest.setPrefix("fun/");
ObjectListing listing = ossClient.listObjects(listObjectsRequest);
// Traverse all objects.
System.out.println("Objects:");
// In the response, objectSummaries lists the objects in the fun/ directory.
for (OSSObjectSummary objectSummary : listing.getObjectSummaries()) {
    System.out.println(objectSummary.getKey());
}
// Traverse all commonPrefix elements.
System.out.println("\nCommonPrefixes:");
// commonPrefixs lists the subdirectories in the fun/ directory. The fun/movie/001.avi and fun/movie/007.avi objects are not listed because they ar
e in the movie/ subdirectory of the fun/ directory.
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  The following objects are returned:

```
Objects:
fun/test.jpg
CommonPrefixes:
fun/movie/
```

- List the sizes of objects in the specified directory

  The following code provides an example on how to list the sizes of objects in a specified directory:

```
// Query the total size of objects in the specified directory.
    private static long calculateFolderLength(OSS ossClient, String bucketName, String folder) {
        long size = 0L;
        ObjectListing objectListing = null;
        do {
              // The default value of MaxKeys is 100. The maximum value of MaxKeys is 1000.
            ListObjectsRequest request = new ListObjectsRequest(bucketName).withPrefix(folder).withMaxKeys(1000);
            if (objectListing != null) {
                request.setMarker(objectListing.getNextMarker());
            }
            objectListing = ossClient.listObjects(request);
            List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
            for (OSSObjectSummary s : sums) {
                size += s.getSize();
            }
        } while (objectListing.isTruncated());
        return size;
    }
    public void Calculate() {
        // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        // Create an OSSClient instance.
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        // Specify the prefix. If you want to traverse the sizes of objects in the root directory, leave this parameter empty.
        final String keyPrefix = "<YourPrefix>";
        ObjectListing objectListing = null;
        do {
            // By default, 100 objects are listed each time.
            ListObjectsRequest request = new ListObjectsRequest(bucketName).withDelimiter("/").withPrefix(keyPrefix);
            if (objectListing != null) {
                request.setMarker(objectListing.getNextMarker());
            }
            objectListing = ossClient.listObjects(request);
            List<String> folders = objectListing.getCommonPrefixes();
            for (String folder : folders) {
                System.out.println(folder + " : " + (calculateFolderLength(ossClient, bucketName, folder) / 1024) + "KB");
            }
            List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
            for (OSSObjectSummary s : sums) {
                System.out.println(s.getKey() + " : " + (s.getSize() / 1024) + "KB");
            }
        } while (objectListing.isTruncated());
        ossClient.shutdown();
    }
```

4.1.8.6. Query objects

This topic describes how to call SelectObject to query CSV and JSON objects by using OSS SDK for Java.

The following code provides an example on how to query CSV and JSON objects:

```
package samples;
import com.aliyun.oss.model.*;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
/**
 * Examples of create select object metadata and select object.
 *
 */
public class SelectObjectSample {
    private static String endpoint = "<endpoint, http://oss-cn-hangzhou.aliyuncs.com>";
    private static String accessKeyId = "<yourAccessKeyId>";
    private static String accessKeySecret = "<yourAccessKeySecret>";
    private static String bucketName = "<yourBucketName>";
    public static void main(String[] args) throws Exception {
        OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
        selectCsvSample("test.csv", ossClient);
        selectJsonSample("test.json", ossClient);
        ossClient.shutdown();
    }
    private static void selectCsvSample(String key, OSS ossClient) throws Exception {
```

```
        String content = "name,school,company,age\r\n" +
                "Lora Francis,School A,Staples Inc,27\r\n" +
                "Eleanor Little,School B,\"Conectiv, Inc\",43\r\n" +
                "Rosie Hughes,School C,Western Gas Resources Inc,44\r\n" +
                "Lawrence Ross,School D,MetLife Inc.,24";
        ossClient.putObject(bucketName, key, new ByteArrayInputStream(content.getBytes()));
        SelectObjectMetadata selectObjectMetadata = ossClient.createSelectObjectMetadata(
                new CreateSelectObjectMetadataRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInputFormat(
                                        new CSVFormat().withHeaderInfo(CSVFormat.Header.Use).withRecordDelimiter("\r\n"))));
        System.out.println(selectObjectMetadata.getCsvObjectMetadata().getTotalLines());
        System.out.println(selectObjectMetadata.getCsvObjectMetadata().getSplits());
        SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                        .withInputSerialization(
                                new InputSerialization().withCsvInputFormat(
                                        new CSVFormat().withHeaderInfo(CSVFormat.Header.Use).withRecordDelimiter("\r\n")))
                        .withOutputSerialization(new OutputSerialization().withCsvOutputFormat(new CSVFormat()));
        selectObjectRequest.setExpression("select * from ossobject where _4 > 40");
        OSSObject ossObject = ossClient.selectObject(selectObjectRequest);
        // read object content from ossObject
        BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.getObjectContent()));
        while (true) {
            String line = reader.readLine();
            if (line == null) {
                break;
            }
            System.out.println(line);
        }
        reader.close();
        ossClient.deleteObject(bucketName, key);
    }
    private static void selectJsonSample(String key, OSS ossClient) throws Exception {
        final String content = "{\n" +
                "\t\"name\": \"Lora Francis\",\n" +
                "\t\"age\": 27,\n" +
                "\t\"company\": \"Staples Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Eleanor Little\",\n" +
                "\t\"age\": 43,\n" +
                "\t\"company\": \"Conectiv, Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Rosie Hughes\",\n" +
                "\t\"age\": 44,\n" +
                "\t\"company\": \"Western Gas Resources Inc\"\n" +
                "}\n" +
                "{\n" +
                "\t\"name\": \"Lawrence Ross\",\n" +
                "\t\"age\": 24,\n" +
                "\t\"company\": \"MetLife Inc.\"\n" +
                "}";
        ossClient.putObject(bucketName, key, new ByteArrayInputStream(content.getBytes()));
        SelectObjectRequest selectObjectRequest =
                new SelectObjectRequest(bucketName, key)
                        .withInputSerialization(new InputSerialization()
                                .withCompressionType(CompressionType.NONE)
                                .withJsonInputFormat(new JsonFormat().withJsonType(JsonType.LINES)))
                        .withOutputSerialization(new OutputSerialization()
                                .withCrcEnabled(true)
                                .withJsonOutputFormat(new JsonFormat()))
                        .withExpression("select * from ossobject as s where s.age > 40");
        OSSObject ossObject = ossClient.selectObject(selectObjectRequest);
        // read object content from ossObject
        BufferedReader reader = new BufferedReader(new InputStreamReader(ossObject.getObjectContent()));
        while (true) {
            String line = reader.readLine();
            if (line == null) {
                break;
            }
            System.out.println(line);
        }
        reader.close();
        ossClient.deleteObject(bucketName, key);
    }
}
```

4.1.8.7. Delete objects

This topic describes how to delete objects.

Warning Deleted objects cannot be recovered. Exercise caution when you delete objects.

Delete a single object

The following code provides an example on how to delete a single object:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Delete the object. To delete a folder, set ObjectName to the folder name. If the folder is not empty, you must delete all objects in the folder be
fore you can delete the folder.
ossClient.deleteObject(bucketName, objectName);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Delete multiple objects

A maximum of 1,000 objects can be deleted each time. The deletion results can be returned in the following two modes:

- Verbose: returns a list of deleted objects. The default mode is Verbose.
- quiet: returns a list of objects you failed to delete.

The following table describes the parameters for DeleteObjectsRequest.

Parameter	Description	Configuration method
Keys	Specifies the objects you want to delete.	setKeys(List<String>)
quiet	Specifies the return mode. true indicates quiet. false indicates Verbose. Default value: false.	setQuiet(boolean)
encodingType	Specifies the encoding type of the object name in the response. Only URL encoding is supported.	setEncodingType(String)

The following table describes the parameters you can configure for DeleteObjectsResult.

Parameter	Description	Configuration method
deletedObjects	Indicates the results of the operation. The list of deleted objects is returned when the Verbose mode is used. The list of objects that fail to be deleted is returned when the quiet mode is used.	List<String> getDeletedObjects()
encodingType	Indicates the method for encoding the objects names listed in deletedObjects. When the parameter is empty, the object names are not encoded.	getEncodingType()

The following code provides an example on how to delete multiple objects at a time:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Delete the objects. key is equivalent to ObjectName, which indicates the complete path of the object you want to delete from OSS, and must include
the extension of the object name. Example: abc/efg/123.jpg.
List<String> keys = new ArrayList<String>();
keys.add("key0");
keys.add("key1");
keys.add("key2");
DeleteObjectsResult deleteObjectsResult = ossClient.deleteObjects(new DeleteObjectsRequest(bucketName).withKeys(keys));
List<String> deletedObjects = deleteObjectsResult.getDeletedObjects();
// Shut down the OSSClient instance.
ossClient.shutdown();
```

For the complete code used to delete multiple objects, visit GitHub.

Delete objects whose names contain a specified prefix

The following code provides an example on how to delete objects whose names contain a specified prefix:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
// Specify the prefix.
final String prefix = "<yourkeyPrefix>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// List all objects whose names contain the specified prefix and delete the objects.
String nextMarker = null;
ObjectListing objectListing = null;
do {
    ListObjectsRequest listObjectsRequest = new ListObjectsRequest(bucketName)
            .withPrefix(prefix)
            .withMarker(nextMarker);
    objectListing = ossClient.listObjects(listObjectsRequest);
    if (objectListing.getObjectSummaries().size() > 0) {
        List<String> keys = new ArrayList<String>();
        for (OSSObjectSummary s : objectListing.getObjectSummaries()) {
            System.out.println("key name: " + s.getKey());
            keys.add(s.getKey());
        }
        DeleteObjectsRequest deleteObjectsRequest = new DeleteObjectsRequest(bucketName).withKeys(keys);
        ossClient.deleteObjects(deleteObjectsRequest);
    }
    nextMarker = objectListing.getNextMarker();
} while (objectListing.isTruncated());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.8.8. Copy objects

This topic describes how to copy a small object and a large object.

Copy a small object

You can use ossClient.copyObject to copy an object that is up to 1 GB in size from a source bucket to another destination bucket within the same region. The following table describes the configuration methods of ossClient.copyObject.

Configuration method	Description
CopyObjectResult copyObject(String sourceBucketName, String sourceKey, String destinationBucketName, String destinationKey)	Allows you to specify the source bucket, source object, destination bucket, and destination object. After an object is copied, the content and metadata of the destination object are the same as that of the source object. This method is called simple copy.
CopyObjectResult copyObject(CopyObjectRequest copyObjectRequest)	Allows you to specify metadata of the destination object and copy conditions. If the URLs of the source object and destination object for the copy operation are the same, the metadata of the source object replaces the metadata of the destination object.

The following table describes the parameters you can configure for CopyObjectRequest.

Parameter	Description	Configuration method
sourceBucketName	The name of the source bucket.	setSourceBucketName(String sourceBucketName)
sourceKey	The name of the source object.	setSourceKey(String sourceKey)
destinationBucketName	The name of the destination bucket.	setDestinationBucketName(String destinationBucketName)
destinationKey	The name of the destination object.	setDestinationKey(String destinationKey)
newObjectMetadata	The metadata of the destination object.	setNewObjectMetadata(ObjectMetadata newObjectMetadata)
matchingETagConstraints	The condition for the copy operation. If the ETag value of the source object is the same as the ETag value provided by the user, the copy operation is performed. Otherwise, an error is returned.	setMatchingETagConstraints(List<String> matchingETagConstraints)
nonmatchingEtagConstraints	The condition for the copy operation. If the ETag value of the source object is different from the ETag value provided by the user, the copy operation is performed. Otherwise, an error is returned.	setNonmatchingETagConstraints(List<String> nonmatchingEtagConstraints)
unmodifiedSinceConstraint	The condition for the copy operation. If the specified time is equal to or later than the actual modified time of the source object, the copy operation is performed. Otherwise, an error is returned.	setUnmodifiedSinceConstraint(Date unmodifiedSinceConstraint)
modifiedSinceConstraint	The condition for the copy operation. If the source object is modified after the specified time, the copy operation is performed. Otherwise, an error is returned.	setModifiedSinceConstraint(Date modifiedSinceConstraint)

The following table describes the parameters for CopyObjectRequest.

Parameter	Description	Configuration method
etag	The unique tag of the object.	String getETag()
lastModified	The last modified time of the object.	Date getLastModified()

Note This operation has the following limits:

- You must have read and write permissions on the source object.
- You cannot copy an object from a region to another region.
- The size of the object you want to copy cannot exceed 1 GB.

The following methods can be used to copy small objects.

- Simple copy

  The following code provides an example on how to perform simple copy:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String sourceBucketName = "<yourSourceBucketName>";
String sourceObjectName = "<yourSourceObjectName>";
String destinationBucketName = "<yourDestinationBucketName>";
String destinationObjectName = "<yourDestinationObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Copy the object.
CopyObjectResult result = ossClient.copyObject(sourceBucketName, sourceObjectName, destinationBucketName, destinationObjectName);
System.out.println("ETag: " + result.getETag() + " LastModified: " + result.getLastModified());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Use CopyObjectRequest to copy an object

  The following code provides an example on how to copy an object by using CopyObjectRequest:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String sourceBucketName = "<yourSourceBucketName>";
String sourceObjectName = "<yourSourceObjectName>";
String destinationBucketName = "<yourDestinationBucketName>";
String destinationObjectName = "<yourDestinationObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a CopyObjectRequest object.
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(sourceBucketName, sourceObjectName, destinationBucketName, destinationObjectName);
// Configure new object metadata.
ObjectMetadata meta = new ObjectMetadata();
meta.setContentType("text/html");
copyObjectRequest.setNewObjectMetadata(meta);
// Copy the object.
CopyObjectResult result = ossClient.copyObject(copyObjectRequest);
System.out.println("ETag: " + result.getETag() + " LastModified: " + result.getLastModified());
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Copy a large object

To copy an object larger than 1 GB, you must use multipart copy. To implement multipart copy, perform the following operations:

1. Use ossClient.initiateMultipartUpload to initiate a multipart copy task.
2. Use ossClient.uploadPartCopy to perform the multipart copy task. All parts except for the last part must be larger than 100 KB.
3. Use ossClient.completeMultipartUpload to complete the multipart copy task.

For the complete code that is used to perform multipart copy, visit GitHub.

The following code provides an example on how to perform multipart copy:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String sourceBucketName = "<yourSourceBucketName>";
String sourceObjectName = "<yourSourceObjectName>";
String destinationBucketName = "<yourDestinationBucketName>";
String destinationObjectName = "<yourDestinationObjectName>";
ObjectMetadata objectMetadata = ossClient.getObjectMetadata(sourceBucketName, sourceObjectName);
// Query the size of the object to copy.
long contentLength = objectMetadata.getContentLength();
// Set the size of each part to 10 MB.
long partSize = 1024 * 1024 * 10;
// Calculate the total number of parts.
int partCount = (int) (contentLength / partSize);
if (contentLength % partSize ! = 0) {
    partCount++;
}
System.out.println("total part count:" + partCount);
// Initiate a multipart copy task. You can call the InitiateMultipartUploadRequest operation to specify the metadata of the target object.
InitiateMultipartUploadRequest initiateMultipartUploadRequest = new InitiateMultipartUploadRequest(destinationBucketName, destinationObjectName);
InitiateMultipartUploadResult initiateMultipartUploadResult = ossClient.initiateMultipartUpload(initiateMultipartUploadRequest);
String uploadId = initiateMultipartUploadResult.getUploadId();
// Start the multipart copy task.
List<PartETag> partETags = new ArrayList<PartETag>();
for (int i = 0; i < partCount; i++) {
     // Calculate the size of each part.
    long skipBytes = partSize * i;
    long size = partSize < contentLength - skipBytes ? partSize : contentLength - skipBytes;
    // Create an UploadPartCopyRequest object. You can call the UploadPartCopyRequest operation to specify conditions.
    UploadPartCopyRequest uploadPartCopyRequest =
        new UploadPartCopyRequest(sourceBucketName, sourceObjectName, destinationBucketName, destinationObjectName);
    uploadPartCopyRequest.setUploadId(uploadId);
    uploadPartCopyRequest.setPartSize(size);
    uploadPartCopyRequest.setBeginIndex(skipBytes);
    uploadPartCopyRequest.setPartNumber(i + 1);
    UploadPartCopyResult uploadPartCopyResult = ossClient.uploadPartCopy(uploadPartCopyRequest);
    // Store the returned part ETag in partETags.
    partETags.add(uploadPartCopyResult.getPartETag());
}
// Complete the multipart copy task.
CompleteMultipartUploadRequest completeMultipartUploadRequest = new CompleteMultipartUploadRequest(
                    destinationBucketName, destinationObjectName, uploadId, partETags);
ossClient.completeMultipartUpload(completeMultipartUploadRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.9. Authorized access

This topic describes how to use Security Token Service (STS) and a signed URL to authorize access.

Use STS to authorize temporary access

Alibaba Cloud Security Token Service (STS) is a web service that provides a temporary access token to a cloud computing user. You can use STS to grant a third-party application or your RAM user an access credential with a customized validity period and permissions.

STS has the following benefits:

- You need only to generate an access token and send the access token to a third-party application, instead of exposing your long-term AccessKey pair to the third-party application. You can customize the access permissions and validity period of this token.
- The access token automatically expires when the validity period ends.

The following code provides an example on how to create a signature request by using STS:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";
// After you obtain a temporary STS credential for your OSSClient, an OSSClient instance is created based on the security token and temporary AccessK
ey pair (AccessKey ID and AccessKey secret) contained in the credential.
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
// Perform an operation in OSS. For example, you can upload or download an object.
// Upload an object to OSS.
// ossClient.putObject(putObjectRequest);
// Download an object to a local file. If the name of the object is the same as that of the local file, the object replaces the local file. If the na
me of the object is different from that of the local file, the object is downloaded.
// ossClient.getObject(new GetObjectRequest(bucketName, objectName), new File("<yourLocalFile>"));
// Shut down the OSSClient instance.
ossClient.shutdown();
```

Use a signed URL to authorize temporary access

- Generate a signed URL

  You can generate a signed URL and provide it to a visitor to grant temporary access. When you generate a signed URL, you can specify the validity period of the URL to restrict the period of access from visitors.

- Generate a signed URL to allow HTTP GET requests

  The following code provides an example on how to generate a signed URL to allow HTTP GET requests:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Set the validity period of the URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
// Generate a signed URL to allow HTTP GET requests. Visitors can use a browser for access.
URL url = ossClient.generatePresignedUrl(bucketName, objectName, expiration);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Generate a signed URL to allow other HTTP methods

  If you want to allow visitors to temporarily perform other operations such as uploading or deleting objects, you must generate a corresponding signed URL. The following code provides an example on how to upload an object by using a signed URL to allow HTTP PUT requests:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// After you obtain a temporary STS credential for your OSSClient, an OSSClient instance is created based on the security token and temporary Acces
sKey pair (AccessKey ID and AccessKey secret) contained in the credential.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret, securityToken);
GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.PUT);
// Set the validity period of the URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
request.setExpiration(expiration);
// Set ContentType.
request.setContentType(DEFAULT_OBJECT_CONTENT_TYPE);
// Set user metadata.
request.addUserMetadata("author", "aliy");
// Generate a signed URL to allow HTTP PUT requests.
URL signedUrl = ossClient.generatePresignedUrl(request);
Map<String, String> requestHeaders = new HashMap<String, String>();
requestHeaders.put(HttpHeaders.CONTENT_TYPE, DEFAULT_OBJECT_CONTENT_TYPE);
requestHeaders.put(OSS_USER_METADATA_PREFIX + "author", "aliy");
// Use the signed URL to upload an object.
ossClient.putObject(signedUrl, new ByteArrayInputStream("Hello OSS".getBytes()), -1, requestHeaders, true);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  Visitors can specify the HttpMethod.PUT parameter and use the signed URL to upload objects.

- Generate a signed URL with specified parameters

  The following code provides an example on how to generate a signed URL with specified parameters:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Create a request.
GeneratePresignedUrlRequest generatePresignedUrlRequest = new GeneratePresignedUrlRequest(bucketName, objectName);
// Set HttpMethod to PUT.
generatePresignedUrlRequest.setMethod(HttpMethod.PUT);
// Add user metadata.
generatePresignedUrlRequest.addUserMetadata("author", "baymax");
// Set Content-Type.
generatePresignedUrlRequest.setContentType("application/octet-stream");
// Set the validity period of the URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
generatePresignedUrlRequest.setExpiration(expiration);
// Generate a signed URL.
URL url = ossClient.generatePresignedUrl(generatePresignedUrlRequest);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

- Use a signed URL to upload or download an object
  - Use a signed URL to download an object

    The following code provides an example on how to download a specified object by using a signed URL:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
Date expiration = DateUtil.parseRfc822Date("Wed, 18 Mar 2022 14:20:00 GMT");
GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.GET);
// Set the validity period.
request.setExpiration(expiration);
// Generate a signed URL to allow HTTP GET requests.
URL signedUrl = ossClient .generatePresignedUrl(request);
System.out.println("signed url for getObject: " + signedUrl);
// Use the signed URL to send a request.
Map<String, String> customHeaders = new HashMap<String, String>();
// Specify the request headers of GetObject.
customHeaders.put("Range", "bytes=100-1000");
OSSObject object = ossClient.getObject(signedUrl,customHeaders);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

  - Use a signed URL to upload an object

    The following code provides an example on how to upload an object by using a signed URL:

```
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Create an OSSClient instance.
OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);
// Generate a signed URL.
Date expiration = DateUtil.parseRfc822Date("Thu, 19 Mar 2019 18:00:00 GMT");
GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucketName, objectName, HttpMethod.PUT);
// Set the validity period.
request.setExpiration(expiration);
// Set Content-Type.
request.setContentType("application/octet-stream");
// Add user metadata.
request.addUserMetadata("author", "aliy");
// Generate a signed URL to allow HTTP PUT requests.
URL signedUrl = ossClient.generatePresignedUrl(request);
System.out.println("signed url for putObject: " + signedUrl);
// Use the signed URL to send a request.
File f = new File("<yourLocalFile>");
FileInputStream fin = new FileInputStream(f);
// Specify the request headers of PutObject.
Map<String, String> customHeaders = new HashMap<String, String>();
customHeaders.put("Content-Type", "application/octet-stream");
customHeaders.put("x-oss-meta-author", "aliy");
PutObjectResult result = ossClient.putObject(signedUrl, fin, f.length(), customHeaders);
// Shut down the OSSClient instance.
ossClient.shutdown();
```

4.1.10. IMG

OSS Image Processing (IMG) is a secure, cost-effective, and highly reliable image processing service that can process large amounts of data. After you upload source images to OSS, you can call RESTful API operations to process the images anytime, anywhere, and on any Internet device. You can use OSS IMG to build your own image processing service.

Basic features

OSS IMG provides the following features:

- Query image information.
- Convert image formats.
- Resize, crop, and rotate images.
- Apply image effect.
- Add image, text, and text-and-image watermarks to images.
- Customize image processing styles. Log on to the OSS console. Click a bucket name to go to the bucket details page. Click the **Image Processing (IMG)** tab and click **Create Rule** to customize an image processing style.
- Use cascade processing to call multiple IMG features.

Usage

You can use standard `HTTP GET` requests to access IMG. All processing parameters are encoded in the query string of a URL.

- Anonymous access

  If the ACL of an image object is public-read, as described in the following table, anonymous users can access the image object.

Bucket ACL	Object ACL
public-read or public-read-write	default
Any ACL	public-read or public-read-write

  You can anonymously access IMG by using a third-level domain in the following format:

```
http://bucket.<endpoint>/object?x-oss-process=image/action,parame_value
```

Parameter	Description
bucket	The name of the bucket.
endpoint	The domain name used to access the region.
object	The name of the image object.
image	The identifier reserved by IMG.
action	The operations performed on images, such as scaling, cropping, and rotating.
param	The parameters corresponding to the operations performed on the images.

  - Basic operations

    For example, you can scale an image to a width of 100 and adjust the height based on the ratio:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

  - Customize image styles

    You can anonymously access IMG by using a third-level domain in the following format:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
```

    - style: the identifier reserved by IMG for the custom style.
    - yourStyleName: the name of the custom style. The name is specified by the Rule Name parameter when you create the custom style in the console.

    Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
```

  - Cascade processing

    You can use cascade processing to perform multiple operations in sequence on an image by using a URL in the following format:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
```

    Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
```

  - Access by using HTTPS

    IMG supports access by using HTTPS. Example:

```
https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

- Authorized access

  If the ACL of an image object is private, you must have permissions on the image object before you can perform operations on it.

Bucket ACL	Object ACL
private	default
Any ACL	private

  The following code provides an example on how to generate a signed URL to access IMG:

```
String endpoint = "<Your OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>";
String accessKeyId = "<accessKeyId>";
String accessKeySecret = "<accessKeySecret>";
String bucketName = "<bucketName>";
private static String key = "example.jpg";
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Configure the IMG style.
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
// Set the validity period to 10 minutes.
Date expiration = new Date(new Date().getTime() + 1000 * 60 * 10 );
GeneratePresignedUrlRequest req = new GeneratePresignedUrlRequest(bucketName, key, HttpMethod.GET);
req.setExpiration(expiration);
req.setProcess(style);
URL signedUrl = ossClient.generatePresignedUrl(req);
System.out.println(signedUrl);
```

Note

- You can set Create Style, HTTPS, and Cascading Processing for authorized access.
- Use code in the following format to specify the expiration time: `Date expiration = DateUtil.parseRfc822Date("Wed, 21 Dec 2022 14:20:00 GMT");` .

- Access by using SDKs

  You can use OSS SDKs to access and process image objects of any ACLs.

  Note

  - For the complete IMG code, visit GitHub.
  - You can use OSS SDKs to process image objects by configuring Create Style, HTTPS, and Cascading Processing.

  - Basic operations

    The following code provides an example on how to perform basic IMG operations including obtaining image information, converting image formats, resizing, cropping, rotating, watermarking, and applying effects:

```
String endpoint = "<Your OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>";
String accessKeyId = "<accessKeyId>";
String accessKeySecret = "<accessKeySecret>";
String bucketName = "<bucketName>";
String key = "example.jpg";
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Resize the image.
String style = "image/resize,m_fixed,w_100,h_100";
GetObjectRequest request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-resize.jpg"));
// Crop the image.
style = "image/crop,w_100,h_100,x_100,y_100,r_1";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-crop.jpg"));
// Rotate the image.
style = "image/rotate,90";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-rotate.jpg"));
// Sharpen the image.
style = "image/sharpen,100";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-sharpen.jpg"));
// Add watermarks to the image.
style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-watermark.jpg"));
// Convert the image format.
style = "image/format,png";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-format.png"));
// Obtain the image information.
style = "image/info";
request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-info.txt"));
ossClient.shutdown();
```

- Custom image styles

```
String endpoint = "<Your OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>";
String accessKeyId = "<accessKeyId>";
String accessKeySecret = "<accessKeySecret>";
String bucketName = "<bucketName>";
String key = "example.jpg";
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Customize the image style.
String style = "style/oss-pic-style-w-100";
GetObjectRequest request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-new.jpg"));
ossClient.shutdown();
```

- Cascade processing

```
String endpoint = "<Your OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>";
String accessKeyId = "<accessKeyId>";
String accessKeySecret = "<accessKeySecret>";
String bucketName = "<bucketName>";
private static String key = "example.jpg";
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);
// Perform cascade processing.
String style = "image/resize,m_fixed,w_100,h_100/rotate,90";
GetObjectRequest request = new GetObjectRequest(bucketName, key);
request.setProcess(style);
ossClient.getObject(request, new File("example-new.jpg"));
ossClient.shutdown();
```

4.1.11. Handle exceptions

If an exception occurs when you call the OSSClient class-related API operation, the related operation fails. Otherwise, the related operation succeeds. If an exception occurs, the data returned by the operation is invalid. OSS SDK for Java exceptions are classified into two types: OSSException and ClientException. Both are subclasses of RuntimeException.

Examples

```
try {
    // Do some operations with the instance here, such as put object...
    client.putObject(...);
} catch (OSSException oe) {
    System.out.println("Caught an OSSException, which means your request made it to OSS, "
            + "but was rejected with an error response for some reason.") ;
    System.out.println("Error Message: " + oe.getErrorCode());
    System.out.println("Error Code:       " + oe.getErrorCode());
    System.out.println("Request ID:      " + oe.getRequestId());
    System.out.println("Host ID:           " + oe.getHostId());
} catch (ClientException ce) {
    System.out.println("Caught an ClientException, which means the client encountered "
            + "a serious internal problem while trying to communicate with OSS, "
            + "such as not being able to access the network.") ;
    System.out.println("Error Message: " + ce.getMessage());
} finally {
    if (client ! = null) {
        client.shutdown();
    }
}
```

ClientException

ClientException indicates the exception that occurs when a client sends a request or transmit data to OSS. For example, ClientException is returned when a request is sent under poor network conditions. ClientException is also returned when an I/O exception occurs during object upload.

OSSException

OSSException indicates an error response from OSS. The error code and error message are returned so that you can find the problem and handle it properly.

OSSException involves the following error information:

- Code: the error code that OSS returns to the user.
- Message: the detailed error message provided by OSS.
- RequestId: the UUID of a request. If the problem persists, you can seek help from OSS development engineers and provide them with the request ID.
- HostId: the ID of a host to identify the accessed OSS cluster, which is the same as the host ID specified in the request.

Error codes

Error code	Error message	HTTP status code
AccessDenied	The error message returned because the access is denied.	403
BucketAlreadyExists	The error message returned because the bucket already exists.	409
BucketNotEmpty	The error message returned because the bucket is not empty.	409

Error code	Error message	HTTP status code
EntityTooLarge	The error message returned because the entity exceeds the maximum allowed size.	400
EntityTooSmall	The error message returned because the entity is smaller than the minimum allowed size.	400
FileGroupTooLarge	The error message returned because the object group exceeds the maximum allowed size.	400
FilePartNotExist	The error message returned because the specified part does not exist.	400
FilePartStale	The error message returned because the part has expired.	400
InvalidArgument	The error message returned because the parameter format is invalid.	400
InvalidAccessKeyId	The error message returned because the specified Accesskey ID does not exist.	403
InvalidBucketName	The error message returned because the bucket name is invalid.	400
InvalidDigest	The error message returned because the digest is invalid.	400
InvalidObjectName	The error message returned because the object name is invalid.	400
InvalidPart	The error message returned because the part is invalid.	400
InvalidPartOrder	The error message returned because the part sequence is invalid.	400
InvalidTargetBucketForLogging	The error message returned because the destination bucket specified for logging is invalid.	400
InternalError	The error message returned because an internal OSS error occurs.	500
MalformedXML	The error message returned because the XML format is invalid.	400
MethodNotAllowed	The error message returned because the method is not supported.	405
MissingArgument	The error message returned because the required parameters are not set.	411
MissingContentLength	The error message returned because the content length is not specified.	411
NoSuchBucket	The error message returned because the specified bucket does not exist.	404
NoSuchKey	The error message returned because the specified object does not exist.	404
NoSuchUpload	The error message returned because the specified multipart upload ID does not exist.	404
NotImplemented	The error message returned because the method cannot be implemented.	501
PreconditionFailed	The error message returned because a precondition error occurs.	412
RequestTimeTooSkewed	The error message returned because the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes.	403
RequestTimeout	The error message returned because the request times out.	400
SignatureDoesNotMatch	The error message returned because the signature is invalid.	403
InvalidEncryptionAlgorithmError	The error message returned because the specified encryption algorithm based on entropy encoding is invalid.	400

4.1.12. FAQ

This topic lists the causes and solutions for common errors you may encounter when you use OSS SDK for Java.

JAR conflicts

If an error similar to the following is displayed when you use OSS SDK for Java, a JAR conflict is in your project:

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/ssl/TrustStrategy
    at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
    at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
    at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:77)
Caused by: java.lang.ClassNotFoundException: org.apache.http.ssl.TrustStrategy
    at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
    ... 3 more
```

Or

```
Exception in thread "main" java.lang.NoSuchFieldError: INSTANCE
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<init>(DefaultHttpRequestWriterFactory.java:52)
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<init>(DefaultHttpRequestWriterFactory.java:56)
 at org.apache.http.impl.io.DefaultHttpRequestWriterFactory.<clinit>(DefaultHttpRequestWriterFactory.java:46)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:82)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:95)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<init>(ManagedHttpClientConnectionFactory.java:104)
 at org.apache.http.impl.conn.ManagedHttpClientConnectionFactory.<clinit>(ManagedHttpClientConnectionFactory.java:62)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager$InternalConnectionFactory.<init>(PoolingHttpClientConnectionManager.java:572)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:174)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:158)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:149)
 at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.<init>(PoolingHttpClientConnectionManager.java:125)
 at com.aliyun.oss.common.comm.DefaultServiceClient.createHttpClientConnectionManager(DefaultServiceClient.java:237)
 at com.aliyun.oss.common.comm.DefaultServiceClient.<init>(DefaultServiceClient.java:78)
 at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
 at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
 at OSSManagerImpl.upload(OSSManagerImpl.java:42)
 at OSSManagerImpl.main(OSSManagerImpl.java:63)
```

Cause: A possible cause is that the OSS Java SDK uses Apache httpclient 4.4.1, whereas your project uses a version of Apache httpclient or commons-httpclient that conflicts with Apache httpclient 4.4.1. To view the JAR package and its version that is used in your project, run the `mvn dependency:tree` command in the directory of the project. The following figure shows that the project uses Apache HttpClient 4.3.

```
[INFO] --- maven-dependency-plugin:2.2:tree (default-cli) @ maven-demo ---
[INFO] com.aliyun.oss:maven-demo:jar:0.1.1-SNAPSHOT
[INFO] +- junit:junit:jar:4.10:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.1:test
[INFO] +- org.apache.httpcomponents:httpclient:jar:4.3:compile
[INFO] |  +- org.apache.httpcomponents:httpcore:jar:4.3:compile
[INFO] |  +- commons-logging:commons-logging:jar:1.1.3:compile
[INFO] |  \- commons-codec:commons-codec:jar:1.6:compile
[INFO] \- com.aliyun.oss:aliyun-sdk-oss:jar:2.2.1:compile
[INFO]    +- org.jdom:jdom:jar:1.1:compile
[INFO]    \- net.sf.json-lib:json-lib:jar:jdk15:2.4:compile
[INFO]       +- commons-beanutils:commons-beanutils:jar:1.8.0:compile
[INFO]       +- commons-collections:commons-collections:jar:3.2.1:compile
[INFO]       +- commons-lang:commons-lang:jar:2.5:compile
[INFO]       \- net.sf.ezmorph:ezmorph:jar:1.0.6:compile
```

Solution: Use one of the following methods to resolve JAR conflicts:

- Use consistent versions of Apache HttpClient. If your project uses a version of Apache HttpClient that conflicts with Apache HttpClient 4.4.1, use Apache HttpClient 4.4.1 and delete all other Apache HttpClient version dependencies in the pom.xml file. If your project uses Commons HttpClient, conflicts may also occur. To resolve these conflicts, delete Commons HttpClient.
- Resolve dependency conflicts. If your project is dependent on multiple third-party packages and the third-party packages are dependent on different versions of Apache HttpClient, your project may contain dependency conflicts. Use exclusions to resolve these conflicts. For more information, visit Maven Guides.

OSS SDK for Java is dependent on the following package versions. The solution to conflicts is similar to that for HttpClient.

```
[INFO] com.aliyun.oss:maven-demo:jar:0.1.1-SNAPSHOT
[INFO] +- junit:junit:jar:4.10:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.1:test
[INFO] \- com.aliyun.oss:aliyun-sdk-oss:jar:2.2.1:compile
[INFO]    +- org.apache.httpcomponents:httpclient:jar:4.4.1:compile
[INFO]    |  +- org.apache.httpcomponents:httpcore:jar:4.4.1:compile
[INFO]    |  +- commons-logging:commons-logging:jar:1.2:compile
[INFO]    |  \- commons-codec:commons-codec:jar:1.9:compile
[INFO]    +- org.jdom:jdom:jar:1.1:compile
[INFO]    \- net.sf.json-lib:json-lib:jar:jdk15:2.4:compile
[INFO]       +- commons-beanutils:commons-beanutils:jar:1.8.0:compile
[INFO]       +- commons-collections:commons-collections:jar:3.2.1:compile
[INFO]       +- commons-lang:commons-lang:jar:2.5:compile
[INFO]       \- net.sf.ezmorph:ezmorph:jar:1.0.6:compile
```

Missing packages

An error similar to the following is displayed when you use OSS SDK for Java:

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/auth/Credentials
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:76)
Caused by: java.lang.ClassNotFoundException: org.apache.http.auth.Credentials
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 3 more
```

Or

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/http/protocol/HttpContext
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:268)
        at com.aliyun.oss.OSSClient.<init>(OSSClient.java:193)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:76)
Caused by: java.lang.ClassNotFoundException: org.apache.http.protocol.HttpContext
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 3 more
```

Or

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/jdom/input/SAXBuilder
        at com.aliyun.oss.internal.ResponseParsers.getXmlRootElement(ResponseParsers.java:645)
        at ... ...
        at com.aliyun.oss.OSSClient.doesBucketExist(OSSClient.java:471)
        at com.aliyun.oss.OSSClient.doesBucketExist(OSSClient.java:465)
        at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:82)
Caused by: java.lang.ClassNotFoundException: org.jdom.input.SAXBuilder
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
        ... 11 more
```

Cause: The JAR package used to compile or run OSS SDK for Java may be missing in your project. OSS SDK for Java depends on the following JAR packages:

- aliyun-sdk-oss-2.2.1.jar
- hamcrest-core-1.1.jar
- jdom-1.1.jar
- commons-codec-1.9.jar
- httpclient-4.4.1.jar
- commons-logging-1.2.jar
- httpcore-4.4.1.jar
- log4j-1.2.15.jar

All packages except the log4j-1.2.15.jar package are required for OSS Java SDK. However, if you want to enable logging, you must also include the log4j-1.2.15.jar package.

Solution: Add packages on which OSS SDK for Java is dependent to your projects by using the following methods:

- If you run your project in Eclipse, import dependencies to Eclipse, as instructed in the "Method 2: Import a JAR package" section of Installation.
- If you run your project in Ant, add the packages on which OSS SDK for Java is dependent to the lib directory of the project.
- If you use the .javac or .java file, use the -classpath or -cp option to specify the path in which OSS Java SDK dependent packages are added, or save the Java SDK dependencies to classpath.

Connection timeout

If a similar exception is displayed when you run OSS SDK for Java:

```
com.aliyun.oss.ClientException: SocketException
    at com.aliyun.oss.common.utils.ExceptionFactory.createNetworkException(ExceptionFactory.java:71)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultServiceClient.java:116)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java:121)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:67)
    at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:92)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:140)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:111)
    at com.aliyun.oss.internal.OSSBucketOperation.getBucketInfo(OSSBucketOperation.java:1152)
    at com.aliyun.oss.OSSClient.getBucketInfo(OSSClient.java:1220)
    at com.aliyun.oss.OSSClient.getBucketInfo(OSSClient.java:1214)
    at com.aliyun.oss.demo.HelloOSS.main(HelloOSS.java:94)
Caused by: org.apache.http.conn.HttpHostConnectException: Connect to oss-test.oss-cn-hangzhou-internal.aliyuncs.com:80 [oss-test.oss-cn-hangzhou-inte
rnal.aliyuncs.com/10.84.135.99] failed: Connection timed out: connect
    at org.apache.http.impl.conn.DefaultHttpClientConnectionOperator.connect(DefaultHttpClientConnectionOperator.java:151)
    at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.connect(PoolingHttpClientConnectionManager.java:353)
    at org.apache.http.impl.execchain.MainClientExec.establishRoute(MainClientExec.java:380)
    at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:236)
    at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:184)
    at org.apache.http.impl.execchain.RedirectExec.execute(RedirectExec.java:110)
    at org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:184)
    at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:82)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultServiceClient.java:113)
    ... 9 more
```

Cause: The endpoint is incorrect or the network is disconnected.

Solution: Solve the network problem and try again.

org.apache.http.NoHttpResponseException: The target server failed to respond

If a similar exception is displayed when you run OSS SDK for Java:

```
com.aliyun.oss.ClientException: Unknown
        at com.aliyun.oss.common.utils.ExceptionFactory.createNetworkException(ExceptionFactory.java:68) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultServiceClient.java:115) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java:121) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:67) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:92) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:140) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:111) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.internal.OSSMultipartOperation.initiateMultipartUpload(OSSMultipartOperation.java:206) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.aliyun.oss.OSSClient.initiateMultipartUpload(OSSClient.java:765) ~[aliyun-sdk-oss-2.1.0.jar:na]
        at com.taobao.agoo.dump.client.OssTools.multipartUpload(OssTools.java:79) ~[agoo-dump-client-2.0.0-SNAPSHOT.jar:na]
        at com.taobao.agoo.dump.biz.manager.TaskExcutorManager$UploadTask.run(TaskExcutorManager.java:114) ~[agoo-dump-biz-2.0.0-SNAPSHOT.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471) [na:1.7.0_51]
        at java.util.concurrent.FutureTask.run(FutureTask.java:262) [na:1.7.0_51]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) [na:1.7.0_51]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615) [na:1.7.0_51]
        at java.lang.Thread.run(Thread.java:744) [na:1.7.0_51]
Caused by: org.apache.http.NoHttpResponseException: The target server failed to respond
        at org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:143) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:57) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.impl.io.AbstractMessageParser.parse(AbstractMessageParser.java:261) ~[httpcore-4.4.jar:4.4]
        at org.apache.http.impl.DefaultBHttpClientConnection.receiveResponseHeader(DefaultBHttpClientConnection.java:165) ~[httpcore-4.4.jar:4.4]
        at org.apache.http.impl.conn.CPoolProxy.receiveResponseHeader(CPoolProxy.java:167) ~[httpclient-4.4.jar:4.4]
        at org.apache.http.protocol.HttpRequestExecutor.doReceiveResponse(HttpRequestExecutor.java:272) ~[httpcore-4.4.jar:4.4]
```

Cause: The preceding error may occur if expired connections are used. This error occurs only when a Java SDK version earlier than 2.1.2 is used.

Solution: Upgrade OSS SDK for Java to V2.1.2 or later.

OSS SDK for Java stops responding when OSS SDK for Java is called

If OSS SDK for Java stops responding when you call OSS SDK for Java, call `jstack -l <pid>` to view the stack. The similar error is displayed at the following positions:

```
"main" prio=6 tid=0x000000000291e000 nid=0xc40 waiting on condition [0x0000000002dae000]
java.lang.Thread.State: WAITING (parking)
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for  <0x00000007d85697f8> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:186)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2043)
    at org.apache.http.pool.PoolEntryFuture.await(PoolEntryFuture.java:138)
    at org.apache.http.pool.AbstractConnPool.getPoolEntryBlocking(AbstractConnPool.java:306)
    at org.apache.http.pool.AbstractConnPool.access$000(AbstractConnPool.java:64)
    at org.apache.http.pool.AbstractConnPool$2.getPoolEntry(AbstractConnPool.java:192)
    at org.apache.http.pool.AbstractConnPool$2.getPoolEntry(AbstractConnPool.java:185)
    at org.apache.http.pool.PoolEntryFuture.get(PoolEntryFuture.java:107)
    at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.leaseConnection(PoolingHttpClientConnectionManager.java:276)
    at org.apache.http.impl.conn.PoolingHttpClientConnectionManager$1.get(PoolingHttpClientConnectionManager.java:263)
    at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:190)
    at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:184)
    at org.apache.http.impl.execchain.RedirectExec.execute(RedirectExec.java:110)
    at org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:184)
    at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:82)
    at com.aliyun.oss.common.comm.DefaultServiceClient.sendRequestCore(DefaultServiceClient.java:113)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequestImpl(ServiceClient.java:123)
    at com.aliyun.oss.common.comm.ServiceClient.sendRequest(ServiceClient.java:68)
    at com.aliyun.oss.internal.OSSOperation.send(OSSOperation.java:94)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:146)
    at com.aliyun.oss.internal.OSSOperation.doOperation(OSSOperation.java:113)
    at com.aliyun.oss.internal.OSSObjectOperation.getObject(OSSObjectOperation.java:229)
    at com.aliyun.oss.OSSClient.getObject(OSSClient.java:629)
    at com.aliyun.oss.OSSClient.getObject(OSSClient.java:617)
    at samples.HelloOSS.main(HelloOSS.java:49)
```

Cause: Connection leaks occur in the connection pool. It may be because `OSSClient.getObject` is not closed after use.

Solution: Check your program and ensure that no connection leaks occur. The following code provides an example on how to close ossObject:

```
// Read an object.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Perform operations in OSS.
// Close your ossObject.
ossObject.close();
```

Connection closure

An error similar to the following is displayed when you use ossClient.getObject:

```
Exception in thread "main" org.apache.http.ConnectionClosedException: Premature end of Content-Length delimited message body (expected: 11990526; rec
eived: 202880
    at org.apache.http.impl.io.ContentLengthInputStream.read(ContentLengthInputStream.java:180)
    at org.apache.http.impl.io.ContentLengthInputStream.read(ContentLengthInputStream.java:200)
    at org.apache.http.impl.io.ContentLengthInputStream.close(ContentLengthInputStream.java:103)
    at org.apache.http.impl.execchain.ResponseEntityProxy.streamClosed(ResponseEntityProxy.java:128)
    at org.apache.http.conn.EofSensorInputStream.checkClose(EofSensorInputStream.java:228)
    at org.apache.http.conn.EofSensorInputStream.close(EofSensorInputStream.java:174)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at com.aliyun.oss.event.ProgressInputStream.close(ProgressInputStream.java:147)
    at java.io.FilterInputStream.close(FilterInputStream.java:181)
    at samples.HelloOSS.main(HelloOSS.java:39)
```

Cause: The interval between the two data reading attempts exceeded one minute. OSS closes the connection if no data is sent or received after more than one minute. This error occurs when your application functions as follows and the time for the application to process data varies: Read part of the data, process the data, read the data, and then process the data.

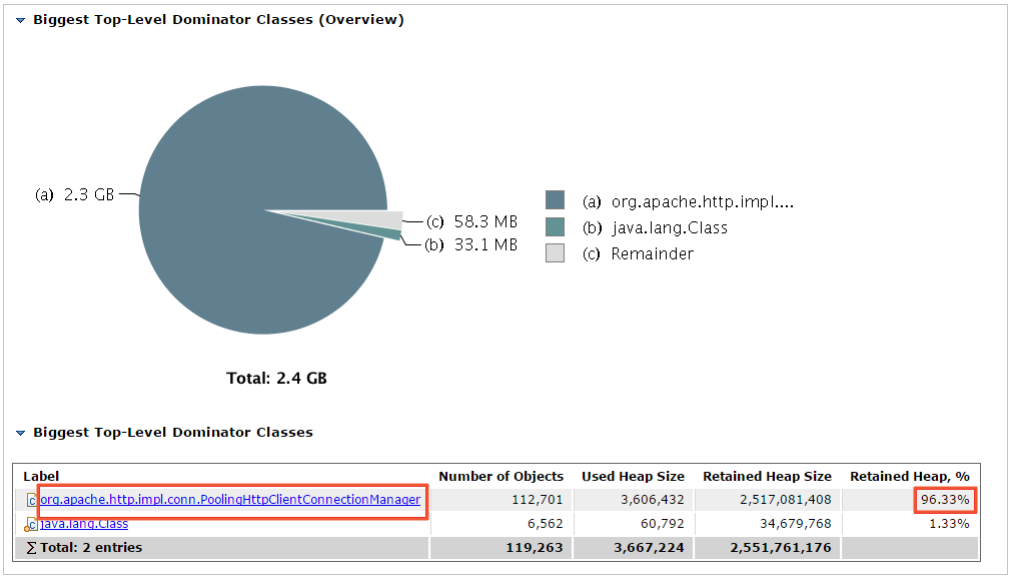
Solution: If you need to read part of the data during unspecified time periods, you can use range download to avoid connection closure during data reading.

Memory leaks

Memory leaks occur when OSS SDK for Java program runs for an extended period of time. This period ranges from several hours to several days based on the amount of access traffic.

Cause: Use the memory analysis tool to analyze the memory usage. We recommend that you use Eclipse Memory Analyzer (MAT) to analyze memory usage. For more information, visit Use MAT for Heap Dump File Analysis.

If the analysis result is similar to that shown in the following figure, PoolingHttpClientConnectionManager takes up 96% of memory. The cause is that the new OSSClient command is run multiple times while ossClient.shutdown is not called. As a result, memory leaks occur.



Solution: You must ensure that ossClient.shutdown is called after you run the new OSSClient command.

InterruptedException occurs when OSSClient.shutdown is called

A similar exception is displayed when OSSClient.shutdown is called:

```
java.lang.InterruptedException: sleep interrupted
        at java.lang.Thread.sleep(Native Method)
        at com.aliyun.oss.common.comm.IdleConnectionReaper.run(IdleConnectionReaper:76)
```

Cause: A background thread IdleConnectionReaper from OSSClient shuts down idle connections in a timely manner. If the shutdown method is called when the IdleConnectionReaper thread is asleep, the preceding exception is displayed.

Solution:

- Upgrade OSS SDK for Java to version 2.3.0 or later because this problem is fixed in OSS SDK for Java V2.3.0.

- For versions earlier than OSS Java SDK version 2.3.0, use the following code to ignore the exception:

```
try {
serviceClient.shutdown();
} catch(Exception e) {
}
```

4.2. Python-SDK

4.2.1. Preface

This topic describes the sample code of OSS SDK for Python in various use cases.

Source code

For the source code of OSS SDK for Python, visit GitHub.

Sample programs

The sample code of OSS SDK for Python is in the examples directory. For more information, visit GitHub. The following table describes the sample files in the sample programs.

Sample file	Description
object_basic.py	Shows you how to perform basic object-related operations such as object upload, download, list, and deletion.
object_extra.py	Shows you how to perform advanced object-related operations such as configure the language and user metadata, copy objects, and append objects.
upload.py	Shows you how to perform advanced object upload operations such as resumable upload and multipart upload.
download.py	Shows you how to perform object download operations such as object download, range download, and resumable download.
object_check.py	Shows you how to verify data during upload and download, including MD5 verification and CRC.
object_progress.py	Shows you how to use a progress bar during upload and download.
object_callback.py	Shows you how to perform upload callback.
object_post.py	Shows you how to use PostObject. The PostObject implementation is independent of the SDK for Python.
sts.py	Shows you how to use STS, including obtain the temporary AccessKey pair by assuming a role and use the temporary AccessKey pair to access OSS.
live_channel.py	Shows you how to use LiveChannel.
image.py	Shows you how to use Image Processing (IMG).
bucket.py	Shows you how to manage buckets such as bucket creation, deletion, and listing.

4.2.2. Installation

This topic describes how to install OSS SDK for Python.

Preparation

The version of OSS SDK for Python in this topic is compatible with Python 2.6, 2.7, 3.3, 3.4, 3.5, and 3.6.

- Install the environment

  For more information about how to install an appropriate version of Python, visit python.

- Check the Python version

  Run the `python -V` command to check Python version.

  In Windows, if a message is displayed, indicating that it is not an internal or external command, edit the environment variable `Path`. Add the installation path of Python and pip. The installation path of pip is the Scripts directory under the installation path of Python.

  > **Note** You may need to restart your computer to ensure that the environment variables take effect.

Download OSS SDK for Python

- Download from GitHub
- Download previous versions

For more information, see API Reference.

Install python-devel

OSS SDK for Python uses the crcmod library to calculate the value of CRC. The crcmod library depends on the Python.h file. If the Python.h file is missing, SDK installation is not affected but the installation of crcmod extensions in C fails. Therefore, the efficiency of operations such as object upload and download is low. Install python-devel if it does not exist.

- When you install Python in Windows or Mac OS X systems, the Python.h file is installed together with Python. Therefore, you do not need to install python-devel.
- When you install Python in CentOS, RHEL, or Fedora systems, run the following command to install python-devel:

```
yum install python-devel
```

- When you install Python in Debian or Ubuntu systems, run the following command to install python-devel:

```
apt-get install python-dev
```

Install OSS SDK for Python

- Use pip

  Run the following command:

```
pip install oss2
```

  > **Note** If pip is not installed in your environment, visit the official pip website to install pip.

- Use the source code

  Click here to visit GitHub and download the OSS SDK for Python package of an appropriate version. Decompress the package. Check whether the setup.py file exists in the directory and then run the following command:

```
python setup.py install
```

Verification

- Verify the version of OSS SDK for Python
  i. Enter `python` in the command line and press Enter to go to the Python environment.
  ii. Run the following command to verify the version of OSS SDK for Python:

```
>>> import oss2
>>> oss2.__version__
'2.0.0'
```

  If an output similar to the preceding example is displayed, OSS SDK for Python version 2.0.0 is installed.

- Verify crcmod

  OSS SDK for Python uses the following methods to calculate the value of CRC: the C extension mode, which uses the crcmod dependent library to call the _crcfunext.so extension library to calculate the CRC value, and the Python-only mode. The performance of the C extension mode is superior than the Python-only mode. The crcmod library is automatically installed together with oss2. For more information about crcmod, visit crcmod introduction.

  When you use OSS SDK for Python, if the object upload and download efficiency remains much lower than that of other tools or other SDKs, the installation of crcmod extensions in C may fail. In this case, the Python-only mode is used to calculate the value of CRC for object upload and download.

  You can use the following methods to determine whether the C extension mode of the crcmod library is installed:

  i. Enter `python` in the command line and press the Enter key to go the Python environment.
  ii. Enter `import crcmod._crcfunext` and press Enter.
    - If no error message is displayed, the crcmod extensions in C are installed.
    - If the following error message is displayed, the installation of crcmod extensions in C failed:

```
>>> import crcmod._crcfunext
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ImportError: No module named _crcfunext
```

      When you compile the crcmod library, the _crcfunext.so file depends on the Python.h file. The error occurs because the Python.h file does not exist in the system, which causes the _crcfunext.so file failed to be generated. Therefore, the Python-only mode is used to calculate the value of CRC. OSS SDK for Python is installed, but the efficiency of operations such as object upload and download is low.

      If the error occurs, perform the following operations:

      a. Run the following command to uninstall the crcmod library:

```
pip uninstall crcmod
```

      b. Install python-devel.
      c. Run the following command to re-install the crcmod library:

```
pip install crcmod
```

      If the crcmod library fails to be installed after you perform the preceding operations, uninstall the crcmod library. Run the following command to view the detailed causes of the installation failures.

```
pip install crcmod -v
```

Uninstall OSS SDK for Python

If OSS SDK for Python fails to be installed, uninstall OSS SDK for Python by using pip and reinstall OSS SDK for Python. Run the following code to uninstall OSS SDK for Python:

```
pip uninstall oss2
```

4.2.3. Quick start

This topic describes how to use OSS SDK for Python to perform routine operations such as create buckets, upload objects, and download objects.

Create a bucket

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set the ACL of the bucket to private.
bucket.create_bucket(oss2.models.BUCKET_ACL_PRIVATE)
```

For more information about how to obtain an AccessKey ID and AccessKey secret, see Obtain an AccessKey pair.

For more information about how to obtain an endpoint, see Obtain an endpoint.

Upload objects

The following code provides an example on how to upload objects to OSS by using stream upload:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# <yourObjectName> indicates the complete path of the object you want to upload to OSS, and must include the extension of the object name. Example: a
bc/efg/123.jpg.
# <yourLocalFile> consists of a local file path and an object name with extension. Example: /users/local/myfile.txt.
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

Download objects

The following code provides an example on how to download a specified object to a local directory:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# <yourObjectName> indicates the complete path of the object you want to download from OSS, and must include the extension of the object name. Exampl
e: abc/efg/123.jpg.
# <yourLocalFile> consists of a local file path and an object name with extension. Example: /users/local/myfile.txt.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

List objects

The following code provides an example on how to list 10 objects in a specified bucket:

```
# -*- coding: utf-8 -*-
import oss2
from itertools import islice
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# oss2.ObjectIteratorr is used to traverse objects.
for b in islice(oss2.ObjectIterator(bucket), 10):
    print(b.key)
```

Delete objects

The following code provides an example on how to delete an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# <yourObjectName> indicates the complete path of the object you want to delete from OSS, and must include the extension of the object name. Example:
abc/efg/123.jpg.
bucket.delete_object('<yourObjectName>')
```

4.2.4. Initialization

This topic describes how to initialize OSS SDK for Python.

Most operations of OSS SDK for Python are performed based on the oss2.Service and oss2.Bucket classes.

- oss2.Service is used to list buckets.
- oss2.Bucket is used to upload, download, and delete objects as well as configure buckets.

You must specify the endpoint when you initiate the two classes. oss2.Service does not support access to buckets by using custom domain names that are mapped to the buckets. For more information about endpoints, see Obtain an endpoint.

Initialize oss2.Service

For more information, see List buckets.

Initialize oss2.Bucket

- Use the endpoint of the region where the bucket is located to initialize the class

  The following code provides an example on how to initialize oss2.Bucket by using the endpoint of the region where the bucket is located:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
endpoint = 'https://oss-cn-hangzhou.aliyuncs.com'
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>')
```

- Configure the connection timeout period

  The following code provides an example on how to configure the connection timeout period:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
endpoint = 'https://oss-cn-hangzhou.aliyuncs.com'
# Set the connection timeout period to 30 seconds.
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', connect_timeout=30)
```

- Disable CRC

  CRC is enabled by default when you upload or download objects to ensure data integrity. The following code provides an example on how to disable CRC:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
endpoint = 'https://oss-cn-hangzhou.aliyuncs.com'
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', enable_crc=False)
```

> **Warning** We recommend that you do not disable CRC. If you disable CRC, data integrity may be affected during object uploads and downloads.

4.2.5. Buckets

4.2.5.1. Create a bucket

A bucket is a container used to store objects in Object Storage Service (OSS). Each object is contained in a bucket. This topic describes how to create a bucket.

The following code provides an example on how to create a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# After you specify the endpoint and bucket name, you can create a bucket in the specified region. In this example, the endpoint of the China (Hangzh
ou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Create the bucket. By default, the ACL of the created bucket is private.
bucket.create_bucket()
```

4.2.5.2. List buckets

A bucket is a container that is used to store objects in Object Storage Service (OSS). All objects in OSS are contained in buckets. Bucket names are listed in alphabetical order. You can list buckets that belong to the current Alibaba Cloud account in all regions and meet the specific conditions.

The following sample code provides an example on how to list all buckets that belong to an Alibaba Cloud account:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
service = oss2.Service(auth, 'http://oss-cn-hangzhou.aliyuncs.com')
print([b.name for b in oss2.BucketIterator(service)])
```

4.2.5.3. Determine whether a bucket exists

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to determine whether a bucket exists.

The following code provides an example on how to determine whether a bucket exists:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
def does_bucket_exist(bucket):
    try:
        bucket.get_bucket_info()
    except oss2.exceptions.NoSuchBucket:
        return False
    except:
        raise
    return True
exist = does_bucket_exist(bucket)
# If the returned value is true, a bucket with the specified name exists. If the returned value is false, a bucket with the specified name does not e
xist.
if exist:
    print('bucket exist')
else:
    print('bucket not exist')
```

4.2.5.4. Manage bucket ACLs

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to configure and query the access control list (ACL) of a bucket.

Configure the ACL of a bucket

The following table describes the ACLs that you can configure for a bucket.

ACL	Description	Value
Private	Only the bucket owner and authorized users have read and write permissions on objects in the bucket. Other users cannot access the objects in the bucket.	oss2.BUCKET_ACL_PRIVATE
Public read	Only the bucket owner and authorized users have read and write permissions on objects in the bucket. Other users have only read permissions on the objects in the bucket. Exercise caution when you set the ACL of the bucket to this value.	oss2.BUCKET_ACL_PUBLIC_READ
Public read/write	All users have read and write permissions on objects in the bucket. Exercise caution when you set the ACL of the bucket to this value.	oss2.BUCKET_ACL_PUBLIC_READ_WRITE

The following code provides an example on how to configure the ACL of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set the ACL of the bucket to private.
bucket.put_bucket_acl(oss2.BUCKET_ACL_PRIVATE)
```

Query the ACL of a bucket

The following code provides an example on how to query the ACL of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
print(bucket.get_bucket_acl().acl)
```

4.2.5.5. Delete buckets

If you no longer use a bucket, delete the bucket to stop unnecessary charges.

Before you delete a bucket, ensure that all objects, LiveChannels, and parts that are generated during multipart upload in the bucket are deleted. The following code provides an example on how to delete buckets:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
try:
    # Delete the bucket.
    bucket.delete_bucket()
except oss2.exceptions.BucketNotEmpty:
    print('bucket is not empty.')
except oss2.exceptions.NoSuchBucket:
    print('bucket does not exist')
```

To delete a bucket that is not empty, you must clear the bucket by listing and deleting objects in the bucket. If a multipart upload task is ongoing, you must terminate the task before you can delete the bucket.

4.2.5.6. Manage lifecycle rules

Object Storage Service (OSS) allows you to configure lifecycle rules to delete expired objects and parts or convert the storage class of expired objects to Infrequent Access (IA) or Archive to reduce your storage costs. This topic describes how to manage lifecycle rules for buckets.

Background

Each lifecycle rule contains the following information:

- Policy: specifies the mode to match objects and parts.
  - Match by prefix: matches objects and parts by prefix. You can create multiple rules to configure different prefixes. Each prefix must be unique.
  - Match by bucket: matches all objects and parts contained in the bucket. You can create only one rule if you select this method.
- Object lifecycle policy: specifies the validity period or expiration data and the operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired objects are deleted.
  - Expiration date: specifies a date. Objects that are last modified before this date expire and are deleted.
- Part lifecycle policy: specifies the validity period or expiration time and the delete operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted.
  - Expiration date: specifies a date. Parts that are last modified before this date expire and are deleted.

Lifecycle rules also apply to the parts uploaded by using uploadPart. In this case, the last modified time of an object is the time the multipart upload task is initiated.

Configure lifecycle rules for a bucket

The following code provides an example on how to configure lifecycle rules for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import LifecycleExpiration, LifecycleRule, BucketLifecycle,AbortMultipartUpload
import datetime
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Specify that objects expire three days after they are last modified.
rule1 = LifecycleRule('rule1', 'tests/',
                      status=LifecycleRule.ENABLED,
                      expiration=LifecycleExpiration(days=3))
# Specify an expiration date. Objects that are created before this date expires.
rule2 = LifecycleRule('rule2', 'logging-',
                      status=LifecycleRule.DISABLED,
expiration=LifecycleExpiration(created_before_date=datetime.date(2018, 12, 12)))
# Specify that parts expire three days after they are last modified.
rule3 = LifecycleRule('rule3', 'tests1/',
                      status=LifecycleRule.ENABLED,
            abort_multipart_upload=AbortMultipartUpload(days=3))
# Specify an expiration date. Parts that are last modified before this date expires.
rule4 = LifecycleRule('rule4', 'logging1-',
                      status=LifecycleRule.DISABLED,
                      abort_multipart_upload = AbortMultipartUpload(created_before_date=datetime.date(2018, 12, 12)))
lifecycle = BucketLifecycle([rule1, rule2,rule3,rule4])
bucket.put_bucket_lifecycle(lifecycle)
```

Query the lifecycle rules of a bucket

The following code provides an example on how to query the lifecycle rules of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
lifecycle = bucket.get_bucket_lifecycle()
for rule in lifecycle.rules:
    if rule.abort_multipart_upload is None:
        print('id={0}, prefix={1}, status={2}, days={3}, created_before_date={4}'
              .format(rule.id, rule.prefix, rule.status,
                      rule.expiration.days,
                      rule.expiration.created_before_date))
    else:
        print('id={0}, prefix={1}, status={2}, days={3}, created_before_date={4}'
              .format(rule.id, rule.prefix, rule.status,
                      rule.abort_multipart_upload.days,
                      rule.abort_multipart_upload.created_before_date))
```

Clear the lifecycle rules of a bucket

The following code provides an example on how to clear the lifecycle rules of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.delete_bucket_lifecycle()
# An exception occurs when you query the lifecycle rules of the bucket again.
try:
    lifecycle = bucket.get_bucket_lifecycle()
except oss2.exceptions.NoSuchLifecycle:
    print('lifecycle is not configured')
```

4.2.5.7. Static website hosting

You can enable static website hosting for your bucket. After you enable static website hosting for a bucket, you can visit the website by directly accessing the bucket domain name. In this case, you are redirected to the index page or error page that are specified for the hosted website.

Configure static website hosting

The following code provides an example on how to configure static website hosting for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import BucketWebsite
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Enable static website hosting for the bucket. Set the index page to index.html and the error page to error.html.
bucket.put_bucket_website(BucketWebsite('index.html', 'error.html'))
```

Query the static website hosting configurations of a bucket

The following code provides an example on how to query the static website hosting configurations of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
try:
    # If static website hosting is not enabled for the bucket, OSS returns NoSuchWebsite for the get_bucket_website method.
    website = bucket.get_bucket_website()
    print('Index file is {0}, error file is {1}'.format(website.index_file, website.error_file))
except oss2.exceptions.NoSuchWebsite as e:
    print('Website is not configured, request_id={0}'.format(e.request_id))
```

Delete the static website hosting configurations of a bucket

The following code provides an example on how to delete the static website hosting configurations of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.delete_bucket_website()
```

4.2.5.8. CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to different regions. Object Storage Service (OSS) provides CORS API operations to control cross-origin access.

Configure CORS rules

The following code provides an example on how to configure CORS rules for a specific bucket:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import BucketCors, CorsRule
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
rule = CorsRule(allowed_origins=['*'],
                allowed_methods=['GET', 'HEAD'],
                allowed_headers=['*'],
                max_age_seconds=1000)
# The existing rules are overwritten.
bucket.put_bucket_cors(BucketCors([rule]))
```

Query CORS rules

The following code provides an example on how to query the CORS rules that are configured for a specific bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
try:
    cors = bucket.get_bucket_cors()
except oss2.exceptions.NoSuchCors:
    print('cors is not set')
else:
    for rule in cors.rules:
        print('AllowedOrigins={0}'.format(rule.allowed_origins))
        print('AllowedMethods={0}'.format(rule.allowed_methods))
        print('AllowedHeaders={0}'.format(rule.allowed_headers))
        print('ExposeHeaders={0}'.format(rule.expose_headers))
        print('MaxAgeSeconds={0}'.format(rule.max_age_seconds))
```

Delete CORS rules

The following code provides an example on how to delete the CORS rules that are configured for a specific bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.delete_bucket_cors()
```

4.2.5.9. Hotlink protection

Object Storage Service (OSS) provides hotlink protection to prevent unauthorized domain names from accessing your OSS data. You can specify the Referer field in HTTP request headers to configure hotlink protection.

To configure hotlink protection, you must configure the following parameters:

- Referer Whitelist: specifies that only specified domain names are allowed to access your resources.
- Allow Empty Referer: determines whether requests that contain an empty Referer field are allowed. If you specify that an empty Referer field is not allowed, only HTTP and HTTPS requests that contain an allowed Referer field can access your OSS resources.

Configure hotlink protection for a bucket

The following code provides an example on how to configure a Referer whitelist for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import BucketReferer
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set BucketReferer to True. True indicates that empty Referer fields are allowed and False indicates that empty Referer fields are not allowed. Then
, configure the Referer whitelist.
bucket.put_bucket_referer(BucketReferer(True, ['http://aliyun.com', 'http://*.aliyuncs.com']))
```

Query the hotlink protection configurations of a bucket

The following code provides an example on how to query the hotlink protection configuration of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
config = bucket.get_bucket_referer()
print('allow empty referer={0}, referers={1}'.format(config.allow_empty_referer, config.referers))
```

Delete the hotlink protection configurations of a bucket

The following code provides an example on how to delete the hotlink protection configurations of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# The Referer whitelist of a bucket cannot be directly cleared. You must create a Referer whitelist that allows empty Referer fields to overwrite the
existing Referer whitelist.
bucket.put_bucket_referer(BucketReferer(True, []))
```

4.2.5.10. Configure logging

You can enable logging to record access to a bucket in log objects, which are stored in a specified bucket.

The names of log objects are in the following format: `<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString`.

Enable logging for a bucket

The following code provides an example on how to enable logging for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import BucketLogging
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Enable logging for the bucket. Save the log objects to the current bucket, and set the directory in which the log objects are stored to 'logging/'.

logging = bucket.put_bucket_logging(BucketLogging(bucket.bucket_name, 'logging/'))
if logging.status == 200:
    print("Enable access logging")
else:
    print("request_id :", logging.request_id)
    print("resp : ", logging.resp.response)
```

Query the logging configurations of a bucket

The following code provides an example on how to query the logging configurations of a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
logging = bucket.get_bucket_logging()
print('TargetBucket={0}, TargetPrefix={1}'.format(logging.target_bucket, logging.target_prefix))
```

Disable logging for a bucket

The following code provides an example on how to disable logging for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
logging = bucket.delete_bucket_logging()
if logging.status == 204:
    print("Disable access logging")
else:
    print("request_id :", logging.request_id)
    print("resp : ", logging.resp.response)
```

4.2.6. Upload objects

4.2.6.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for Python provides a variety of methods to upload objects.

OSS SDK for Python supports the following object upload methods:

- Simple upload: uploads an object up to 5 GB in size.
- Append upload: uploads an object up to 5 GB in size.
- Resumable upload: supports concurrent upload and resumable upload in which you can define the size of each part. This method is suitable for the upload of large objects. You can use this method to upload an object up to 48.8 TB in size.
- Multipart upload: uploads an object up to 48.8 TB in size. This method is suitable for the upload of large objects.

During object upload, you can configure object metadata and view upload progress by using a progress bar. For more information, see Use progress bars. After the object is uploaded, you can perform upload callback. For more information, see Upload callback.

4.2.6.2. Simple upload

You can use simple upload to upload the following types of data: strings, bytes, Unicode characters, network streams, and local files.

You can use the bucket.put_object method to upload an object. The following table describes the types of inputs supported by this method.

Type	Upload method
Strings	Strings are uploaded directly.
Bytes	Bytes are uploaded directly.
Unicode	Unicode characters are converted to bytes encoded in UTF-8 for upload.
Local files	Local files are uploaded as file objects. File objects must be opened in binary mode such as the rb mode.
Network streams	Network streams are uploaded as iterable objects by using chunked encoding.

Note If you upload an object with the same name as that of an accessible existing object, the existing object is overwritten by the uploaded object and 200 OK is returned.

Upload strings

The following code provides an example on how to upload a string to an OSS object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Obtain the return result.
result = bucket.put_object('<yourObjectName>', 'content of object')
# Display the returned HTTP status code.
print('http status: {0}'.format(result.status))
# Display the request ID. A request ID uniquely identifies a request. We recommend that you add this parameter in the logs.
print('request_id: {0}'.format(result.request_id))
# Obtain the ETag value returned by the put_object method.
print('ETag: {0}'.format(result.etag))
# Display the HTTP response headers.
print('date: {0}'.format(result.headers['date']))
```

Upload bytes

The following code provides an example on how to upload bytes:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.put_object('<yourObjectName>', b'content of object')
```

Upload Unicode characters

The following code provides an example on how to upload Unicode:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.put_object('<yourObjectName>', u'content of object')
```

Upload a local file

The following code provides an example on how to upload a local file to OSS:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# The object must be opened in binary mode because the number of bytes in the object must be available before the upload.
with open('<yourLocalFile>', 'rb') as fileobj:
    # Use the seek method to read data from byte 1,000 of the file. The data is uploaded from byte 1,000 to the last byte of the local file.
    fileobj.seek(1000, os.SEEK_SET)
    # Use the tell method to obtain the current position.
    current = fileobj.tell()
    bucket.put_object('<yourObjectName>', fileobj)
```

OSS SDK for Python also provides a more convenient method to upload local files:

```
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

Upload a network stream

The following code provides an example on how to upload a network stream:

```
# -*- coding: utf-8 -*-
import oss2
import requests
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# The requests.get method returns an iterable object. OSS SDK for Python uploads the object by means of chunked encoding.
input = requests.get('http://www.aliyun.com')
bucket.put_object('<yourObjectName>', input)
```

4.2.6.3. Append upload

Object Storage Service (OSS) allows you to use append upload to append content directly to the end of an object. You can apply this method only to appendable objects.

The following code provides an example on how to upload local files to OSS by using append upload:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set the position from which the first append operation starts to 0.
result = bucket.append_object('<yourObjectName>', 0, 'content of first append')
# If you have appended content to the object, you can obtain the position from which the operation starts this time from the next_position field in t
he response returned by the last operation or by using the bucket.head_object method.
bucket.append_object('<yourObjectName>', result.next_position, 'content of second append')
```

If the object already exists, exceptions occur in the following cases:

- If the object is not appendable, ObjectNotAppendable is returned.
- If the object is appendable but the append position does not match the current object size, PositionNotEqualToLength is returned.

4.2.6.4. Resumable upload

If the system stops responding during multipart upload, you can resume the upload by using the ListMultipartUploads and ListParts operations to retrieve all multipart upload tasks on an object and list the uploaded parts in each task. This method allows an upload task to be resumed from the last uploaded part.

You can use the oss2.resumable_upload method to perform resumable upload for a specified object. The following table describes the parameters you can configure for the oss2.resumable_upload method.

Parameter	Description	Required	Default value
bucket	The name of the bucket.	Yes	None
key	The name of the object.	Yes	None
filename	The local file path from which to upload the object.	Yes	None
store	The directory where the checkpoint information is stored.	No	The *.py-oss-upload* directory created in the HOME directory.
headers	The HTTP headers.	No	None

Parameter	Description	Required	Default value
multipart_threshold	The object length threshold for multipart upload. If the object length exceeds the threshold, multipart upload is used.	No	10 MB
part_size	The size of the part.	No	Automatically calculated
progress_callback	The callback function to return upload progress.	No	None
num_threads	The number of concurrent upload threads.	No	1

The following code provides an example on how to perform resumable upload:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Use multipart upload when the object length is greater than or equal to the multipart_threshold value. The multipart_threshold parameter is optiona
l, and the default value of multipart_threshold is 10 MB. If you do not specify a directory by using the store parameter, the .py-oss-upload director
y is created in the HOME directory to store the checkpoint information.
oss2.resumable_upload(bucket, '<yourObjectName>', '<yourLocalFile>')
```

OSS SDK for Python version 2.1.0 and later support resumable upload by using optional parameters. The following code provides an example on how to use optional parameter in resumable upload:

```
# If you specify a directory by using the store parameter, the checkpoint information is stored in the specified directory. If you use num_threads to
specify the number of concurrent upload threads, ensure that oss2.defaults.connection_pool_size is set to a value greater than the number of concurre
nt upload threads. The default number of concurrent upload threads is 1.
oss2.resumable_upload(bucket, '<yourObjectName>', '<yourLocalFile>',
    store=oss2.ResumableStore(root='/tmp'),
    multipart_threshold=100*1024,
    part_size=100*1024,
    num_threads=4)
```

Note

- Multiple threads are used in resumable upload. Therefore, you do not need to encapsulate multiple upload threads when you call the resumable upload method. Otherwise, data may be transferred repeatedly.
- We recommend that you increase the part size when network conditions are good, and decrease the part size when network conditions are poor.

4.2.6.5. Multipart upload

OSS allows you to use multipart upload to split an object into several parts and upload them separately. After all parts are uploaded, you can combine the uploaded parts into a complete object to complete the upload.

To implement multipart upload, perform the following operations:

The following code provides an example on how to perform multipart upload:

```
# -*- coding: utf-8 -*-
import os
from oss2 import SizedFileAdapter, determine_part_size
from oss2.models import PartInfo
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
key = '<yourObjectName>'
filename = '<yourLocalFile>'
total_size = os.path.getsize(filename)
# Use the determine_part_size method to determine the part size.
part_size = determine_part_size(total_size, preferred_size=100 * 1024)
# Initiate a multipart upload task.
# To set the storage class for the object when you initiate the multipart upload task, set the headers parameter in the init_multipart_upload functio
n.
# headers = dict()
# headers["x-oss-storage-class"] = "Standard"
# upload_id = bucket.init_multipart_upload(key, headers=headers).upload_id
upload_id = bucket.init_multipart_upload(key).upload_id
parts = []
# Upload the parts one by one.
with open(filename, 'rb') as fileobj:
    part_number = 1
    offset = 0
    while offset < total_size:
        num_to_upload = min(part_size, total_size - offset)
        # The SizedFileAdapter(fileobj, size) method generates a new object, and resets the append position.
        result = bucket.upload_part(key, upload_id, part_number,
                                    SizedFileAdapter(fileobj, num_to_upload))
        parts.append(PartInfo(part_number, result.etag))
        offset += num_to_upload
        part_number += 1
# Complete the multipart upload task.
# To configure ACL for the object when you complete the multipart upload task, set the headers parameter in the complete_multipart_upload function.
# headers = dict()
# headers["x-oss-object-acl"] = oss2.OBJECT_ACL_PRIVATE
# bucket.complete_multipart_upload(key, upload_id, parts, headers=headers)
bucket.complete_multipart_upload(key, upload_id, parts)
# Verify multipart upload.
with open(filename, 'rb') as fileobj:
    assert bucket.get_object(key).read() == fileobj.read()
```

Cancel a multipart upload task

You can call bucket.abort_multipart_upload to cancel a multipart upload task. If you cancel a multipart upload task, you cannot use the upload ID to perform any operations on the parts. The uploaded parts are deleted.

The following code provides an example on how to cancel a multipart upload task:

```
# -*- coding: utf-8 -*-
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
key = '<yourObjectName>'
# Obtain the upload ID returned by the init_multipart_upload operation.
upload_id = '<yourUploadId>'
# When you cancel the multipart upload task by using the specified upload ID, the uploaded parts are deleted.
bucket.abort_multipart_upload(key, upload_id)
```

List uploaded parts

The following code provides an example on how to list uploaded parts:

```
# -*- coding: utf-8 -*-
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
key = '<yourObjectName>'
# Obtain the upload ID returned by the init_multipart_upload operation.
upload_id = '<yourUploadId>'
# List uploaded parts by using the specified upload ID.
for part_info in oss2.PartIterator(bucket, key, upload_id):
    print('part_number:', part_info.part_number)
    print('etag:', part_info.etag)
    print('size:', part_info.size)
```

List multipart upload tasks

- List multipart upload tasks of a specified object

The following code provides an example on how to list multipart upload tasks of a specified object:

```
# -*- coding: utf-8 -*-
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
key = '<yourObjectName>'
# List all multipart upload tasks of the object. Each time init_multipart_upload is called for the same object, different upload_id values are retu
rned.
# One upload ID corresponds to one multipart upload task.
for upload_info in oss2.ObjectUploadIterator(bucket, key):
    print('key:', upload_info.key)
    print('upload_id:', upload_info.upload_id)
```

- List all multipart upload tasks in a bucket

  The following code provides an example on how to list all multipart upload tasks in a bucket:

```
# -*- coding: utf-8 -*-
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# List all multipart upload tasks in the bucket.
for upload_info in oss2.MultipartUploadIterator(bucket):
    print('key:', upload_info.key)
    print('upload_id:', upload_info.upload_id)
```

- List all multipart upload tasks for objects whose names contain a specified prefix in a bucket

  The following code provides an example on how to list multipart upload tasks for objects whose names contain a specified prefix in a bucket:

```
# -*- coding: utf-8 -*-
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# List multipart upload tasks for objects whose names contain the prefix test in the bucket.
for upload_info in oss2.MultipartUploadIterator(bucket, prefix='test'):
    print('key:', upload_info.key)
    print('upload_id:', upload_info.upload_id)
```

4.2.6.6. Use progress bars

Progress bars are used to indicate the progress of an upload or download.

For the complete code of progress bar, visit GitHub.

The following code uses bucket.put_object as an example to describe how to use a progress bars:

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# If the length of the data to upload cannot be determined, the value of total_bytes is None.
def percentage(consumed_bytes, total_bytes):
    if total_bytes:
        rate = int(100 * (float(consumed_bytes) / float(total_bytes)))
        print('\r{0}% '.format(rate), end='')
        sys.stdout.flush()
# progress_callback is an optional parameter used to return progress information.
bucket.put_object('<yourObjectName>', 'a'*1024*1024, progress_callback=percentage)
```

4.2.6.7. Upload callback

This topic describes how to use upload callback.

For the complete code used to implement upload callback, visit GitHub.

You can use upload callback when you use put_object, put_object_from_file, and complete_multipart_upload.

The following code provides an example on how to use upload callback.

```
# -*- coding: utf-8 -*-
import json
import base64
import os
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Configure the upload callback parameter.
callback_dict = {}
# Configure the IP address of the server to which you want to send the callback request. Example: http://oss-demo.aliyuncs.com:23450 or http://127.0.
0.1:9090.
callback_dict['callbackUrl'] = 'http://oss-demo.aliyuncs.com:23450'
# Optional. Configure the value of the host field carried in the callback request header.
# callback_dict['callbackHost'] = 'yourCallbackHost'
# Configure the value of the body field included in the callback request.
callback_dict['callbackBody'] = 'filename=${object}&size=${size}&mimeType=${mimeType}'
# Configure Content-Type for the callback request.
callback_dict['callbackBodyType'] = 'application/x-www-form-urlencoded'
# The callback parameter is in the JSON format and encoded in Base64.
callback_param = json.dumps(callback_dict).strip()
base64_callback_body = base64.b64encode(callback_param)
# Configure the header with the encoded parameter and send the header to OSS.
headers = {'x-oss-callback': base64_callback_body}
# Upload the object and perform upload callback.
result = bucket.put_object('<yourObjectName>', 'a'*1024*1024, headers)
```

4.2.7. Download objects

4.2.7.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for Python provides the following object download methods:

You can use one of the following methods to download an object:

- Streaming download
- Download to local files
- Range download
- Resumable download

You can view the download progress by using a progress bar. For more information, see Use progress bars.

4.2.7.2. Streaming download

If you must download a large object or if the download requires a long period of time to complete, you can perform streaming download to download the object in increments.

The following code provides an example on how to download an object by using streaming download:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# The value returned by bucket.get_object is a file-like object that can be iterated.
object_stream = bucket.get_object('<yourObjectName>')
print(object_stream.read())
# You must call read() to read the object from the stream that is returned by get_object before you can calculate the CRC-64 checksum of the object.
if object_stream.client_crc != object_stream.server_crc:
    print "The CRC checksum between client and server is inconsistent!"
```

The following code provides an example on how to download data to a local file in streaming mode:

```
import shutil
# object_stream is a file-like object. You can use the shutil.copyfileobj method to download the data to your local file.
object_stream = bucket.get_object('<yourObjectName>')
with open('<yourLocalFile>', 'wb') as local_fileobj:
    shutil.copyfileobj(object_stream, local_fileobj)
```

The following code provides an example on how to copy data to another object in streaming mode:

```
# object_stream is an object that can be iterated. You can copy object_stream to another object in streaming mode.
object_stream = bucket.get_object('<yourObjectName>')
bucket.put_object('<yourBackupObjectName>', object_stream)
```

4.2.7.3. Download to local files

This topic describes how to download an object to a local file.

The following code provides an example on how to download a specified object to a local file:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Download the object to a local file. If the specified local file already exists, the downloaded object replaces the local file. If the specified lo
cal file does not exist, it is created.
# <yourLocalFile> consists of a local file path and an object name with an extension. Example: /users/local/myfile.txt.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

4.2.7.4. Range download

You can use range download to download a specified range of data of an object.

Specify a valid range to download data

The following code provides an example on how to specify a valid range to download data:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# If an object is 1,000 bytes in size, the valid range is from byte 0 to byte 999.
# Query data that is within the range from byte 0 to byte 999, which includes a total of 1,000 bytes. If the specified range is invalid, the entire o
bject is downloaded. For example, if the specified range includes a negative number, or if the specified value is larger than the object size, all da
ta of the object is downloaded.
object_stream = bucket.get_object('<yourObjectName>', byte_range=(0, 999))
```

Specify an invalid range

The valid range for an object whose size is 1,000 bytes is from byte 0 to byte 999. If the specified range is not within the range, the range does not take effect. Then, OSS returns the 200 HTTP status code and the data of the entire object. The following list provides examples of invalid requests and the returned results:

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.

Specify the compatible behavior to download data by range

If you add x-oss-range-behavior:standard to the request header, the download behavior is modified when the specified range is not within the valid range. Example: Use range download to download an object whose size is 1,000 bytes.

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns HTTP status code 206 and the data that is within the range from byte 500 to byte 999.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns HTTP status code 416 and error code InvalidRange.

The following code provides an example on how to specify the compatible behavior to download data by range:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Create an object whose size is 1,000 bytes.
object_name = 'rangeTest.txt'
content = 'a' * 1000
bucket.put_object(object_name, content)
headers = {'x-oss-range-behavior':'standard'}
# Specify the compatible behavior.
# When the end value of the range is invalid, OSS returns the data that is within the range from byte 500 to byte 999 and HTTP status code 206.
object_stream = bucket.get_object(object_name, byte_range=(500, 2000), headers=headers)
print('standard get 500~2000 http status code:', object_stream.status)
print('standard get 500~2000 contnet_length:', object_stream.content_length)
try:
    # When the start value of the range is invalid, OSS returns HTTP status code 416 and the InvalidRange error code.
    object_stream = bucket.get_object(object_name, byte_range=(1000, 2000), headers=headers)
except oss2.exceptions.ServerError as e:
    print('standard get 1000~2000 http status code:', e.status)
    print('standard get 1000~2000 error code:', e.code)
```

4.2.7.5. Resumable download

You may fail to download a large object if the network is unstable or if the program stops responding. In some cases, you may still fail to download the object even after multiple attempts. To solve this problem, OSS SDK for Java provides the resumable download feature.

You can perform the following steps to perform resumable download:

1. Create a temporary local file with a name that consists of the original object name and a random suffix.
2. Read the object based on the range specified in the Range header in the HTTP request and write the read content to the corresponding position of the temporary local file.
3. After the download is complete, rename the temporary file as the destination file. If the destination file already exists, the downloaded data overwrites the data in the existing file. Otherwise, a new file is created.

You can use the oss2.resumable_download method to perform resumable download. The following table describes the parameters used in this method.

Parameter	Description	Required	Default value
bucket	The name of the bucket.	Yes	None
key	The name of the object.	Yes	None
filename	The path to the local file. The local file to which the object is downloaded.	Yes	None
store	The file that records the results of multipart download. This file stores information about download progress. If you fail to download a part, the download continues based on the recorded progress.	No	The *.py-oss-upload* directory created in the HOME directory.
multipart_threshold	The threshold for the object size. If the object size exceeds the threshold, resumable download is enabled.	No	10 MB
part_size	The size of the part.	No	Automatically calculated.
progress_callback	The callback function used to return download progress information.	No	None
num_threads	The number of threads for simultaneous downloads.	No	1

Run the following code for resumable download:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
oss2.resumable_download(bucket, '<yourObjectName>', '<yourLocalFile>')
```

You can configure optional parameters for resumable upload when you use OSS SDK for Python V2.1.0 or later version. The following code provides you an example:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set oss2.defaults.connection_pool_size to a value greater than or equal to the number of threads. Then set part_size to a value greater than or equ
al to the oss2.defaults.multiget_part_size value.
oss2.resumable_download(bucket, '<yourObjectName>', '<yourLocalFile>',
  store=oss2.ResumableDownloadStore(root='/tmp'),
  multiget_threshold=20*1024*1024,
  part_size=10*1024*1024,
  num_threads=3)
```

Note Do not use multiple programs or threads to call the method simultaneously to download the same object to a same destination file. This is because one piece of checkpoint information overwrites another on the local disk, or one temporary file name conflicts with another.

4.2.7.6. Use progress bars

Progress bars are used to indicate the progress of an upload or download.

For the complete code of progress bar, visit GitHub.

The following code provides an example on how to use a progress bar in object download performed by using the bucket.get_object_to_file method:

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# If Content-Length is not configured in the HTTP response header, the value of total_bytes is None.
def percentage(consumed_bytes, total_bytes):
    if total_bytes:
        rate = int(100 * (float(consumed_bytes) / float(total_bytes)))
        print('\r{0}% '.format(rate), end='')
        sys.stdout.flush()
# progress_callback is an optional parameter used to implement a progress bar.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>', progress_callback=percentage)
```

4.2.8. Manage objects

4.2.8.1. Overview

You can manage objects in a bucket by using a variety of operations such as SetObjectAcl, GetObjectAcl, ListObjects, DeleteObject, and CopyObject.

By using the preceding operations, you can perform the following operations:

- Determine whether an object exists
- Manage object ACLs
- Manage object metadata
- List objects
- Query objects
- Delete objects
- Copy an object

4.2.8.2. Determine whether an object exists

This topic describes how to determine whether an object exists.

The following code provides an example on how to determine whether an object exists:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
exist = bucket.object_exists('<yourObjectName>')
# If the returned value is true, the specified object exists. If the returned value is false, the specified object does not exist.
if exist:
    print('object exist')
else:
    print('object not exist')
```

4.2.8.3. Manage object ACLs

This topic describes how to manage the access control list (ACL) of an object.

The following table describes the ACLs that you can configure for an object.

ACL	Description	Value
Inherited from the bucket	The ACL of the object is the same as the ACL of the bucket in which the object is stored.	oss2.OBJECT_ACL_DEFAULT
Private	Only the object owner and authorized users are granted read and write permissions on the object.	oss2.OBJECT_ACL_PRIVATE
Public read	Only the object owner and authorized users are granted read and write permissions on the object. Other users are granted only read permissions on the object. Exercise caution when you set the ACL of the object to this value.	oss2.OBJECT_ACL_PUBLIC_READ
Public read/write	All users are granted read and write permissions on the object. Exercise caution when you set the ACL of the object to this value.	oss2.OBJECT_ACL_PUBLIC_READ_WRITE

The ACL of the object takes precedence over the ACL of the bucket. For example, if the ACL of an object in a private bucket is public read/write, all users have read and write permissions on the object. If the ACL of an object is not configured, the ACL of the object is the same as the ACL of the bucket in which the object is stored.

Configure the ACL of an object

The following code provides an example on how to configure the ACL of an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Configure the object ACL.
bucket.put_object_acl('<yourObjectName>', oss2.OBJECT_ACL_PUBLIC_READ)
```

Query the ACL of an object

The following code provides an example on how to query the ACL of a specified object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
print(bucket.get_object_acl('<yourObjectName>').acl)
```

4.2.8.4. Manage object metadata

Object metadata includes HTTP headers and user metadata.

Configure HTTP headers

The following code provides an example on how to configure the HTTP headers of an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.put_object('<yourObjectName>', '{"age": 1}', headers={'Content-Type': 'application/json; charset=utf-8'})
```

For more information about HTTP headers, see RFC 2616.

Configure user metadata

You can configure the user metadata of an object to describe the object.

The following code provides an example on how to configure the user metadata of an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.put_object('<yourObjectName>', 'a novel', headers={'x-oss-meta-author': 'O. Henry', 'Content-Type': 'application/json; charset=utf-8'})
```

Modify object metadata

The following code provides an example on how to modify the metadata of an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-author': 'O. Henry'})
# Each time you call bucket.update_object_meta, the existing user metadata is updated.
bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-price': '100 dollar'})
```

The following code provides an example on how to modify object metadata such as Content-Type:

```
bucket.update_object_meta('<yourObjectName>', {'x-oss-meta-author': 'O. Henry'})
# Each time you call bucket.update_object_meta, the existing user metadata is updated.
bucket.update_object_meta('<yourObjectName>', {'Content-Type': 'text/plain'})
```

Query object metadata

You can use the API operations provided by OSS SDK for Python to query object metadata.

Method	Description	Advantage
get_object_meta	Obtains part of object metadata such as the ETag, size, and last modified time of the object.	More lightweight and faster
head_object	Obtains all object metadata.	None

The following code provides an example on how to query object metadata:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
// Query part of the object metadata.
simplifiedmeta = bucket.get_object_meta("<yourObjectName>")
print(simplifiedmeta.headers['Last-Modified'])
print(simplifiedmeta.headers['Content-Length'])
print(simplifiedmeta.headers['ETag'])
// Query all object metadata.
objectmeta = bucket.head_object("<yourObjectName>")
print(objectmeta.headers['Content-Type'])
print(objectmeta.headers['Last-Modified'])
print(objectmeta.headers['x-oss-object-type'])
```

4.2.8.5. List objects

Objects in a bucket are sorted alphabetically. OSS SDK for Python provides a series of object listing methods.

Simple list

The following code provides an example on how to list 10 objects in a specified bucket:

```
# -*- coding: utf-8 -*-
import oss2
from itertools import islice
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
for b in islice(oss2.ObjectIterator(bucket), 10):
    print(b.key)
```

List all objects in a bucket

The following code provides an example on how to list all objects in a bucket:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# List all objects in the specified bucket.
for obj in oss2.ObjectIterator(bucket):
    print(obj.key)
```

List objects whose names contain a specified prefix

Example: A bucket contains the following objects: oss.jpg, fun/test.jpg, fun/movie/001.avi, and fun/movie/007.avi. The forward slash (/) is used as the folder delimiter.

The following code provides an example on how to list all objects whose names contain a specified prefix:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# List all objects in the fun directory and its subdirectories.
for obj in oss2.ObjectIterator(bucket, prefix='fun/'):
    print(obj.key)
```

The following figure shows the result.

```
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
```

List objects and subdirectories in a specified directory

OSS uses a flat structure for objects instead of a hierarchical structure to store objects. A directory is an object whose size is 0 and whose name ends with a forward slash (/). You can upload and download this object. By default, objects whose names end with a forward slash (/) are displayed as directories in the OSS console. For the complete sample code that is used to create directories, visit GitHub.

You can specify the delimiter and prefix parameters to list objects by directory.

- If you set prefix to the name of a directory, objects and subdirectories whose names contain the prefix are listed.
- If you set delimiter to a forward slash (/), only the objects and subdirectories in the directory are listed. The objects and directories in subdirectories are not listed.

The following code provides an example on how to list objects and subdirectories in a specified directory:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# List objects and subdirectories in the fun directory. Objects in the subdirectories are not listed.
for obj in oss2.ObjectIterator(bucket, prefix = 'fun/', delimiter = '/'):
    # Use is_prefix to determine whether obj is a directory.
    if obj.is_prefix():  # Determine that obj is a directory.
        print('directory: ' + obj.key)
    else:                # Determine that obj is an object.
        print('file: ' + obj.key)
```

The following figure shows the result.

```
directory: fun/movie/
file: fun/test.jpg
```

List the sizes of objects in a specified directory

The following code provides an example on how to query the sizes of objects in a specified directory:

```
# -*- coding: utf-8 -*-
import oss2
def CalculateFolderLength(bucket, folder):
    length = 0
    for obj in oss2.ObjectIterator(bucket, prefix=folder):
        length += obj.size
    return length
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
for obj in oss2.ObjectIterator(bucket, delimiter='/'):
    if obj.is_prefix():  # Determine that obj is a directory.
        length = CalculateFolderLength(bucket, obj.key)
        print('directory: ' + obj.key + '  length:' + str(length) + "Byte.")
    else: # Determine that obj is an object.
        print('file:' + obj.key + '  length:' + str(obj.size) + "Byte.")
```

4.2.8.6. Query objects

This topic describes how to use SelectObject in OSS SDK for Python to query CSV and JSON objects.

OSS SDK for Python examples

The following code provides an example on how to query CSV and JSON objects:

```
import oss2
def select_call_back(consumed_bytes, total_bytes =  None):
        print('Consumed Bytes:' + str(consumed_bytes) + '\n')
access_key_id = '<yourAccessKeyId>'
access_key_secret = '<yourAccessKeySecret>'
bucket_name = '<yourtBucketName>'
endpoint = '<yourEndpoint>'
# Create an OSS bucket. All object-related methods must be called based on the bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
key =  'python_select.csv'
content =  'Tom Hanks,USA,45\r\n'*1024
filename =  'python_select.csv'
# Upload a CSV object.
bucket.put_object(key, content)
# Set the parameters for the SelectObject API operation.
csv_meta_params = {'RecordDelimiter': '\r\n'}
select_csv_params = {'CsvHeaderInfo': 'None',
                     'RecordDelimiter': '\r\n',
                     'LineRange': (500, 1000)}
csv_header = bucket.create_select_object_meta(key, csv_meta_params)
print(csv_header.rows)
print(csv_header.splits)
result = bucket.select_object(key, "select * from ossobject where _3 > 44", select_call_back, select_csv_params)
select_content = result.read()
print(select_content)
result = bucket.select_object_to_file(key, filename,
      "select * from ossobject where _3 > 44", select_call_back, select_csv_params)
bucket.delete_object(key)
###JSON DOCUMENT
key =  'python_select.json'
content =  "{\"contacts\":[{\"key1\":1,\"key2\":\"hello world1\"},{\"key1\":2,\"key2\":\"hello world2\"}]}"
filename =  'python_select.json'
# Upload a JSON Document object.
bucket.put_object(key, content)
select_json_params = {'Json_Type': 'DOCUMENT'}
result = bucket.select_object(key, "select s.key2 from ossobject.contacts[*] s where s.key1 = 1", None, select_json_params)
select_content = result.read()
print(select_content)
result = bucket.select_object_to_file(key, filename,
      "select s.key2 from ossobject.contacts[*] s where s.key1 = 1", None, select_json_params)
bucket.delete_object(key)
###JSON LINES
key =  'python_select_lines.json'
content =  "{\"key1\":1,\"key2\":\"hello world1\"}\n{\"key1\":2,\"key2\":\"hello world2\"}"
filename =  'python_select.json'
# Upload a JSON Lines object.
bucket.put_object(key, content)
select_json_params = {'Json_Type': 'LINES'}
json_header = bucket.create_select_object_meta(key,select_json_params)
print(json_header.rows)
print(json_header.splits)
result = bucket.select_object(key, "select s.key2 from ossobject s where s.key1 = 1", None, select_json_params)
select_content =  result.read()
print(select_content)
result = bucket.select_object_to_file(key, filename,
          "select s.key2 from ossobject s where s.key1 = 1", None, select_json_params)
bucket.delete_object(key)
```

Python Select API

The following section describes the elements used for the SelectObject API operation in Python in detail. These elements include select_object, select_object_to_file, and create_select_object_meta.

- select_object
  - The following code provides an example on how to set parameters for select_object:

    ```
    def select_object(self, key, sql,
                      progress_callback=None,
                      select_params=None
                      byte_range=None
                      headers=None
                      ):
    ```

    The preceding code example is used to execute an SQL statement on the object that has the specified key and return the query results.

    - sql: the SQL statement that can be directly used as the value of the sql parameter and does not need to be Base64-encoded.
    - Progress_callback: optional. This parameter specifies a callback function used to report the query progress.
    - select_params: the parameters and actions of the SelectObject operation.
    - headers: the header information included in the request. The header information functions the same as that for the GetObject operation. For example, you can configure the bytes field in the request header to specify the range of a CSV object to query.
  - The following table describes the parameters supported by the select_params parameter.

Parameter	Description
Json_Type	■ By default, the object is a CSV object if this parameter is not specified. ■ The object is a JSON Document object if this parameter is set to DOCUMENT. ■ The object is a JSON Lines object if this parameter is set to LINES.
CsvHeaderInfo	The header information of the CSV object. Valid values: *None*, *Ignore*, and *Use*. ■ *None*: This object has no header information configured. ■ *Ignore*: This object has header information configured, which is not used when the SQL statement is executed. ■ *Use*: This object has header information configured. The column names in the header information are used when the SQL statement is executed.
CommentCharacter	The comment character in the CSV object. This parameter value can be only one character in length. The default value is None, which indicates that no comment characters are allowed.
RecordDelimiter	The row delimiter in the CSV object. This parameter value can be only one or two characters in length. Default value: \n.
OutputRecordDelimiter	The row delimiter in the output result of the SELECT statement. Default value: \n.
FieldDelimiter	The column delimiter in the CSV object. This parameter value can be only one character in length. Default value: comma (,).
OutputFieldDelimiter	The column delimiter in the output result of the SELECT statement. Default value: comma (,).
QuoteCharacter	The quote character for the columns in the CSV object. This parameter value can be only one character in length. Default value: double quotation marks ("). Row and column delimiters enclosed in quote characters are processed as normal characters.
SplitRange	The split range in multipart query. The parameter value is a closed interval in the (start, end) format, which indicates the range of splits to query.
LineRange	The row range in multipart query. The parameter value is a closed interval in the (start, end) format, which indicates the range of rows to query.
CompressionType	The format in which the object is compressed. Valid value: GZIP. Default value: None.
KeepAllColumns	If this parameter is set to true, columns that are excluded by the SELECT statement in the CSV object are left empty in the output result. However, the column positions are kept. Default value: false. Example: The columns in the CSV object are firstname, lastname, and age. The SQL statement is `select firstname, age from ossobject`. ■ If KeepAllColumns is set to *true*, the output result is firstname,,age, in which a comma (,) is added to indicate the position of the excluded lastname column. ■ If KeepAllColumns is set to *false*, the output result is firstname,age. **Note** By using this parameter, you can directly use the code that is used to process GetObject to process SelectObject without modifications.
OutputRawData	■ If this parameter is set to *True*, SelectObject directly returns the original data. If data is not returned for a long time, a timeout error may occur. ■ If this parameter is set to *False*, the output data is encapsulated in frames. Default value: False.
EnablePayloadCrc	Specifies whether a cyclic redundancy check (CRC) value is calculated for each frame. Default value: False.
OutputHeader	The header information in the first line of the output result. This parameter applies only to CSV objects.
SkipPartialDataRecord	If this parameter is set to True for a CSV object, the current record is skipped when a column of this record has no data. If this parameter is set to True for a JSON object, the current record is skipped when a key of this record does not exist. If this parameter is set to False, a column without data is left empty in the output result. Example: A row includes the columns firstname, lastname, and age. The SQL statement is `select _1, _4 from ossobject`. ■ If this parameter is set to True, this row is skipped. ■ If this parameter is set to False, firstname,\n is returned.
MaxSkippedRecordsAllowed	The maximum number of skipped rows. The default value is 0, which indicates that an error is returned if a row is skipped.
ParseJsonNumberAsString	■ If this parameter is set to True, all numbers in the JSON object are parsed as strings. ■ If this parameter is set to False, all numbers in the JSON object are parsed as integers or floating-point numbers. Default value: False. High-accuracy floating-point numbers in a JSON object suffer loss of accuracy when they are parsed as floating-point numbers. To keep the accuracy, you can set this parameter to True and use the CAST function to convert the parsed data to the decimal type.

- Returned results of select_object: A SelectObjectResult object is returned. You can use the read() function or the __iter__ method to obtain all results.

  > **Note** If you call the read() function to read all results when there are multiple results to return, excess memory resources are used and you have to wait for a long time. We recommend that you use the __iter__ method (foreach chunk in result) to obtain results and process each chunk in the results. The __iter__ method reduces memory usage and enables the client to process each chunk request processed by the OSS server in a timely manner. This way, the client does not need to wait until all results are returned.

- select_object_to_file

```
def select_object_to_file(self, key, filename, sql,
                  progress_callback=None,
                  select_params=None
                  headers=None
                  ):
```

The preceding code example is used to execute an SQL statement on the object that has the specified key and write the query results to another specified object.

- create_select_object_meta
  - Syntax of create_select_object_meta:

```
def create_select_object_meta(self, key, select_meta_params=None, header=None):
```

The preceding code example is used to create Select Meta for or obtain Select Meta from the object that has the specified key. Select Meta includes the total number of rows, total number of columns (for CSV objects), and total number of splits in an object.

If Select Meta has been created for an object, this function does not re-create Select Meta unless the value of the OverwriteIfExists parameter is set to true.

To create Select Meta for an object, the object must be completely scanned.

  - The following table describes the parameters supported by the select_meta_params parameter.

Parameter	Description
Json_Type	■ By default, the object is a CSV object if this parameter is not specified. ■ If this parameter is specified, the parameter value must be LINES, which indicates that the object is a JSON Lines object. **Note** This operation does not apply to JSON Document objects.
RecordDelimiter	The row delimiter in the CSV object.
FieldDelimiter	The column delimiter in the CSV object.
QuoteCharacter	The quote character in the CSV object. Row and column delimiters enclosed in quote characters are processed as normal characters.
CompressionType	The format in which the object is compressed. No compression formats are supported. The default value is None.
OverwriteIfExists	Specifies whether the created Select Meta overwrites the original Select Meta. You do not need to set this parameter in most cases.

  - Returned results of create_select_object_meta: A GetSelectObjectMetaResult object is returned, which includes the rows and splits attributes. If the object is a CSV object, the select_resp object in the result includes the columns attribute, which indicates the number of columns in the CSV object.

4.2.8.7. Delete objects

This topic describes how to delete objects.

> **Warning** Deleted objects cannot be recovered. Exercise caution when you delete objects.

Delete a single object

The following code provides an example on how to delete a single object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Delete the object. <yourObjectName> indicates the full path of the object that you want to delete from OSS. The path must include the extension of the object name. Example: abc/efg/123.jpg.
# To delete a directory, set <yourObjectName> to the name of the directory you want to delete. If the specified directory is not empty, you must delete all objects in the directory before you can delete the directory.
bucket.delete_object('<yourObjectName>')
```

Delete multiple objects

The following code provides an example on how to delete multiple objects at a time:

```
# Delete three objects at a time. You can delete up to 1,000 objects at a time.
result = bucket.batch_delete_objects(['<yourObjectName-a>', '<yourObjectName-b>', '<yourObjectName-c>'])
# Display the names of the deleted objects.
print('\n'.join(result.deleted_keys))
```

Delete objects whose names contain a specified prefix

The following code provides an example on how to delete objects whose names contain a specified prefix:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
prefix = "<yourKeyPrefix>"
# Delete the objects whose names contain the specified prefix.
for obj in oss2.ObjectIterator(bucket, prefix=prefix):
    bucket.delete_object(obj.key)
```

4.2.8.8. Copy objects

You can copy an object from a source bucket to a destination bucket within the same region.

When you copy a object, take note of the following items:

- You must have read permissions on the source object and write permissions on the destination bucket.
- The source bucket and the destination bucket must be in the same region.

Simple copy

You can use simple copy for objects up to 1 GB in size. The following code provides an example on how to perform simple copy:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourDestinationBucketName>')
bucket.copy_object('<yourSourceBucketName>', '<yourSourceObjectName>', '<yourDestinationObjectName>')
```

Copy a large object

To copy an object whose size is larger than 1 GB, you must split the object into parts and sequentially copy the parts by using UploadPartCopy. To implement multipart copy, perform the following steps:

1. Use bucket.init_multipart_upload to initiate a multipart copy task.
2. Use bucket.upload_part_copy to copy the parts. Each part must be larger than 100 KB in size, except the last part.
3. Use bucket.complete_multipart_upload to complete the multipart copy task.

The following code provides an example on how to perform multipart copy:

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import PartInfo
from oss2 import determine_part_size
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
src_object = '<yourSourceObjectName>'
dst_object = '<yourDestinationObjectName>'
total_size = bucket.head_object(src_object).content_length
part_size = determine_part_size(total_size, preferred_size=100 * 1024)
# Initiate a multipart upload task.
upload_id = bucket.init_multipart_upload(dst_object).upload_id
parts = []
# Copy each part sequentially.
part_number = 1
offset = 0
while offset < total_size:
    num_to_upload = min(part_size, total_size - offset)
    byte_range = (offset, offset + num_to_upload - 1)
    result = bucket.upload_part_copy(bucket.bucket_name, src_object, byte_range,dst_object, upload_id, part_number)
    parts.append(PartInfo(part_number, result.etag))
    offset += num_to_upload
    part_number += 1
# Complete the multipart copy task.
bucket.complete_multipart_upload(dst_object, upload_id, parts)
```

4.2.9. Authorized access

This topic describes how to use Security Token Service (STS) and a signed URL to authorize access.

Use STS to authorize temporary access

Alibaba Cloud STS is a web service that provides a temporary access token to a cloud computing user. You can use STS to grant a third-party application or your RAM user an access credential with a customized validity period and permissions.

STS has the following benefits:

- You need only to generate an access token and send the access token to a third-party application, instead of exposing your long-term AccessKey pair to the third-party application. You can customize the access permissions and validity period of this token.
- The access token automatically expires when the validity period ends.

First, run the following command to install the official Python STS client:

```
pip install aliyun-python-sdk-sts
```

The Python version must be 2.0.6 or later. The following code provides an example on how to use STS to authorize temporary access to upload an object:

```
# -*- coding: utf-8 -*-
from aliyunsdkcore import client
from aliyunsdksts.request.v20150401 import AssumeRoleRequest
import json
import oss2
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
endpoint = 'oss-cn-hangzhou.aliyuncs.com'
access_key_id = '<yourAccessKeyId>'
access_key_secret = '<yourAccessKeySecret>'
bucket_name = '<yourBucketName>'
# The role_arn parameter specifies the resource descriptor of the role.
role_arn = '<yourRoleArn>'
clt = client.AcsClient(access_key_id, access_key_secret, 'cn-hangzhou')
req = AssumeRoleRequest.AssumeRoleRequest()
# Set the format of the returned value to JSON.
req.set_accept_format('json')
req.set_RoleArn(role_arn)
req.set_RoleSessionName('session-name')
body = clt.do_action(req)
# Use the AccessKey pair of the RAM user to apply for a temporary token from STS.
token = json.loads(body)
# Initialize the StsAuth instance based on the authentication information in the temporary token.
auth = oss2.StsAuth(token['Credentials']['AccessKeyId'],
                    token['Credentials']['AccessKeySecret'],
                    token['Credentials']['SecurityToken'])
# Initialize the bucket based on the StsAuth instance.
bucket = oss2.Bucket(auth, endpoint, bucket_name)
# Upload a string.
bucket.put_object('object-name.txt', b'hello world')
```

For the complete sample code of STS usage, visit GitHub.

Download a private object by using a signed URL

You can generate a signed URL for users to access a private bucket.

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('Your AccessKey ID', 'Your AccessKey secret')
bucket = oss2.Bucket(auth, 'Your endpoint', 'Your bucket name')
print(bucket.sign_url('GET', 'object-in-bucket.txt', 60))
```

A signed URL is displayed after you run the preceding code. You can share the URL with other users so that they can download the objects in the bucket by accessing the URL through a browser or by using tools such as wget. The URL is valid only within 60 seconds after it is generated.

Use a signed URL to authorize temporary access

You can generate a signed URL and provide the URL to a visitor for temporary access. When you generate a signed URL, you can specify the validity period of the URL to limit the period of time during which the visitor can access OSS.

The following code provides an example on how to use a signed URL to download an object:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
# Set the validity period of the signed URL to 60 seconds.
print(bucket.sign_url('GET', '<yourObjectName>', 60))
```

4.2.10. Chinese characters and time

This topic describes how to use Chinese characters and time in OSS SDK for Python.

Chinese characters

To use Chinese characters in Python code, you must include a character encoding declaration in the beginning of the code. Otherwise, errors occur when you run the code. Example of a character encoding statement:

```
# -*- coding: utf-8 -*-
```

- Date types

  The following table describes the data types supported by Python 2.x.

Date type	Description
str	The string type. Corresponds to the bytes type in Python 3.x.
unicode	Unicode streams. The length is calculated based on the number of Chinese characters. For example, the length of `u'中文'` is 2.

  The following table describes the data types supported by Python 3.x.

Data type	Description
str	The string type. Corresponds to the unicode type in Python 2.x.
bytes	Byte streams. The length is calculated based on the number of bytes. For example, the length of `b'中文'` is calculated based on the encoding mode. If UTF-8 is used, the length is 6 bytes.

- Input and output limits

  The following table describes the limits on input data.

Input data	Type	Description
OSS object name	str	If the input data type is bytes, it must be encoded in UTF-8.
Local file name	str, unicode	If the input data type is bytes, it must be encoded in UTF-8. Example: the yourLocalFile parameter value used in bucket.get_object_to_file.
Input data stream	bytes	Example: data in bucket.put_object.

  The following table describes limits on output data.

Output data	Type	Description
The result obtained by parsing the XML document	str	Example: a string in the result obtained by using bucket.list_object.
Downloaded content	bytes	By default, data of the bytes type used by OSS SDK for Python must be encoded in UTF-8. Therefore, ensure that the source Python file is encoded in UTF-8.

- Type conversion functions

  OSS SDK for Python provides three functions for type conversion:

Function	Description
to_bytes	- Converts the unicode type to the str type in Python 2.x. Data of other types is returned without changes. - Converts the str type to the bytes type in Python 3.x. Data of other types is returned without changes.
to_unicode	- Converts the str type to the unicode type in Python 2.x. Data of other types is returned without changes. - Converts the bytes type to the str type in Python 3.x. Data of other types is returned without changes.
to_string	This function is equivalent to to_bytes in Python 2.x. This function is equivalent to to_unicode in Python 3.x.

Time

OSS SDK for Python converts timestamp strings of the datetime.datetime type that are obtained from the server to a Unix timestamp corresponding to the amount of time in seconds elapsed since UTC 00:00 on January 1, 1970. For example, the last_modified parameter returned for bucket.get_object is an int Unix timestamp.

You can use the datetime.datetime.fromtimestamp() method to convert time to obtain timestamp strings.

4.2.11. IMG

OSS Image Processing (IMG) is a secure, cost-effective, and highly reliable image processing service that can process large amounts of data. After you upload source images to OSS, you can call `RESTful` API operations to process the images anytime, anywhere, and on any Internet device. IMG provides image processing API operations. To upload images, use the upload API operations provided by OSS. You can use OSS IMG to build your own image processing service.

Basic features

OSS IMG provides the following features:

- Query image information.
- Convert image formats.
- Resize, crop, and rotate images.
- Apply image effect.
- Add image, text, and text-and-image watermarks to images.
- Customize image processing styles. Log on to the OSS console. Click a bucket name to go to the bucket details page. Click the **Image Processing (IMG)** tab and click **Create Rule** to customize an image processing style.
- Call multiple IMG features by using cascade processing.

Usage

You can use standard `HTTP GET` requests to access IMG. All processing parameters are encoded in the query string of a `URL`.

- Anonymous access

  If the ACL of an image object is `public-read`, as described in the following table, anonymous users can access the image object.

Bucket ACL	Object ACL
public-read or public-read-write	default
Any ACL	public-read or public-read-write

  You can access IMG anonymously by using a third-level domain in the following format:

```
http://bucket.<endpoint>/object?x-oss-process=image/action,parame_value
```

  - bucket: the name of the bucket.
  - endpoint: the domain name used to access the region.
  - object: the image object stored in OSS.
  - image: the identifier reserved by IMG.
  - action: the operation performed on the image, such as resizing, cropping, or rotating.
  - parame: the parameter corresponding to the operation performed on the image.

  Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

  If you process an image by using a custom style, you can access IMG anonymously through a third-level domain in the following format:

```
http://bucket.<endpoint>/object?x-oss-process=x-oss-process=style/name
```

  - style: the identifier reserved by the system for the custom style.
  - name: the name of the custom style. The name is specified by the Rule Name parameter when you create the custom style in the console.

  Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
```

  You can use cascade processing to perform multiple operations in sequence on an image by using a URL in the following format:

```
http://bucket.<endpoint>/object?x-oss-process=image/action,parame_value/action,parame_value/...
```

  Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
```

  IMG also supports HTTPS access. The following code provides an example:

```
https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

- Authorized access

  If the ACL of an image object is private, you must have permissions on the image object before you can perform operations on the image object.

Bucket ACL	Object ACL
private	default
Any permission	private

  The following code provides an example on how to generate a signed URL to access IMG:

```
# -*- coding: utf-8 -*-
import oss2
endpoint = '<The OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>'
access_key_id = '<access_key_id>'
access_key_secret = '<access_key_secret>'
bucket_name = '<bucket_name>'
key = 'example.jpg'
# Create a bucket. All object-related API operations can be called based on the bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
# Upload a sample image.
bucket.put_object_from_file(key, 'example.jpg')
# Generate a signed URL and set the validity period to 10 minutes.
style = 'image/resize,m_fixed,w_100,h_100/rotate,90'
url = bucket.sign_url('GET', key, 10 * 60, params={'x-oss-process': style})
print(url)
```

Note

- You can configure Create Style, HTTPS, and Cascading Processing for authorized access.
- The validity period of sign_url is set in seconds.

- Access by using SDKs

  You can use `OSS SDKs` to access and process image objects of any ACLs.

  Note

  - For the complete IMG code, visit GitHub.
  - You can use OSS SDKs to process image objects by configuring Create Style, HTTPS, and Cascading Processing.

  - Basic operations

    The following code provides an example on how to perform basic IMG operations including obtaining image information, converting image formats, resizing, cropping, rotating, watermarking, and applying effects:

```
# -*- coding: utf-8 -*-
import os
import oss2
endpoint = '<The OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>'
access_key_id = '<access_key_id>'
access_key_secret = '<access_key_secret>'
bucket_name = '<bucket_name>'
key = 'example.jpg'
new_pic = 'example-new.jpg'
# Create a bucket. All object-related API operations can be called based on the bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
# Upload a sample image.
bucket.put_object_from_file(key, 'example.jpg')
# Resize the image.
style = 'image/resize,m_fixed,w_100,h_100'
bucket.get_object_to_file(key, new_pic, process=style)
# Crop the image.
style = 'image/crop,w_100,h_100,x_100,y_100,r_1'
bucket.get_object_to_file(key, new_pic, process=style)
# Rotate the image.
style = 'image/rotate,90'
bucket.get_object_to_file(key, new_pic, process=style)
# Sharpen the image.
style = 'image/sharpen,100'
bucket.get_object_to_file(key, new_pic, process=style)
# Add a text watermark to the image.
style = 'image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ'
bucket.get_object_to_file(key, new_pic, process=style)
# Convert the image format.
style = 'image/format,png'
bucket.get_object_to_file(key, new_pic, process=style)
# Delete the sample image.
bucket.delete_object(key)
# Delete the local file.
os.remove(new_pic)
```

    Note `get_object` also supports the IMG feature.

- Custom styles

```
# -*- coding: utf-8 -*-
import os
import oss2
endpoint = '<The OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>'
access_key_id = '<access_key_id>'
access_key_secret = '<access_key_secret>'
bucket_name = '<bucket_name>'
key = 'example.jpg'
new_pic = 'example-new.jpg'
# Create a bucket. All object-related API operations can be called based on the bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
# Upload a sample image.
bucket.put_object_from_file(key, 'example.jpg')
# Customize the image style.
style = 'style/oss-pic-style-w-100'
# Process the image.
bucket.get_object_to_file(key, new_pic, process=style)
# Delete the sample image.
bucket.delete_object(key)
# Delete the local file.
os.remove(new_pic)
```

Note `get_object` also supports the IMG feature.

- Cascading processing

```
# -*- coding: utf-8 -*-
import os
import oss2
endpoint = '<The OSS endpoint. Example: http://oss-cn-hangzhou.aliyuncs.com>'
access_key_id = '<access_key_id>'
access_key_secret = '<access_key_secret>'
bucket_name = '<bucket_name>'
key = 'example.jpg'
new_pic = 'example-new.jpg'
# Create a bucket. All object-related API operations can be called based on the bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
# Upload a sample image.
bucket.put_object_from_file(key, 'example.jpg')
# Perform cascade processing.
style = 'image/resize,m_fixed,w_100,h_100/rotate,90'
# Process the image.
bucket.get_object_to_file(key, new_pic, process=style)
# Delete the sample image.
bucket.delete_object(key)
# Delete the local file.
os.remove(new_pic)
```

Note `get_object` also supports the IMG feature.

4.2.12. Handle exceptions

OssError exceptions of OSS SDK for Python are classified into ClientError, RequestError, and ServerError. Definitions of these errors are provided in the submodule of oss2.exceptions .

The following table describes the variable, type, and description of exceptions.

Variable	Type	Description
status	int	• If a ServerError exception occurs, an HTTP status code is returned. • If a ClientError exception or a RequestError exception occurs, the error status is returned.
request_id	str	• If a ServerError exception occurs, the OSS server returns the request ID. • If a ClientError exception or a RequestError exception occurs, the request ID parameter in the response is empty.
code and message	str	This variable indicates the text in the Code and Message tags in an OSS error response.

Examples

The following code provides an example on how to handle an exception and display the HTTP status code of the error when you download an object that does not exist:

```
# -*- coding: utf-8 -*-
import oss2
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<yourBucketName>')
try:
    stream = bucket.get_object('random-key.txt')
except oss2.exceptions.NoSuchKey as e:
    print('status={0}, request_id={1}'.format(e.status, e.request_id))
```

ClientError

ClientError occurs due to incorrect inputs on the client. For example, the error occurs when you use bucket.batch_delete_objects and receive an empty object list. The status value of ClientError is oss2.exceptions.OSS_CLIENT_ERROR_STATUS.

ServerError

When the OSS server returns the HTTP error code, OSS SDK for Python converts the error code to ServerError. ServerError derives multiple subclasses based on the HTTP status code and OSS error code. The NotFound subclass corresponds to 404 HTTP status code. The Conflict subclass corresponds to 409 HTTP status code.

The following table describes common error codes.

Error	HTTP status code	OSS error code	Description
NotModified	304	None	The error message returned because no modifications are performed.
AccessDenied	403	AccessDenied	The error message returned because the access is denied.
NoSuchBucket	404	NoSuchBucket	The error message returned because the specified bucket does not exist.
NoSuchKey	404	NoSuchKey	The error message returned because the specified object does not exist.
NoSuchUpload	404	NoSuchUpload	The error message returned because the specified multipart upload task does not exist.
NoSuchWebsite	404	NoSuchWebsiteConfiguration	The error message returned because static website hosting is not configured for the specified bucket.
NoSuchLifecycle	404	NoSuchLifecycle	The error message returned because no lifecycle rule is configured for the specified bucket.
NoSuchCors	404	NoSuchCORSConfiguration	The error message returned because CORS rules are not configured for the specified bucket.
BucketNotEmpty	409	BucketNotEmpty	The error message returned because the bucket is not empty.
PositionNotEqualToLength	409	PositionNotEqualToLength	The error message returned because the configured position is not equal to the object length.
ObjectNotAppendable	409	ObjectNotAppendable	The error message returned because the object is not appendable.

4.3. PHP-SDK

4.3.1. Preface

This topic provides the sample code of OSS SDK for PHP in various use cases.

Source code

For the source code of OSS SDK for PHP, visit GitHub.

Sample codes

OSS SDK for PHP provides a variety of sample code for your reference or use. The following table describes the sample codes.

Sample file	Description
Object.php	Shows you how to perform object-related operations, including upload, download, copy, delete, and list objects as well as object metadata-related operations.
MultipartUpload.php	Shows you how to upload large objects and perform multipart upload.
Signature.php	Shows you how to generate a signed URL to authorize access.

Sample file	Description
Callback.php	Shows you how to perform upload callback.
Image.php	Shows you how to use Image Processing (IMG).
LiveChannel.php	Shows you how to use LiveChannel.
Bucket.php	Shows you how to manage buckets, including create, delete, and list buckets as well as configure ACL for a bucket.
BucketLifecycle.php	Shows you how to configure, query, and delete the lifecycle rules for a bucket.
BucketLogging.php	Shows you how to configure, query, and delete the logging configurations for a bucket.
BucketReferer.php	Shows you how to configure, query, and delete the hotlink protection configurations for a bucket.
BucketWebsite.php	Shows you how to configure, query, and delete the static website hosting configurations for a bucket.
BucketCors.php	Shows you how to configure, query, and delete the CORS configurations for a bucket.

4.3.2. Installation

This topic describes how to install OSS SDK for PHP.

Preparation

OSS SDK for PHP is applicable to PHP 5.3 and later versions. This topic uses PHP 5.6.22 as an example.

- Install the environment

  You must install PHP and cURL extension:

  - If a message that indicates the specified module is unavailable is displayed, set extension_dir to `C:/Windows/System32/` .
  - In Ubuntu, run the `sudo apt-get install php-curl` command to install the cURL extension of PHP by using apt-get.
  - In CentOS, run the `sudo yum install php-curl` command to install the cURL extension of PHP by using yum.

- Check the PHP version
  - You can run the `php -v` command to view the current PHP version.
  - You can run the `php -m` command to check whether the cURL extension is installed.

Download OSS SDK for PHP

- Download from GitHub
- Download previous versions

For more information, see API Documentation.

Install OSS SDK for PHP

You can use one of the following method to install SDK:

- Composer
  i. Go to the root directory of your project and run the `composer require aliyuncs/oss-sdk-php` command or add the following dependency to the `composer.json` file:

  ```
  "require": {
      "aliyuncs/oss-sdk-php": "~2.x.x"
  }
  ```

  ii. Run the `composer install` command to install the dependency. After the dependency is installed, check whether your directory structure complies with the following structure:

  ```
  .
  ├── app.php
  ├── composer.json
  ├── composer.lock
  └── vendor
  ```

  In the preceding directory structure, `app.php` indicates your application, and the `vendor/` directory contains the dependent library. You need to add the following dependency to `app.php` :

  ```
  require_once __DIR__ . '/vendor/autoload.php';
  ```

  Note
  - If `autoload.php` is imported to your project, you do not need to import it again after you add the dependency of OSS SDK for PHP.
  - If a network error occurs when you use Composer, you can use the image source of Composer China by running the `composer config -g repositories.packagist composer http://packagist.phpcomposer.com` command.

- PHAR
  i. Select the corresponding version and download the packaged PHAR file from GitHub.

ii. Import the PHAR file in your code:

```
require_once '/path/to/oss-sdk-php.phar';
```

- Source code
  i. Select the required version and download the ZIP package from GitHub.
  ii. The decompressed root directory contains the `autoload.php` file. Import the file to the code:

```
require_once '/path/to/oss-sdk/autoload.php';
```

4.3.3. Initialization

An OSSClient serves as the OSS PHP client to manage OSS resources such as buckets and objects.

Create an OSSClient instance

To create an OSSClient instance, you must specify an endpoint. For more information about endpoints, see Obtain an endpoint.

To create an OSSClient, you must specify an AccessKey pair. For more information about how to obtain an AccessKey pair, see Obtain an AccessKey pair.

- Use an endpoint to create an OSSClient

  The following code provides an example on how to create an OSSClient instance based on an OSS endpoint. OSSClient supports concurrent tasks. Therefore, a project can contain one or more OSSClient instances.

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
} catch (OssException $e) {
    print $e->getMessage();
}
```

- Use STS to create an OSSClient

  The following code provides an example on how to create an OSSClient instance based on Security Token Service (STS):

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$securityToken = "<yourSecurityToken>";
try {
    $ossClient = new OssClient(
        $accessKeyId, $accessKeySecret, $endpoint, false, $securityToken);
} catch (OssException $e) {
    print $e->getMessage();
}
```

- Use a proxy server to create an OSSClient

  You can use a proxy server to create an OSSClient when you use OSS SDK for PHP version 5.3 and later. The following code provides an example on how to use a proxy server to create an OSSClient:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// Specify the IP address of the proxy server. Example: http://<username>:<password>@<proxyip>:<proxyport>.
$requestProxy = "<yourRequestProxy>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false, $requestProxy);
} catch (OssException $e) {
    print $e->getMessage();
}
```

Configure network parameters

The following code provides an example on how to configure network parameters for an OSSClient:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
try {
    $ossClient = new OssClient(
        $accessKeyId, $accessKeySecret, $endpoint);
    // Set the timeout period for data transmission at the socket layer. Unit: seconds. Default value: 5184000.
    $ossClient->setTimeout(3600);
    // Set the timeout period for connection establishment. Unit: seconds. Default value: 10.
    $ossClient->setConnectTimeout(10);
} catch (OssException $e) {
    print $e->getMessage();
}
```

4.3.4. Getting Started

This topic describes how to use Object Storage Service (OSS) SDK for PHP to perform routine operations, such as creating buckets, uploading objects, and downloading objects.

Create a bucket

The following code provides an example on how to create a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket = "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->createBucket($bucket);
} catch (OssException $e) {
    print $e->getMessage();
}
```

For more information about how to obtain an AccessKey ID and AccessKey secret, see Obtain an AccessKey pair.

For more information about how to obtain an endpoint, see Obtain an endpoint.

Upload an object

The following code provides an example on how to upload objects to OSS by using stream upload:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= " <yourBucketName>";
// <yourObjectName> indicates the full path of the object that you want to upload to OSS. <yourObjectName> must include the file extension of the obj
ect. Example: abc/efg/123.jpg.
$object = " <yourObjectName>";
$content = "Hi, OSS.";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putObject($bucket, $object, $content);
} catch (OssException $e) {
    print $e->getMessage();
}
```

Download an object

The following code provides an example on how to download an object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
// <yourObjectName> indicates the full path of the object that you want to download from OSS. <yourObjectName> must include the file extension of the
object. Example: abc/efg/123.jpg.
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $content = $ossClient->getObject($bucket, $object);
    print("object content: " . $content);
} catch (OssException $e) {
    print $e->getMessage();
}
```

List objects

The following code provides an example on how to list objects in a bucket. By default, 100 objects are listed.

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $listObjectInfo = $ossClient->listObjects($bucket);
    $objectList = $listObjectInfo->getObjectList();
    if (!empty($objectList)) {
        foreach ($objectList as $objectInfo) {
        print($objectInfo->getKey() . "\t" . $objectInfo->getSize() . "\t" . $objectInfo->getLastModified() . "\n");
        }
    }
} catch (OssException $e) {
    print $e->getMessage();
}
```

Delete an object

The following code provides an example on how to delete an object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
// <yourObjectName> indicates the full path of the object that you want to delete from OSS. <yourObjectName> must include the file extension of the o
bject. Example: abc/efg/123.jpg.
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteObject($bucket, $object);
} catch (OssException $e) {
    print $e->getMessage();
}
```

4.3.5. Buckets

4.3.5.1. Create buckets

A bucket is a container used to store objects in Object Storage Service (OSS). Each object is contained in a bucket. This topic describes how to create a bucket.

The following code provides an example on how to create a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // Set the storage class of the bucket to Standard. The default storage class is Standard.
    $options = array(
        OssClient::OSS_STORAGE => OssClient::OSS_STORAGE_Standard
    );
    // Set the access control list (ACL) of the bucket to public read. The default bucket ACL is private.
    $ossClient->createBucket($bucket, OssClient::OSS_ACL_TYPE_PUBLIC_READ, $options);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

For the complete code that is used to create a bucket, visit GitHub.

4.3.5.2. List buckets

A bucket is a container that is used to store objects in Object Storage Service (OSS). All objects in OSS are contained in buckets. Bucket names are listed in alphabetical order. You can list buckets that belong to the current Alibaba Cloud account in all regions and meet the specific conditions.

The following sample code provides an example on how to list all buckets that belong to an Apsara Stack tenant account:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $bucketListInfo = $ossClient->listBuckets();
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
$bucketList = $bucketListInfo->getBucketList();
foreach($bucketList as $bucket) {
    print($bucket->getLocation() . "\t" . $bucket->getName() . "\t" . $bucket->getCreatedate() . "\n");
}
```

4.3.5.3. Determine whether a bucket exists

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to determine whether a bucket exists.

The following code provides an example on how to determine whether a bucket exists:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $res = $ossClient->doesBucketExist($bucket);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
if ($res === true) {
    print(__FUNCTION__ . ": OK" . "\n");
} else {
    print(__FUNCTION__ . ": FAILED" . "\n");
}
```

4.3.5.4. Manage the ACL of a bucket

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to configure and query the access control list (ACL) of a bucket.

Configure the ACL for a bucket

The following table describes the ACLs that you can configure for a bucket.

ACL	Description	Value
Private	Only the bucket owner and authorized users have read and write permissions on objects in the bucket. Other users cannot access the objects in the bucket.	OssClient::OSS_ACL_TYPE_PRIVATE
Public read	Only the bucket owner and authorized users have read and write permissions on objects in the bucket. Other users have only read permissions on the objects in the bucket. Exercise caution when you set the ACL of the bucket to this value.	OssClient::OSS_ACL_TYPE_PUBLIC_READ
Public read/write	All users have read and write permissions on objects in the bucket. Exercise caution when you set the ACL of the bucket to this value.	OssClient::OSS_ACL_TYPE_PUBLIC_READ_WRITE

The following code provides an example on how to configure the ACL of a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
// Set the ACL of the bucket to private.
$acl = OssClient::OSS_ACL_TYPE_PRIVATE;
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putBucketAcl($bucket, $acl);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query the ACL of a bucket

The following code provides an example on how to query the ACL of a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $res = $ossClient->getBucketAcl($bucket);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print('acl: ' . $res);
```

4.3.5.5. Delete buckets

If you no longer use a bucket, delete the bucket to stop unnecessary charges.

Before you delete a bucket, ensure that all objects, LiveChannels, and parts that are generated from multipart upload in the bucket are deleted. The following code provides an example on how to delete a bucket:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name.
$bucket= "<yourBucketName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteBucket($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.5.6. Manage lifecycle rules

Object Storage Service (OSS) allows you to configure lifecycle rules to delete expired objects and parts or convert the storage class of expired objects to Infrequent Access (IA) or Archive to reduce your storage costs. This topic describes how to manage lifecycle rules for buckets.

Background

Each lifecycle rule contains the following information:

- Policy: specifies the mode to match objects and parts.
  - Match by prefix: matches objects and parts by prefix. You can create multiple rules to configure different prefixes. Each prefix must be unique.
  - Match by bucket: matches all objects and parts contained in the bucket. You can create only one rule if you select this method.
- Object lifecycle policy: specifies the validity period or expiration data and the operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired objects are deleted.
  - Expiration date: specifies a date. Objects that are last modified before this date expire and are deleted.
- Part lifecycle policy: specifies the validity period or expiration time and the delete operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted.

- Expiration date: specifies a date. Parts that are last modified before this date expire and are deleted.

Lifecycle rules also apply to the parts uploaded by using uploadPart. In this case, the last modified time of an object is the time the multipart upload task is initiated.

Configure lifecycle rules

The following code provides an example on how to configure lifecycle rules for a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\LifecycleConfig;
use OSS\Model\LifecycleRule;
use OSS\Model\LifecycleAction;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
// Specify the rule ID and the prefix of the object names that match the rule.
$ruleId0 = "rule0";
$matchPrefix0 = "A0/";
$ruleId1 = "rule1";
$matchPrefix1 = "A1/";
$lifecycleConfig = new LifecycleConfig();
$actions = array();
// Specify that objects expire three days after they are last modified.
$actions[] = new LifecycleAction(OssClient::OSS_LIFECYCLE_EXPIRATION, OssClient::OSS_LIFECYCLE_TIMING_DAYS, 3);
$lifecycleRule = new LifecycleRule($ruleId0, $matchPrefix0, "Enabled", $actions);
$lifecycleConfig->addRule($lifecycleRule);
$actions = array();
// Specify that objects created before the specified date expire.
$actions[] = new LifecycleAction(OssClient::OSS_LIFECYCLE_EXPIRATION, OssClient::OSS_LIFECYCLE_TIMING_DATE, '2022-10-12T00:00:00.000Z');
$lifecycleRule = new LifecycleRule($ruleId1, $matchPrefix1, "Enabled", $actions);
$lifecycleConfig->addRule($lifecycleRule);
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putBucketLifecycle($bucket, $lifecycleConfig);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query lifecycle rules

The following code provides an example on how to query lifecycle rules of a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$lifecycleConfig = null;
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $lifecycleConfig = $ossClient->getBucketLifecycle($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print($lifecycleConfig->serializeToXml() . "\n");
```

Delete lifecycle rules

The following code provides an example on how to delete lifecycle rules from a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteBucketLifecycle($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.5.7. Hotlink protection

Object Storage Service (OSS) provides hotlink protection to prevent unauthorized domain names from accessing your OSS data. You can configure the Referer field in the HTTP header.

To configure hotlink protection, you must configure the following parameters:

- Referer Whitelist: Only specified domain names are allowed to access OSS resources.
- Allow Empty Referer: If you do not allow empty Refer fields, a request is allowed to access OSS resources only if the request includes the Referer field configured in the HTTP or HTTPS header.

Configure hotlink protection

The following code provides an example on how to configure hotlink protection:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\RefererConfig;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$refererConfig = new RefererConfig();
// Allow empty Referers.
$refererConfig->setAllowEmptyReferer(true);
// Add a Referer allowlist. You can use asterisks (*) and question marks (?) as wildcard characters in Referers.
$refererConfig->addReferer("www.aliiyun.com");
$refererConfig->addReferer("www.aliiyuncs.com");
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putBucketReferer($bucket, $refererConfig);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query the hotlink protection configurations

The following code provides an example on how to query the hotlink protection configurations:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\RefererConfig;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$refererConfig = null;
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $refererConfig = $ossClient->getBucketReferer($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print($refererConfig->serializeToXml() . "\n");
```

Clear the hotlink protection configurations

The following code provides an example on how to clear the hotlink protection configurations:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\RefererConfig;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$refererConfig = new RefererConfig();
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // The hotlink protection configurations of a bucket cannot be directly cleared. You must configure a new hotlink protection rule that allows emp
ty Referer fields to overwrite the existing hotlink protection configurations.
    $ossClient->putBucketReferer($bucket, $refererConfig);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.5.8. Configure logging

You can enable logging to record access to a bucket in log objects, which are stored in a specified bucket.

The log objects are in the following format: `<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString`.

Enable logging

The following code provides an example on how to enable logging for a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$option = array();
// Specify the bucket that is used to store log objects.
$targetBucket = "<yourBucketName>";
$targetPrefix = "access.log";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // Enable logging.
    $ossClient->putBucketLogging($bucket, $targetBucket, $targetPrefix, $option);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

View logging configurations

The following code provides an example on how to view the logging configurations of a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$loggingConfig = null;
$options = array();
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // View the logging configurations.
    $loggingConfig = $ossClient->getBucketLogging($bucket, $options);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print($loggingConfig->serializeToXml() . "\n");
```

Disable logging

The following code provides an example on how to disable logging for a bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // Disable logging.
    $ossClient->deleteBucketLogging($bucket);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.5.9. Configure static website hosting

You can configure static website hosting for your bucket. After the configurations take effect, you can use a bucket domain name as a static website address that can be used to access objects stored in the bucket. For example, index pages and error pages.

For the complete code for static website hosting, visit GitHub.

Configure static website hosting

The following code provides an example on how to configure static website hosting:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\WebsiteConfig;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
// Configure static web hosting. Set the index page to index.html and the error page to error.html.
$websiteConfig = new WebsiteConfig("index.html", "error.html");
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putBucketWebsite($bucket, $websiteConfig);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query static website hosting configurations

The following code provides an example on how to query static website hosting configurations:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$websiteConfig = null;
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $websiteConfig = $ossClient->getBucketWebsite($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print($websiteConfig->serializeToXml() . "\n");
```

Delete static website hosting configurations

The following code provides an example on how to delete static website hosting configurations:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteBucketWebsite($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.5.10. Configure CORS

CORS allows web applications to access resources that belong to different regions. OSS supports CORS API operations for cross-origin access control.

For the complete CORS rule code, visit GitHub.

Configure CORS rules

The following code provides an example on how to configure CORS rules for a specific bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Model\CorsConfig;
use OSS\Model\CorsRule;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$corsConfig = new CorsConfig();
$rule = new CorsRule();
// AllowedHeaders and ExposeHeaders do not support wildcard characters.
$rule->addAllowedHeader("x-oss-header");
// AllowedOlowedMethods allows only one asterisk (*) wildcard character. The asterisk (*) wildcard character specifies that all origins or operations
are allowed.
$rule->addAllowedOrigin("http://www.b.com");
$rule->addAllowedMethod("POST");
$rule->setMaxAgeSeconds(10);
// Up to 10 rules can be configured for each bucket.
$corsConfig->addRule($rule);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // The existing rules are overwritten.
    $ossClient->putBucketCors($bucket, $corsConfig);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query CORS rules

The following code provides an example on how to query CORS rules that are configured for a specific bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$corsConfig = null;
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $corsConfig = $ossClient->getBucketCors($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
print($corsConfig->serializeToXml() . "\n");
```

Delete CORS rules

The following code provides an example on how to delete the CORS rules that are configured for a specific bucket:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteBucketCors($bucket);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.6. Upload objects

4.3.6.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for PHP provides a variety of methods to upload objects.

You can upload objects to OSS by using one of the following methods:

- Simple upload: uploads an object up to 5 GB in size.
- Append upload: uploads an object up to 5 GB in size.
- Multipart upload: uploads an object up to 48.8 TB in size. Use multipart upload for large objects.

4.3.6.2. Simple upload

Simple upload allows you to upload strings or local files. This topic describes how to upload strings and local files.

Upload a string

The following code provides an example on how to upload a string to an OSS object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
// Specify the name of the object.
$object = "<yourObjectName>";
// Configure the object content.
$content = "Hello OSS";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putObject($bucket, $object, $content);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
// You can configure headers in upload requests when you upload the string. For example, you can set the object ACL to private and configure the user
metadata of the object.
$options = array(
    OssClient::OSS_HEADERS => array(
        'x-oss-object-acl' => 'private',
        'x-oss-meta-info' => 'your info'
    ),
);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putObject($bucket, $object, $content, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Upload a local file

The following code provides an example on how to upload a local file to OSS:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the name of the bucket.
$bucket= "<yourBucketName>";
// Specify the name of the object.
$object = "<yourObjectName>";
// <yourLocalFile> consists of the local file path and the file name with an extension. Example: /users/local/myfile.txt.
$filePath = "<yourLocalFile>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->uploadFile($bucket, $object, $filePath);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.6.3. Append upload

You can append strings and local files when you use append upload. The CopyObject operation cannot be performed on append objects.

Append strings

The following code provides an example on how to upload strings to OSS by using append upload:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name.
$bucket= "<yourBucketName>";
// Specify the object name.
$object = "<yourObjectName>";
// Query the content of the object to upload.
$content_array = array('Hello OSS', 'Hi OSS', 'OSS OK');
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // Start the first append. // If the object is appended for the first time, the append position is 0. The return value is the position for the ne
xt append. The position to start the next append is the total length of appended bytes and the append object.
    $position = $ossClient->appendObject($bucket, $object, $content_array[0], 0);
    $position = $ossClient->appendObject($bucket, $object, $content_array[1], $position);
    $position = $ossClient->appendObject($bucket, $object, $content_array[2], $position);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Append local files

The following code provides an example on how to upload local files to OSS by using append upload:

```
 <? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name.
$bucket= "<yourBucketName>";
// Specify the object name.
$object = "<yourObjectName>";
// Obtain Local File 1.
$filePath = "<yourLocalFile1>";
// Obtain Local File 2.
$filePath1 = "<yourLocalFile2>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $position = $ossClient->appendFile($bucket, $object, $filePath, 0);
    $position = $ossClient->appendFile($bucket, $object, $filePath1, $position);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.6.4. Multipart upload

Object Storage Service (OSS) allows you to use multipart upload to split an object into several parts and upload them separately. After all parts are uploaded, you can combine the uploaded parts into a complete object to finish the upload.

To implement multipart upload, perform the following operations:

For the complete code that is used to perform multipart upload, visit GitHub.

The following code provides an example on how to perform multipart upload:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
```

```
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Core\OssUtil;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadFile = "<yourLocalFile>";
/**
 * Step 1. Initiate a multipart upload task and obtain the upload ID.
 */
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    // An upload ID is returned. It is the unique identifier for a multipart upload task. You can initiate related operations such as canceling or qu
erying a multipart upload task based on the upload ID.
    $uploadId = $ossClient->initiateMultipartUpload($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": initiateMultipartUpload FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": initiateMultipartUpload OK" . "\n");
/*
 * Step 2: Upload parts.
 */
$partSize = 10 * 1024 * 1024;
$uploadFileSize = filesize($uploadFile);
$pieces = $ossClient->generateMultiuploadParts($uploadFileSize, $partSize);
$responseUploadPart = array();
$uploadPosition = 0;
$isCheckMd5 = true;
foreach ($pieces as $i => $piece) {
    $fromPos = $uploadPosition + (integer)$piece[$ossClient::OSS_SEEK_TO];
    $toPos = (integer)$piece[$ossClient::OSS_LENGTH] + $fromPos - 1;
    $upOptions = array(
        $ossClient::OSS_FILE_UPLOAD => $uploadFile,
        $ossClient::OSS_PART_NUM => ($i + 1),
        $ossClient::OSS_SEEK_TO => $fromPos,
        $ossClient::OSS_LENGTH => $toPos - $fromPos + 1,
        $ossClient::OSS_CHECK_MD5 => $isCheckMd5,
    );
    // Perform the MD5 verification.
    if ($isCheckMd5) {
        $contentMd5 = OssUtil::getMd5SumForFile($uploadFile, $fromPos, $toPos);
        $upOptions[$ossClient::OSS_CONTENT_MD5] = $contentMd5;
    }
    try {
        // Upload parts.
        $responseUploadPart[] = $ossClient->uploadPart($bucket, $object, $uploadId, $upOptions);
    } catch(OssException $e) {
        printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} OK\n");
}
// $uploadParts is an array that consists of the ETag and PartNumber of each part.
$uploadParts = array();
foreach ($responseUploadPart as $i => $eTag) {
    $uploadParts[] = array(
        'PartNumber' => ($i + 1),
        'ETag' => $eTag,
    );
}
/**
 * Step 3: Complete the multipart upload task.
 */
try {
    // You must provide all valid $uploadParts when you perform this operation. After OSS receives the $uploadParts, OSS verifies all parts one by on
e. After all parts are verified, OSS combines these parts into a complete object.
    $ossClient->completeMultipartUpload($bucket, $object, $uploadId, $uploadParts);
}  catch(OssException $e) {
    printf(__FUNCTION__ . ": completeMultipartUpload FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
printf(__FUNCTION__ . ": completeMultipartUpload OK\n");
```

The following table describes the parameters contained in $options involved in the preceding example.

Parameter	Description
$ossClient::OSS_FILE_UPLOAD	Uploads the object.
OssClient::OSS_PART_NUM	The part number.
OssClient::OSS_SEEK_TO	The start position.
OssClient::OSS_LENGTH	The size of the object.
OssClient::OSS_PART_SIZE	Specifies the part size.
OssClient::OSS_CHECK_MD5	Specifies whether to enable MD5 verification. A value of true indicates that the MD5 verification is enabled.

Cancel a multipart upload task

You can call $ossClient->abortMultipartUpload to cancel a multipart upload task. If you cancel a multipart upload task, you cannot use the upload ID to upload any parts. The uploaded parts are deleted.

The following code provides an example on how to cancel a multipart upload task:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$upload_id = "<yourUploadId>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->abortMultipartUpload($bucket, $object, $upload_id);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Upload a local file by using multipart upload

The following code provides an example on how to upload a local file to OSS by using multipart upload:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$file = "<yourLocalFile>";
$options = array(
    OssClient::OSS_CHECK_MD5 => true,
    OssClient::OSS_PART_SIZE => 1,
);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->multiuploadFile($bucket, $object, $file, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ":  OK" . "\n");
```

Upload a local directory by using multipart upload

The following code provides an example on how to upload a local directory including all files in the directory to OSS by using multipart upload:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$localDirectory = ".";
$prefix = "samples/codes";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->uploadDir($bucket, $prefix, $localDirectory);
}  catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ":  OK" . "\n");
```

List uploaded parts

The following code provides an example on how to list uploaded parts:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadId = "<yourUploadId>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $listPartsInfo = $ossClient->listParts($bucket, $object, $uploadId);
    foreach ($listPartsInfo->getListPart() as $partInfo) {
        print($partInfo->getPartNumber() . "\t" . $partInfo->getSize() . "\t" . $partInfo->getETag() . "\t" . $partInfo->getLastModified() . "\n");
    }
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

List multipart upload tasks

The following code provides an example on how to list multipart upload tasks:

```
<?php
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$options = array(
    'delimiter' => '/',
    'max-uploads' => 100,
    'key-marker' => '',
    'prefix' => '',
    'upload-id-marker' => ''
);
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $listMultipartUploadInfo = $ossClient->listMultipartUploads($bucket, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": listMultipartUploads FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
printf(__FUNCTION__ . ": listMultipartUploads OK\n");
$listUploadInfo = $listMultipartUploadInfo->getUploads();
var_dump($listUploadInfo);
```

The following table describes the parameters contained in $options from the preceding sample code.

Parameter	Description
delimiter	The character used to group object names. Objects whose names contain the same string from the prefix and the next occurrence of the delimiter are grouped as a single result element.
key-marker	Specifies the name of the object after which the listing of multipart upload tasks begins. All multipart upload tasks with objects whose names are alphabetically greater than the keyMarker value are listed. You can use key-marker and upload-id-marker together to specify the position after which the next list begins.
max-uploads	The maximum number of multipart upload tasks to return this time. Default value: 1000. Maximum value: 1000.
prefix	The prefix that the names of returned objects must contain. If you use a prefix for a query, the returned object names contain the prefix.
upload-id-marker	The upload ID of the multipart upload task after which the list begins. This parameter is used together with the key-marker parameter. If key-marker is not configured, upload-id-marker is invalid. If key-marker is configured, the query result includes: • All objects whose names are alphabetically greater than the value of keyMarker. • All objects whose names start with the letter the same as the value of key-marker. The value of uploadId is greater than that of upload-id-marker.

4.3.6.5. Upload callback

This topic describes how to use upload callback in simple upload and multipart upload.

Configure callback for simple upload

The following code provides an example on how to configure callback for simple upload:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
// Configure callback for upload.
// callbackUrl is the IP address of the callback server. Examples: http://oss-demo.aliyuncs.com:23450 and http://127.0.0.1:9090.
// Optional. callbackHost is the value of the host field carried in the callback request.
$url =
    '{
        "callbackUrl":"<yourCallbackServerUrl>",
        "callbackHost":"<yourCallbackServerHost>",
        "callbackBody":"bucket=${bucket}&object=${object}&etag=${etag}&size=${size}&mimeType=${mimeType}&imageInfo.height=${imageInfo.height}&imageIn
fo.width=${imageInfo.width}&imageInfo.format=${imageInfo.format}&my_var1=${x:var1}&my_var2=${x:var2}",
        "callbackBodyType":"application/x-www-form-urlencoded"
    }';
// Configure custom parameters for the callback request. Each custom parameter consists of a key and a value. The key must start with x:.
$var =
    '{
        "x:var1":"value1",
        "x:var2":"value2"
    }';
$options = array(OssClient::OSS_CALLBACK => $url,
    OssClient::OSS_CALLBACK_VAR => $var
);
$result = $ossClient->putObject($bucket, $object, file_get_contents(__FILE__), $options);
print_r($result['body']);
print_r($result['info']['http_code']);
```

Configure callback for multipart upload

The following code provides an example on how to configure callback for multipart upload:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Core\OssUtil;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$uploadFile = "<yourLocalFile>";
/**
 * Step 1. Initiate a multipart upload task and obtain the uploadId.
 */
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
// An upload ID is returned. It is the unique identifier for a multipart upload task. You can initiate related operations such as cancel or query a m
ultipart upload task based on the upload ID.
$uploadId = $ossClient->initiateMultipartUpload($bucket, $object);
print(__FUNCTION__ . ": initiateMultipartUpload OK" . "\n");
/*
 * Step 2. Upload parts.
 */
$partSize = 10 * 1024 * 1024;
$uploadFileSize = filesize($uploadFile);
$pieces = $ossClient->generateMultiuploadParts($uploadFileSize, $partSize);
$responseUploadPart = array();
$uploadPosition = 0;
$isCheckMd5 = true;
foreach ($pieces as $i => $piece) {
    $fromPos = $uploadPosition + (integer)$piece[$ossClient::OSS_SEEK_TO];
    $toPos = (integer)$piece[$ossClient::OSS_LENGTH] + $fromPos - 1;
    $upOptions = array(
        $ossClient::OSS_FILE_UPLOAD => $uploadFile,
        $ossClient::OSS_PART_NUM => ($i + 1),
        $ossClient::OSS_SEEK_TO => $fromPos,
        $ossClient::OSS_LENGTH => $toPos - $fromPos + 1,
        $ossClient::OSS_CHECK_MD5 => $isCheckMd5,
```

```
    );
    // Perform the MD5 verification.
    if ($isCheckMd5) {
        $contentMd5 = OssUtil::getMd5SumForFile($uploadFile, $fromPos, $toPos);
        $upOptions[$ossClient::OSS_CONTENT_MD5] = $contentMd5;
    }
    try {
        // Upload the parts.
        $responseUploadPart[] = $ossClient->uploadPart($bucket, $object, $uploadId, $upOptions);
    } catch(OssException $e) {
        printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    printf(__FUNCTION__ . ": initiateMultipartUpload, uploadPart - part#{$i} OK\n");
}
// $uploadParts is an array that consists of the ETags and the PartNumbers of each part.
$uploadParts = array();
foreach ($responseUploadPart as $i => $eTag) {
    $uploadParts[] = array(
        'PartNumber' => ($i + 1),
        'ETag' => $eTag,
    );
}
/**
 * Step 3. Complete the multipart upload task.
 */
// callbackUrl is the IP address of the callback server. Examples: http://oss-demo.aliyuncs.com:23450 and http://127.0.0.1:9090.
// Optional. callbackHost is the value of the host field carried in the callback request.
$json =
    '{
        "callbackUrl":"<yourCallbackServerUrl>",
        "callbackHost":"<yourCallbackServerHost>",
        "callbackBody":"{\"mimeType\":${mimeType},\"size\":${size},\"x:var1\":${x:var1},\"x:var2\":${x:var2}}",
        "callbackBodyType":"application/json"
    }';
// Configure custom parameters for the callback request. Each custom parameter consists of a key and a value. The key must start with x:.
$var =
    '{
        "x:var1":"value1",
        "x:var2":"value2"
    }';
$options = array(OssClient::OSS_CALLBACK => $json,
    OssClient::OSS_CALLBACK_VAR => $var);
// You must provide all valid $uploadParts when you perform this operation. OSS verifies the validity of each part after it receives the submitted $u
ploadParts. After all parts are verified, OSS combines these parts into a complete object.
$ossClient->completeMultipartUpload($bucket, $object, $uploadId, $uploadParts, $options);
printf(__FUNCTION__ . ": completeMultipartUpload OK\n");
```

4.3.7. Download objects

4.3.7.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for PHP provides a variety of methods to download objects.

You can use one of the following methods to download objects:

- Download to local files
- Download objects to local memory
- Range download
- Conditional download

4.3.7.2. Download to local files

This topic describes how to use OSS SDK for PHP to download specific objects to local files.

The following code provides an example on how to download a specific object to a local file:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
// <yourObjectName> indicates the complete path of the object you want to download from OSS. The path must include the extension of the object name.
Example: abc/efg/123.jpg.
$object = "<yourObjectName>";
// <yourLocalFile> consists of a local file path and a file name with an extension. Example: /users/local/myfile.txt.
$localfile = "<yourLocalFile>";
$options = array(
        OssClient::OSS_FILE_DOWNLOAD => $localfile
    );
// Use try catch to catch exceptions. If an exception is returned, the download fails. If no exceptions are returned, the object is downloaded.
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK, please check localfile: 'upload-test-object-name.txt'" . "\n");
```

4.3.7.3. Download objects to local memory

This topic describes how to use OSS SDK for PHP to download specified objects to local memory.

The following code provides an example on how to download a specified object to local memory and display the content of the object:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
// Specify the object name.
$object = "<yourObjectName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $content = $ossClient->getObject($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print($content);
print(__FUNCTION__ . ": OK" . "\n");
```

Note If a power outage occurs, data stored in the memory is lost. For more information about how to store the downloaded object in a local disk, see Download to local files.

4.3.7.4. Range download

If you need only part of the data in an object, you can use range download to download data within the specified range.

The following code provides an example on how to download data within the range from byte 0 to byte 4 in an object to local memory:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
// Query the data within the range from byte 0 to byte 4, including a total of 5 bytes of data. If the specified range is invalid, for example, the s
pecified range includes a negative number, or the specified value is larger than the object size, the entire object is downloaded.
$options = array(OssClient::OSS_RANGE => '0-4');
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $content = $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print ($content);
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.7.5. Conditional download

OSS allows you to configure one or more conditions for object download. If the specified conditions are met, the object is downloaded. If the specified conditions are not met, an error is returned and the object is not downloaded.

The following table describes the conditions that you can configure for object download.

Parameter	Description	Configuration method
If-Modified-Since	If the specified time is earlier than the time the object is modified, the object is downloaded. Otherwise, 304 Not modified is returned.	OssClient::OSS_IF_MODIFIED_SINCE
If-Unmodified-Since	If the specified time is later than or equal to the time the object is modified, the object is downloaded. Otherwise, 412 Precondition failed is returned.	OssClient::OSS_IF_UNMODIFIED_SINCE
If-Match	If the specified ETag matches that of an object, the object is downloaded. Otherwise, 412 Precondition failed is returned.	OssClient::OSS_IF_MATCH
If-None-Match	If the specified ETag does not match that of an object, the object is downloaded. Otherwise, 304 Not modified is returned.	OssClient::OSS_IF_NONE_MATCH

Note

- You can use If-Modified-Since and If-Unmodified-Since as object download conditions at the same time. You can use If-Match and If-None-Match as object download conditions at the same time.
- You can use $ossClient->getObjectMeta to obtain the ETag.

You can use conditional download to download objects to the local file or local memory. The following code provides an example on how to perform conditional download:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try{
    $options = array(
        OssClient::OSS_HEADERS => array(
            OssClient::OSS_IF_MODIFIED_SINCE => "Fri, 13 Nov 2015 14:47:53 GMT"),
    );
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $content = $ossClient->getObject($bucket, $object, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print ($content);
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.8. Manage objects

4.3.8.1. Overview

You can manage objects in a bucket by using a variety of operations including SetObjectAcl, GetObjectAcl, ListObjects, DeleteObject, and CopyObject.

By using the preceding operations, you can perform the following operations:

- Determine whether objects exist
- Manage object ACLs
- Manage object metadata
- List objects
- Delete objects
- Copy an object

4.3.8.2. Determine whether an object exists

This topic describes how to determine whether an object exists.

The following code provides an example on how to determine whether an object exists:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $exist = $ossClient->doesObjectExist($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($exist);
```

4.3.8.3. Manage the ACL of an object

This topic describes how to manage the access control list (ACL) of an object.

The following table describes the ACLs that you can configure for an object.

ACL	Description	Value
Inherited from the bucket	The ACL of the object is the same as the ACL of the bucket in which the object is stored.	default
Private	Only the object owner and authorized users are granted the read and write permissions on the object.	private
Public read	Only the object owner and authorized users are granted the read and write permissions on the object. Other users are granted only the read permissions on the object. Exercise caution when you set the ACL of the object to this value.	public-read
Public read/write	All users are granted the read and write permissions on the object. Exercise caution when you set the ACL of the object to this value.	public-read-write

Configure the ACL of an object

The following code provides an example on how to configure the ACL of an object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
// Set the ACL of the object to public read. If you do not specify the ACL of the object, the ACL of the object inherits the ACL of the bucket in whi
ch the object is stored.
$acl = "public-read";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putObjectAcl($bucket, $object, $acl);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query the ACL of an object

The following code provides an example on how to query the ACL of an object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $objectAcl = $ossClient->getObjectAcl($bucket, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($objectAcl);
```

4.3.8.4. Manage object metadata

This topic describes how to manage object metadata.

Configure object metadata

The following code provides an example on how to configure object metadata:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$content = file_get_contents(__FILE__);
$options = array(
    OssClient::OSS_HEADERS => array(
        'Expires' => '2012-10-01 08:00:00',
        'Content-Disposition' => 'attachment; filename="xxxxxx"',
        'x-oss-meta-self-define-title' => 'user define meta info',
    ));
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->putObject($bucket, $object, $content, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Modify object metadata

The following code provides an example on how to modify object metadata:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$fromBucket = "<yourFromBucketName>";
$fromObject = "<yourFromObjectName>";
$toBucket = "<yourToBucketName>";
$toObject = "<yourToObjectName>";
$copyOptions = array(
    OssClient::OSS_HEADERS => array(
        'Expires' => '2018-10-01 08:00:00',
        'Content-Disposition' => 'attachment; filename="xxxxxx"',
        'x-oss-meta-location' => 'location',
    ),
);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->copyObject($fromBucket, $fromObject, $toBucket, $toObject, $copyOptions);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Query object metadata

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $objectMeta = $ossClient->getObjectMeta($bucket, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
if (isset($objectMeta[strtolower('Content-Disposition')]) &&
    'attachment; filename="xxxxxx"' === $objectMeta[strtolower('Content-Disposition')]
) {
    print(__FUNCTION__ . ": ObjectMeta checked OK" . "\n");
} else {
    print(__FUNCTION__ . ": ObjectMeta checked FAILED" . "\n");
}
```

4.3.8.5. List objects

This topic describes how to list all objects or objects that meet specific conditions in a specified bucket.

List all objects

The following code provides an example on how to list all the objects in a specified bucket:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
$nextMarker = '';
while (true) {
    try {
        $options = array(
            'delimiter' => '',
            'marker' => $nextMarker,
        );
        $listObjectInfo = $ossClient->listObjects($bucket, $options);
    } catch (OssException $e) {
        printf(__FUNCTION__ . ": FAILED\n");
        printf($e->getMessage() . "\n");
        return;
    }
    // Obtain nextMarker to list the remaining objects. The reading starts from the object next to the last object previously obtained through the li
stObjects operation.
    $nextMarker = $listObjectInfo->getNextMarker();
    $listObject = $listObjectInfo->getObjectList();
    $listPrefix = $listObjectInfo->getPrefixList();
    if (! empty($listObject)) {
        print("objectList:\n");
        foreach ($listObject as $objectInfo) {
            print($objectInfo->getKey() . "\n");
        }
    }
    if (! empty($listPrefix)) {
        print("prefixList: \n");
        foreach ($listPrefix as $prefixInfo) {
            print($prefixInfo->getPrefix() . "\n");
        }
    }
    if ($listObjectInfo->getIsTruncated() ! == "true") {
       break;
    }
}
```

Note

- In the preceding sample code, $options is optional and can be used to list all objects in the entire bucket or in a folder.
- The listObjects method is used to obtain a list of objects. For information about how to obtain attributes such as the name and size of an object, see Manage object metadata.

List the objects that meet specific conditions

The following code provides an example on how to list objects that meet specific conditions:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
$prefix = 'dir/';
$delimiter = '/';
$nextMarker = '';
$maxkeys = 10;
$options = array(
    'delimiter' => $delimiter,
    'prefix' => $prefix,
    'max-keys' => $maxkeys,
    'marker' => $nextMarker,
);
try {
    $listObjectInfo = $ossClient->listObjects($bucket, $options);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
$objectList = $listObjectInfo->getObjectList(); // object list
$prefixList = $listObjectInfo->getPrefixList(); // directory list
if (! empty($objectList)) {
    print("objectList:\n");
    foreach ($objectList as $objectInfo) {
        print($objectInfo->getKey() . "\n");
    }
}
if (! empty($prefixList)) {
    print("prefixList: \n");
    foreach ($prefixList as $prefixInfo) {
        print($prefixInfo->getPrefix() . "\n");
    }
}
```

The following table describes the parameters contained in $options from the preceding sample code.

Parameter	Description	Required
delimiter	Specifies the character used to group objects by name. CommonPrefixes specifies a set of substrings of objects whose names share a specific prefix and end with the next occurrence of the specified delimiter.	No
prefix	Specifies the prefix that the returned object names must contain.	No
max-keys	Specifies the maximum number of objects that can be listed each time. The default value is 100. The maximum value is 1000.	No
marker	Specifies the name of the object after which the list begins.	No

4.3.8.6. Delete objects

This topic describes how to delete objects.

Warning Deleted objects cannot be recovered. Exercise caution when you delete objects.

Delete a single object

The following code provides an example on how to delete a single object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteObject($bucket, $object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Delete multiple objects

The following code provides an example on how to delete multiple objects:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$objects = array();
$objects[] = "<yourObjectName1>";
$objects[] = "<yourObjectName2>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->deleteObjects($bucket, $objects);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Note If an exception is thrown when you use try{}catch{} to catch errors, no objects are deleted. In this case, you can use $e->getMessage to obtain the error message and analyze the causes of the error.

4.3.8.7. Copy objects

This topic describes how to copy an object.

Take note of the following items when you copy an object:

- You must have read permissions on the source object and write permissions on the destination bucket.
- The source bucket and the destination bucket must be in the same region.

Copy a small object

You can call $ossClient->copyObject to copy an object less than or equal to 1 GB in size from the source bucket to the destination bucket within the same region.

The following code provides an example on how to copy a small object:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$from_bucket = "<yourFromBucketName>";
$from_object = "<yourFromObjectName>";
$to_bucket = $bucket;
$to_object = $from_object . '.copy';
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $ossClient->copyObject($from_bucket, $from_object, $to_bucket, $to_object);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Copy a large object

To copy an object whose size is greater than 1 GB, you must split the object into parts and sequentially copy the parts by using UploadPartCopy. To implement multipart copy, perform the following steps:

1. Call $ossClient->initiateMultipartUpload to initiate a multipart copy task.
2. Call $ossClient->uploadPartCopy to perform multipart copy. All parts must be larger than 100 KB, except for the very last part.
3. Call $ossClient->completeMultipartUpload to complete the multipart copy task.

The following code provides an example on how to use multipart copy to copy large objects:

```
<?php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeySecret = "<yourAccessKeySecret>";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$src_bucket = "<yourSourceBucketName>";
$src_object = "<yourSourceObjectName>";
$dst_bucket = "<yourDestinationBucketName>";
$dst_object = "<yourDestinationObjectName>";
// Set the part size.
$part_size = 256*1024*1024;
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
    $objectMeta = $ossClient->getObjectMeta($src_bucket, $src_object);
    $length = $objectMeta['content-length'] + 0;
    // Initiate a multipart copy task.
    $upload_id = $ossClient->initiateMultipartUpload($dst_bucket, $dst_object);
    // Copy each part sequentially.
    $pieces = $ossClient->generateMultiuploadParts($length, $part_size);
    $response_upload_part = array();
    $copyId = 1;
    $upload_position = 0;
    foreach ($pieces as $i => $piece) {
        $from_pos = $upload_position + (integer)$piece['seekTo'];
        $to_pos = (integer)$piece['length'] + $from_pos - 1;
        $up_options = array(
            'start' => $from_pos,
            'end' => $to_pos,
        );
        $response_upload_part[] = $ossClient->uploadPartCopy( $src_bucket, $src_object, $dst_bucket, $dst_object, $copyId, $upload_id, $up_options);
        $copyId = $copyId + 1;
    }
    // Complete the multipart copy task.
    $upload_parts = array();
    foreach ($response_upload_part as $i => $etag) {
        $upload_parts[] = array(
            'PartNumber' => ($i + 1),
            'ETag' => $etag,
        );
    }
    $result = $ossClient->completeMultipartUpload($dst_bucket, $dst_object, $upload_id, $upload_parts);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.9. Authorized access

This topic describes how to use Security Token Service (STS) and a signed URL to authorize access.

Use STS to authorize temporary access

Alibaba Cloud STS is a web service that provides a temporary access token to a cloud computing user. You can use STS to grant a third-party application or your RAM user an access credential with a customized validity period and permissions.

STS has the following benefits:

- You need only to generate an access token and send the access token to a third-party application, instead of exposing your long-term AccessKey pair to the third-party application. You can customize the access permissions and validity period of this token.
- The access token automatically expires when the validity period ends.

The following code provides an example on how to create a signature request by using STS:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$securityToken = "<yourSecurityToken>";
$content = "Hi, OSS.";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false, $securityToken);
    $ossClient->putObject($bucket, $object, $content);
} catch (OssException $e) {
    print $e->getMessage();
}
```

Use a signed URL to authorize temporary access

You can generate a signed URL and provide the URL to a visitor for temporary access. When you generate a signed URL, you can specify the validity period of the URL to limit the period of time during which the visitor can access OSS.

For the complete code of authorized access, visit GitHub.

- Generate a signed URL to download an object

  The following code provides an example on how to generate a signed URL to download an object:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Http\RequestCore;
use OSS\Http\ResponseCore;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$securityToken = "<yourSecurityToken>";
// Set the validity period of a URL to 3,600 seconds.
$timeout = 3600;
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false, $securityToken);
    // Generate the signed URL for GetObject.
    $signedUrl = $ossClient->signUrl($bucket, $object, $timeout);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": signedUrl: " . $signedUrl . "\n");
// You can run code to access the signed URL, or access the signed URL in a browser.
$request = new RequestCore($signedUrl);
// Set GET as the default method to access the singed URL.
$request->set_method('GET');
$request->add_header('Content-Type', '');
$request->send_request();
$res = new ResponseCore($request->get_response_header(), $request->get_response_body(), $request->get_response_code());
if ($res->isOK()) {
    print(__FUNCTION__ . ": OK" . "\n");
} else {
    print(__FUNCTION__ . ": FAILED" . "\n");
};
```

- Generate a signed URL to upload an object

  The following code provides an example on how to generate a signed URL to upload an object:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
use OSS\Core\OssException;
use OSS\Http\RequestCore;
use OSS\Http\ResponseCore;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$securityToken = "<yourSecurityToken>";
$timeout = 3600;
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false, $securityToken);
    // Generate the signed URL for PutObject.
    $signedUrl = $ossClient->signUrl($bucket, $object, $timeout, "PUT");
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": signedUrl: " . $signedUrl . "\n");
$content = "Hello OSS.";
$request = new RequestCore($signedUrl);
// Set PUT as the default method to access the signed URL.
$request->set_method('PUT');
$request->add_header('Content-Type', '');
$request->add_header('Content-Length', strlen($content));
$request->set_body($content);
$request->send_request();
$res = new ResponseCore($request->get_response_header(),
    $request->get_response_body(), $request->get_response_code());
if ($res->isOK()) {
    print(__FUNCTION__ . ": OK" . "\n");
} else {
    print(__FUNCTION__ . ": FAILED" . "\n");
};
```

4.3.10. IMG

OSS Image Processing (IMG) is a secure, cost-effective, and highly reliable image processing service that can process large amounts of data. After you upload source images to OSS, you can call RESTful API operations to process the images anytime, anywhere, and on any Internet device.

For the complete IMG code, visit GitHub.

Basic features

OSS IMG provides the following features:

- Query image information.
- Convert image formats.
- Resize, crop, and rotate images.
- Apply image effect.
- Add image, text, and text-and-image watermarks to images.
- Customize image processing styles. Log on to the OSS console. Click a bucket name to go to the bucket details page. Click the **Image Processing (IMG)** tab and click **Create Rule** to customize an image processing style.
- Use cascade processing to call multiple IMG features.

Usage

You can use standard `HTTP GET` requests to access IMG. All processing parameters are encoded in the query string of a URL.

If the ACL of an image object is public-read, as described in the following table, anonymous users can access the image object.

Bucket ACL	Object ACL
public-read or public-read-write	default
Any ACL	public-read or public-read-write

You can anonymously access IMG by using a third-level domain in the following format:

```
http://bucket.<endpoint>/object?x-oss-process=image/action,parame_value
```

Parameter	Description
bucket	The name of the bucket.
endpoint	The domain name used to access the region.
object	The name of the image object.
image	The identifier reserved by IMG.
action	The operations performed on images, such as scaling, cropping, and rotating.
param	The parameters corresponding to the operations performed on the images.

- Basic operations

  For example, you can scale an image to a width of 100 and adjust the height based on the ratio:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- Customize an image style

  You can anonymously access IMG by using a third-level domain in the following format:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  - style: the identifier reserved by IMG for the custom style.
  - yourStyleName: the name of the custom style. The name is specified by the Rule Name parameter when you create the custom style in the console.

  Example:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- Cascade processing

  You can use cascade processing to perform multiple operations in sequence on an image by using a URL in the following format:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  Example:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- Access by using HTTPS

  IMG supports access by using HTTPS. Example:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- Authorized access

  If the ACL of an image object is private, you must have permissions on the image object before you can perform operations on the image object.

Bucket ACL	Object ACL
private	default
Any ACL	private

  The following code provides an example on how to generate a signed URL to access IMG:

  ```
  <? php
  if (is_file(__DIR__ . '/../autoload.php')) {
      require_once __DIR__ . '/../autoload.php';
  }
  if (is_file(__DIR__ . '/../vendor/autoload.php')) {
      require_once __DIR__ . '/../vendor/autoload.php';
  }
  use OSS\OssClient;
  $accessKeyId = "<yourAccessKeyId>";
  $accessKeySecret = "<yourAccessKeySecret>";
  // The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
  $endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
  $bucket= "<yourBucketName>";
  $object = "<yourObjectName>";
  $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
  // Generate a signed URL that you can directly access by using a browser. The validity period of the URL is 3,600 seconds.
  $timeout = 3600;
  $options = array(
      OssClient::OSS_PROCESS => "image/resize,m_lfit,h_100,w_100" );
  $signedUrl = $ossClient->signUrl($bucket, $object, $timeout, "GET", $options);
  print("rtmp url: \n" . $signedUrl);
  ```

- Access by using SDKs

  You can use OSS SDKs to access and process the image objects of any ACL.

- Basic operations

  The following code provides an example on how to perform basic IMG operations including obtaining image information, converting image formats, resizing, cropping, rotating, watermarking, and applying effects:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");
// Resize the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/resize,m_fixed,h_100,w_100" );
$ossClient->getObject($bucket, $object, $options);
// Crop the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/crop,w_100,h_100,x_100,y_100,r_1" );
$ossClient->getObject($bucket, $object, $options);
// Rotate the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/rotate,90" );
$ossClient->getObject($bucket, $object, $options);
// Sharpen the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/sharpen,100" );
$ossClient->getObject($bucket, $object, $options);
// Add watermarks to the image.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ" );
$ossClient->getObject($bucket, $object, $options);
// Convert the image format.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/format,png" );
$ossClient->getObject($bucket, $object, $options);
// Obtain the image information.
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => "image/info" );
$ossClient->getObject($bucket, $object, $options);
// Delete the sample image.
$ossClient->deleteObject($bucket, $object);
```

- Custom image styles

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");
// Customize the image style.
$style = "style/oss-pic-style-w-300";
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => $style);
$ossClient->getObject($bucket, $object, $options);
// Delete the sample image.
$ossClient->deleteObject($bucket, $object);
```

- Cascade processing

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket = "<yourBucketName>";
$object = "<yourObjectName>";
$download_file = "download.jpg";
$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint, false);
// Upload a sample image.
$ossClient->uploadFile($bucket, $object, "example.jpg");
// Perform cascade processing.
$style = "image/resize,m_fixed,w_100,h_100/rotate,90";
$options = array(
    OssClient::OSS_FILE_DOWNLOAD => $download_file,
    OssClient::OSS_PROCESS => $style);
$ossClient->getObject($bucket, $object, $options);
// Delete the sample image.
$ossClient->deleteObject($bucket, $object);
```

Persistent storage of processed images

The following code provides an example on how to save processed images:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    require_once __DIR__ . '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}
use OSS\OssClient;
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// The endpoint of the China (Hangzhou) region is used in this example. Specify the actual endpoint.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
// Specify the object name.
$object = "<yourObjectName>";
// Specify the object name for the processed image.
$targetObject = "<yourTargetObjectName>";
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint);
// The following code provides an example on how to convert the image format of a processed image. To crop, rotate, or watermark an image, modify the
$options parameters.
$options = array(
  OssClient::OSS_PROCESS => "image/format,png"
);
$content = $ossClient->getObject($bucket, $object, $options);
$ossClient->putObject($bucket, $targetObject, $content);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print($content);
print(__FUNCTION__ . ": OK" . "\n");
```

4.3.11. Error handling

OSS SDK for PHP exceptions (OssException) include errors caused by invalid parameters and objects that do not exist. You can use getMessage to obtain an error message.

For more information about OssException, visit GitHub.

Error handling example

The following code provides an example on how to rectify an error and display the error information when you create a bucket that already exists:

```
    try {
        $ossClient->createBucket($bucket);
    } catch (OssException $e) {
        print("Exception:" . $e->getMessage() . "\n");
    }
```

The following table describes the error messages you can obtain.

Parameter	Description
HTTPStatus	The HTTP status code that can be obtained by using getHTTPStatus.
ErrorCode	The error code returned by OSS. The error code can be obtained by using getErrorCode.
ErrorMessage	The error information returned by OSS. The error information can be obtained by using getErrorMessage.
RequestId	The UUID used to uniquely identify the request. If the problem persists, you can provide the request ID of a problem to the OSS development engineers for help. You can obtain the UUID by using getRequestId.
Details	The error information details returned by OSS. You can obtain the error message details by using getDetails.

OSS error codes

Error code	Error message	HTTP status code
AccessDenied	The error message returned because the access is denied.	403
BucketAlreadyExists	The error message returned when the bucket already exists.	409
BucketNotEmpty	The error message returned when the bucket is not empty.	409
EntityTooLarge	The error message returned because the entity exceeds the maximum allowed size.	400

Error code	Error message	HTTP status code
EntityTooSmall	The error message returned because the entity is smaller than the minimum allowed size.	400
FileGroupTooLarge	The error message returned because the object group exceeds the maximum allowed size.	400
FilePartNotExist	The error message returned because the specified part does not exist.	400
FilePartStale	The error message returned because the part has expired.	400
InvalidArgument	The error message returned because the parameter format is invalid.	400
InvalidAccessKeyId	The error message returned because the specified AccessKey ID does not exist.	403
InvalidBucketName	The error message returned because the bucket name is invalid.	400
InvalidDigest	The error message returned because the digest is invalid.	400
InvalidObjectName	The error message returned because the object name is invalid.	400
InvalidPart	The error message returned because the part is invalid.	400
InvalidPartOrder	The error message returned because the part order is invalid.	400
InvalidTargetBucketForLogging	The error message returned because the target bucket specified for logging is invalid.	400
InternalError	The error message returned because an internal OSS error occurs.	500
MalformedXML	The error message returned because the XML format is invalid.	400
MethodNotAllowed	The error message returned because the method is not supported.	405
MissingArgument	The error message returned because a parameter is missing.	411
MissingContentLength	The error message returned because the content length is not specified.	411
NoSuchBucket	The error message returned because the specified bucket does not exist.	404
NoSuchKey	The error message returned because the specified object does not exist.	404
NoSuchUpload	The error message returned because the specified multipart upload ID does not exist.	404
NotImplemented	The error message returned because the method cannot be implemented.	501
PreconditionFailed	The error message returned because a precondition error occurs.	412
RequestTimeTooSkewed	The error message returned because the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes.	403
RequestTimeout	The error message returned because the request times out.	400
SignatureDoesNotMatch	The error message returned because a signature error occurs.	403
InvalidEncryptionAlgorithmError	The error message returned because the specified encryption algorithm based on entropy encoding is invalid.	400

4.4. C-SDK

4.4.1. Preface

This topic describes the information about OSS SDK for C.

Compatibility

- For OSS SDK for C V3.*.*: *. compatible.
- For OSS SDK for C V2. *. *.*:
  - Compatible with Windows
  - Compatible with Linux API operations and incompatible with the following linked list (aos_list_t) traversal API operations:
    - aos_list_for_each_entry
    - aos_list_for_each_entry_reverse
    - aos_list_for_each_entry_safe
    - aos_list_for_each_entry_safe_reverse
- For OSS SDK for C V1.*.*: compatible except the following structures and API operations:
  - oss_request_options_t
  - oss_get_object_to_buffer
  - oss_get_object_to_file
  - oss_get_object_to_buffer_by_url
  - oss_get_object_to_file_by_url
  - oss_init_multipart_upload
  - oss_complete_multipart_upload
- For OSS SDK for C V0.0.*: incompatible.

SDK source code

For the source code of OSS SDK for C, visit GitHub.

Sample codes

The OSS SDK for C provides a variety of sample codes for your reference or use. The following table describes the sample codes provided by OSS SDK for C.

Document	Description
oss_put_object_sample	Shows you how to upload objects, including upload objects from the memory, upload objects by using MD5 verification, upload objects with object metadata, and upload objects based on a signed URL.
oss_get_object_sample.c	Shows you how to download objects, including download objects to the memory or local files, range download, and object download based on a signed URL.
oss_append_object_sample.c	Shows you how to perform append upload, including append upload from the memory and append upload from local files.
oss_multipart_upload_sample.c	Shows you how to perform multipart upload, including multipart upload from the memory, multipart upload from local files, and multipart upload cancellation.
oss_resumable_sample.c	Shows you how to perform resumable upload.
oss_head_object_sample.c	Shows you how to query object metadata.
oss_delete_object_sample.c	Shows you how to delete objects, including the deletion of a single object and multiple objects.
oss_callback_sample.c	Shows you how to perform upload callback, including object upload callback and multipart upload callback.
oss_progress_sample.c	Shows you how to use a progress bar during object upload, append upload, multipart upload, and download.
oss_crc_sample.c	Shows you how to perform CRC in object upload and download.
oss_image_sample.c	Shows you how to use Image Processing (IMG).

4.4.2. Installation

This topic describes how to install OSS SDK for C.

Install OSS SDK for C on Linux

- Install CMake and third-party libraries

  When you install OSS SDK for C on Linux, you must install CMake and the following third-party libraries: cRUL, APR, APR-util, and Mini-XML.

  The following table describes parameters required to install CMake and the third-party libraries.

Parameter	Description	Version
CMake	Manages the build process of software by using a compiler-independent method.	2.6.0 or later
curl	Resolves network issues.	7.32.0 or later
apr-util	Resolves the issues of memory and different platforms.	1.5.2 or later
minixml	Parses the XML data returned by a request.	2.9

  Install CMake and the third-party libraries based on the following systems:

- Ubuntu and Debian
  - Install CMake

    ```
    sudo apt-get install cmake
    ```

  - Install third-party libraries

    ```
    sudo apt-get install libcurl4-openssl-dev libapr1-dev libaprutil1-dev libmxml-dev
    ```

- RedHat, Aliyun, and CentOS
  - Install CMake

    ```
    sudo yum install cmake
    ```

  - Install third-party libraries

    ```
    sudo yum install curl-devel apr-devel apr-util-devel  mxml mxml-devel
    ```

    If the YUM source does not contain the MXML installation package, use RPM to install MXML. Download the RPM package based on your operating system:

    - 64-bit: MXML-2.9-1.x86_64.rpm
    - 32-bit: MXML-2.9-1.i386.rpm

    Use the RPM package to install MXML:

    ```
    rpm -ivh mxml-2.9-1.x86_64.rpm
    ```

- SuSE
  - Install CMake

    ```
    zypper install cmake
    ```

  - Install third-party libraries

    ```
    zypper install libcurl-devel libapr1-devel libapr-util1-devel mxml-devel
    ```

- Other Linux-based systems
  - Install CMake

    Download CMake, and run the following commands to install CMake:

    ```
    ./configure
    make
    make install
    ```

    > **Note** When you run ./configure, the default installation path is /usr/local/. To specify the installation path, add the -prefix option when you run ./configure.

  - Install libcurl

    Download libcurl, and run the following commands to install libcurl:

    ```
    ./configure
    make
    make install
    ```

  - Install APR

    Download APR, and run the following commands to install APR:

    ```
    ./configure
    make
    make install
    ```

  - Install APR-util

    Download APR-util, and run the following commands to install APR-util:

    ```
    // You must specify the --with-APR option when you install APR-util.
    ./configure --with-apr=/your/apr/install/path
    make
    make install
    ```

  - Install Mini-XML

    > **Note** Use Mini-XML 2.x for installation.

    Download Mini-XML, and run the following commands to install Mini-XML:

    ```
    ./configure
    make
    sudo make install
    ```

- Download the SDK
  - Download directly
  - Download from GitHub
  - Download previous versions

- Install the SDK
  - If third-party libraries cRUL, APR, APR-util, and MXML are installed in the default path, run the following commands to install OSS SDK for C:

```
cmake .
make
make install
```

  - You can specify a path to which third-party libraries cRUL, APR, APR-util, and MXML are installed. Run the following commands to install OSS SDK for C:

```
cmake -f CMakeLists.txt
// Set the build type to Release. Typical build types include Debug, Release, RelWithDebInfo, and MinSizeRel. By default, Debug is used.
-DCMAKE_BUILD_TYPE=Release
// Set the directory to install the SDK.
-DCMAKE_INSTALL_PREFIX=/usr/local/
// Specify the path where third-party libraries cRUL, APR, APR-util, and XML are installed.
-DCURL_INCLUDE_DIR=/usr/include/curl
-DCURL_LIBRARY=/usr/lib64/libcurl.so
-DAPR_INCLUDE_DIR=/usr/include/apr-1
-DAPR_LIBRARY=/usr/lib64/libapr-1.so
-DAPR_UTIL_INCLUDE_DIR=/usr/include/apr-1
-DAPR_UTIL_LIBRARY=/usr/lib64/libaprutil-1.so
-DMINIXML_INCLUDE_DIR=/usr/include
-DMINIXML_LIBRARY=/usr/lib64/libmxml.so
// If the 'Could not find APR-config/APR-1-config' error is reported during compilation, one possible reason is that the APR-1-config file does n
ot exist in the default path. Add the following options:
-DAPR_CONFIG_BIN=/path/to/bin/apr-1-config
// If the 'Could not find apu-config/apu-1-config' error is reported, add the following options:
-DAPU_CONFIG_BIN=/path/to/bin/apu-1-config
```

Install OSS SDK for C on Windows

- Download the SDK
  - Download directly
  - Download from GitHub
  - Download previous versions
- Install the SDK

  For more information about how to use Visual Studio to compile OSS SDK for C, visit Use Alibaba Cloud OSS SDK for C on Windows.

  > **Note** If you use Visual Studio 2012 or later, you are prompted to choose whether to upgrade the project version to adapt to the latest versions of the compiler and libraries. We recommend that you use versions that match your project. If the project uses the compiler and libraries of the latest versions, upgrade the project. Otherwise, you do not need to upgrade the project.

4.4.3. Initialization

You must initialize oss_request_options_t and specify an endpoint to use OSS SDK for C.

You must specify an endpoint when you initialize OSS SDK for C. For more information about endpoints, see Obtain an endpoint.

You must specify an AccessKey pair when you initialize OSS SDK for C. For more information about how to obtain an AccessKey pair, see Obtain an AccessKey pair.

Use the OSS domain name to initialize request options

The following code provides an example on how to use the OSS domain name to initialize request options:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters. The second parameter in this function indicates the ownership of ctl. By default, the value is 0. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* The code for logic operations is omitted. */
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note For more information about how to set the request options, see How to set parameters for cRUL-based communication in OSS SDK for C.

Use the custom domain name to initialize request options

The following code provides an example on how to use custom domain name to initialize request options:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourCustomEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options) {
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Enable the CNAME and bind the custom domain name to the bucket. */
    options->config->is_cname = 1;
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* The code for logic operations is omitted. */
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Use STS to initialize request options

The following code provides an example on how to use STS to initialize request options:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *sts_token = "<yourSecurityToken>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Configure STS. */
    aos_str_set(&options->config->sts_token, sts_token);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main() {
    aos_pool_t *p;
    oss_request_options_t *options;
    /* Initialize global variables, which are run before the program starts. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        return -1;
    }
    /* Initialize the memory pool and options. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    /* The code for logic operations is omitted. */
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Configure a timeout period

The following code provides an example on how to configure a timeout period:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters. The second parameter in this function indicates the ownership of ctl. By default, the value is 0. */
    options->ctl = aos_http_controller_create(options->pool, 0);
    /* Set the connection timeout period. Default value: 10 seconds. */
    options->ctl->options->connect_timeout = 10;
    /* Set the DNS cache timeout period. Default value: 60 seconds. */
    options->ctl->options->dns_cache_timeout = 60;
    /*
    Configure the request timeout period:
    Set the value of speed_limit to specify the minimum acceptable speed. Default value: 1 KB/s.
    Set the value of speed_time to specify the minimum acceptable speed. Default value: 15 seconds.
    The default values indicate that if the transmission speed is continuously smaller than 1 KB/s for 15 consecutive seconds, the request times out.
    */
    options->ctl->options->speed_limit = 1024;
    options->ctl->options->speed_time = 15;
}
```

4.4.4. Getting Started

This topic describes how to use Object Storage Service (OSS) SDK for C to complete routine operations such as bucket creation, object upload, and object download.

Sample projects

- Sample projects for Linux
  i. Download a sample project and extract it.
  ii. When you install oss-c-sdk-demo-specified-installation, you must specify the installation directory: **/home/your/oss/csdk**. You do not need to specify installation directories for other sample projects because other sample projects are automatically installed based on OSS SDK for C and other dependent third-party libraries.

iii. Replace OSS_ENDPOINT, ACCESS_KEY_ID, ACCESS_KEY_SECRET, and BUCKET_NAME in the sample project with valid values. If OSS SDK for C and the dynamic libraries of the dependent libraries are not in the system directory, use `LD_LIBRARY_PATH` to specify the directory of OSS SDK for C and the dynamic libraries of the dependent libraries.

iv. Go to the oss-c-sdk-demo-xxx directory. Run `make` to compile the sample project. Run `./main` to run the executable program. To re-compile the project, run `make clean`.

For more information about the sample projects, see Use Alibaba Cloud OSS SDK for C in Linux.

- Sample projects for Windows

i. Download a sample project and extract it.

For more sample projects, visit GitHub.

ii. Use Visual Studio to open the sample project, and then set the following parameters in the oss_config.c file to valid values: OSS_ENDPOINT, ACCESS_KEY_ID, ACCESS_KEY_SECRET, BUCKET_NAME, OBJECT_NAME, MULTIPART_UPLOAD_FILE_PATH, and DIR_NAME. Compile and run the project. You can develop your own project based on the sample project.

> **Note** For more information about how to use Visual Studio, see Use Alibaba Cloud OSS SDK for C in Windows.

Create a bucket

The following code provides an example on how to create a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Create the bucket. */
    aos_status_t *resp_status;
    aos_table_t *resp_headers;
    oss_acl_e oss_acl = OSS_ACL_PRIVATE;
    aos_string_t bucket;
    /* Assign the char* data to the bucket. */
    aos_str_set(&bucket, bucket_name);
    resp_status = oss_create_bucket(oss_client_options, &bucket, oss_acl, &resp_headers);
    /* Determine whether the bucket is created. */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Upload an object

The following code provides an example on how to upload objects to OSS by using stream upload:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_status_t *resp_status;
    aos_table_t *headers;
    aos_table_t *resp_headers;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    const char *data = "More than just cloud.";
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers = aos_table_make(pool, 0);
    /* Convert char* data to aos_list_t data. */
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, data, strlen(data));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload the object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the object is uploaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("put file succeeded\n");
    } else {
        printf("put file failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Download an object

The following code provides an example on how to download a specified object to a local computer:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *download_filename = "<yourDownloadedFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_status_t *resp_status;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, download_filename);
    headers = aos_table_make(pool, 0);
    /* Download the object. */
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    /* Determine whether the object is downloaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("get object to local file succeeded\n");
    } else {
        printf("get object to local file failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List objects

The following code provides an example on how to list objects in a specified bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool = NULL;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    /* List objects. */
    resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
    /* Determine whether the objects are listed. */
    if (!aos_status_is_ok(resp_status))
    {
        printf("list object failed\n");
    }
    else{
        /* Display the objects. */
        printf("Object\tSize\tLastModified\n");
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        printf("Total %d\n", size);
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete an object

The following code provides an example on how to delete an object:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool = NULL;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_status_t *resp_status = NULL;
    aos_table_t *resp_headers = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Delete the object. */
    resp_status = oss_delete_object(oss_client_options, &bucket, &object, &resp_headers);
    /* Determine whether the object is deleted. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeeded\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5. Buckets

4.4.5.1. Create buckets

A bucket is a container used to store objects in Object Storage Service (OSS). Each object is contained in a bucket. This topic describes how to create a bucket.

The following code provides an example on how to create a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. The value 0 indicates that CNAME is not used.  */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory.  */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    oss_acl_e oss_acl = OSS_ACL_PRIVATE;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign char* data to a bucket of the aos_string_t type.  */
    aos_str_set(&bucket, bucket_name);
    /* Create the bucket. */
    resp_status = oss_create_bucket(oss_client_options, &bucket, oss_acl, &resp_headers);
    /* Determine whether the bucket is created.  */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources.  */
    aos_http_io_deinitialize();
    return 0;
}
```

The following code provides an example on how to specify the access control list (ACL) of a bucket when you create the bucket:

```
    /* Create a bucket and set the ACL of the bucket to public read. The default bucket ACL is private. */
    resp_status = oss_create_bucket(oss_client_options, &bucket, OSS_ACL_PUBLIC_READ, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
```

4.4.5.2. List buckets

A bucket is a container that is used to store objects in Object Storage Service (OSS). All objects in OSS are contained in buckets. Bucket names are listed in alphabetical order. You can list buckets that belong to the current Alibaba Cloud account in all regions and meet the specific conditions.

The following code provides an example on how to list buckets that belong to the current Alibaba Cloud account in all regions:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t *pool is equivalent to apr_pool_t. The code to create a memory pool is included in the apr l
ibrary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, acces_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_list_buckets_params_t *params = NULL;
    oss_list_bucket_content_t *content = NULL;
    int size = 0;
    params = oss_create_list_buckets_params(pool);
    /* List the buckets. */
    resp_status = oss_list_bucket(oss_client_options, params, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("list buckets succeeded\n");
    } else {
        printf("list buckets failed\n");
    }
    /* Display the list of buckets. */
    aos_list_for_each_entry(oss_list_bucket_content_t, content, &params->bucket_list, node) {
        printf("BucketName: %s\n", content->name.data);
        ++size;
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.3. Manage bucket ACLs

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to configure and query the access control list (ACL) of a bucket.

Configure ACL for a bucket

The following table describes the bucket ACLs.

ACL	Description	Value
Private	Only the owner or authorized users of the bucket can read and write the object.	OSS_ACL_PRIVATE
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	OSS_ACL_PUBLIC_READ
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	OSS_ACL_PUBLIC_READ_WRITE

The following code provides an example on how to configure the ACL of a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign char* data to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Set the ACL of the bucket to public read (OSS_ACL_PUBLIC_READ).
    resp_status = oss_put_bucket_acl(oss_client_options, &bucket, OSS_ACL_PUBLIC_READ, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("set bucket acl succeeded\n");
    } else {
        printf("set bucket acl failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Query the ACL of a bucket

The following code provides an example on how to query the ACL of a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information, such as endpoint, access_key_id, acces_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t oss_acl;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign char* data to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Query the ACL of the bucket. */
    resp_status = oss_get_bucket_acl(oss_client_options, &bucket, &oss_acl, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket acl succeeded : %s \n", oss_acl.data);
    } else {
        printf("get bucket acl failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.4. Delete buckets

If you no longer use a bucket, delete the bucket to stop unnecessary charges.

Before you delete a bucket, ensure that all objects, LiveChannels, and parts that are generated from multipart upload in the bucket are deleted. The following code provides an example on how to delete a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t *pool is equivalent to apr_pool_t. The code to create a memory pool is included in the apr l
ibrary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, acces_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign the char* data to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Delete the bucket. */
    resp_status = oss_delete_bucket (oss_client_options, &bucket, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("delete bucket succeeded\n");
    } else {
        printf("delete bucket failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.5. Manage lifecycle rules

Object Storage Service (OSS) allows you to configure lifecycle rules to delete expired objects and parts or convert the storage class of expired objects to Infrequent Access (IA) or Archive to reduce your storage costs. This topic describes how to manage lifecycle rules for buckets.

Background

Each lifecycle rule contains the following information:

- Policy: specifies the mode to match objects and parts.
  - Match by prefix: matches objects and parts by prefix. You can create multiple rules to configure different prefixes. Each prefix must be unique.
  - Match by bucket: matches all objects and parts contained in the bucket. You can create only one rule if you select this method.
- Object lifecycle policy: specifies the validity period or expiration data and the operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired objects are deleted.
  - Expiration date: specifies a date. Objects that are last modified before this date expire and are deleted.
- Part lifecycle policy: specifies the validity period or expiration time and the delete operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted.
  - Expiration date: specifies a date. Parts that are last modified before this date expire and are deleted.

Lifecycle rules also apply to the parts uploaded by using uploadPart. In this case, the last modified time of an object is the time the multipart upload task is initiated.

Configure lifecycle rules

The following code provides an example on how to configure lifecycle rules for a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_list_t lifecycle_rule_list;
    /* Create lifecycle rules for the bucket. */
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&lifecycle_rule_list);
    /* Specify the validity period. */
    oss_lifecycle_rule_content_t *rule_content_days = oss_create_lifecycle_rule_content(pool);
    aos_str_set(&rule_content_days->id, "rule-1");
    aos_str_set(&rule_content_days->prefix, "obsoleted");
    aos_str_set(&rule_content_days->status, "Enabled");
    rule_content_days->days = 3;
    aos_list_add_tail(&rule_content_days->node, &lifecycle_rule_list);
    /* Specify the expiration date. */
    oss_lifecycle_rule_content_t *rule_content_date = oss_create_lifecycle_rule_content(pool);
    aos_str_set(&rule_content_date->id, "rule-2");
    aos_str_set(&rule_content_date->prefix, "delete");
    aos_str_set(&rule_content_date->status, "Enabled");
    aos_str_set(&rule_content_date->date, "2022-10-11T00:00:00.000Z");
    aos_list_add_tail(&rule_content_date->node, &lifecycle_rule_list);
    /* Configure the lifecycle rules. */
    resp_status = oss_put_bucket_lifecycle(oss_client_options, &bucket, &lifecycle_rule_list, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put bucket lifecycle succeeded\n");
    } else {
        printf("put bucket lifecycle failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Query lifecycle rules

The following code provides an example on how to query lifecycle rules of a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_list_t lifecycle_rule_list;
    oss_lifecycle_rule_content_t *rule_content;
    char *rule_id;
    char *prefix;
    char *status;
    int days = INT_MAX;
    char* date = "";
    aos_str_set(&bucket, bucket_name);
    /* Query and display the lifecycle rules configured for the bucket. */
    aos_list_init(&lifecycle_rule_list);
    resp_status = oss_get_bucket_lifecycle(oss_client_options, &bucket, &lifecycle_rule_list, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket lifecycle succeeded\n");
        aos_list_for_each_entry(oss_lifecycle_rule_content_t, rule_content, &lifecycle_rule_list, node) {
            rule_id = apr_psprintf(pool, "%.*s", rule_content->id.len, rule_content->id.data);
            prefix = apr_psprintf(pool, "%.*s", rule_content->prefix.len, rule_content->prefix.data);
            status = apr_psprintf(pool,"%.*s", rule_content->status.len, rule_content->status.data);
            date = apr_psprintf(pool, "%.*s", rule_content->date.len, rule_content->date.data);
            days = rule_content->days;
        }
        printf("rule_id: %s \n", rule_id);
        printf("prefix: %s \n", prefix);
        printf("status: %s \n", status);
        printf("date: %s \n", date);
        printf("days: %d \n", days);
    } else {
        printf("get bucket lifecycle failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete lifecycle rules

The following code provides an example on how to delete lifecycle rules from a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete the lifecycle rules configured for the bucket. */
    aos_str_set(&bucket, bucket_name);
    resp_status = oss_delete_bucket_lifecycle(oss_client_options, &bucket, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("delete bucket lifecycle succeeded\n");
    } else {
        printf("delete bucket lifecycle failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.6. CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to different regions. OSS provides CORS operations to facilitate cross-origin access control.

Configure CORS rules

Run the following code to configure CORS rules for a specified bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_list_t cors_rule_list;
    oss_cors_rule_t *cors_rule1 = NULL, *cors_rule2 = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&cors_rule_list);
    cors_rule1 = oss_create_cors_rule(pool);
    aos_list_add_tail(&cors_rule1->node, &cors_rule_list);
    oss_create_sub_cors_rule(pool, &cors_rule1->allowed_origin_list, "allowed_origin_1_1");
    oss_create_sub_cors_rule(pool, &cors_rule1->allowed_origin_list, "allowed_origin_1_1");
    oss_create_sub_cors_rule(pool, &cors_rule1->allowed_method_list, "PUT");
    oss_create_sub_cors_rule(pool, &cors_rule1->allowed_method_list, "GET");
    oss_create_sub_cors_rule(pool, &cors_rule1->allowed_head_list, "Authorization");
    oss_create_sub_cors_rule(pool, &cors_rule1->expose_head_list, "expose_head_1_1");
    oss_create_sub_cors_rule(pool, &cors_rule1->expose_head_list, "expose_head_1_1");
    cors_rule2 = oss_create_cors_rule(pool);
    aos_list_add_tail(&cors_rule2->node, &cors_rule_list);
    oss_create_sub_cors_rule(pool, &cors_rule2->allowed_origin_list, "allowed_origin_2_1");
    oss_create_sub_cors_rule(pool, &cors_rule2->allowed_origin_list, "allowed_origin_2_2");
    oss_create_sub_cors_rule(pool, &cors_rule2->allowed_method_list, "PUT");
    oss_create_sub_cors_rule(pool, &cors_rule2->allowed_method_list, "GET");
    oss_create_sub_cors_rule(pool, &cors_rule2->allowed_head_list, "Authorization");
    oss_create_sub_cors_rule(pool, &cors_rule2->expose_head_list, "expose_head_2_1");
    oss_create_sub_cors_rule(pool, &cors_rule2->expose_head_list, "expose_head_2_2");
    /* Configure the CORS rules. */
    resp_status = oss_put_bucket_cors(oss_client_options, &bucket, &cors_rule_list, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put bucket cors succeeded\n");
    } else {
        printf("put bucket cors failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Query CORS rules

Run the following code to obtain CORS rules:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_list_t cors_rule_list;
    oss_cors_rule_t *cors_rule = NULL;
    oss_sub_cors_rule_t *sub_cors_rule = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Query the CORS rules. */
    aos_list_init(&cors_rule_list);
    resp_status = oss_get_bucket_cors(oss_client_options, &bucket, &cors_rule_list, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket cors succeeded\n");
        aos_list_for_each_entry(oss_cors_rule_t, cors_rule, &cors_rule_list, node) {
            printf("max_age_seconds: %d\n", cors_rule->max_age_seconds);
            aos_list_for_each_entry(oss_sub_cors_rule_t, sub_cors_rule, &cors_rule->allowed_origin_list, node) {
                printf("allowed_origin_list: %s \n", sub_cors_rule->rule.data);
            }
            aos_list_for_each_entry(oss_sub_cors_rule_t, sub_cors_rule, &cors_rule->allowed_method_list, node) {
                printf("allowed_method_list: %s \n", sub_cors_rule->rule.data);
            }
            aos_list_for_each_entry(oss_sub_cors_rule_t, sub_cors_rule, &cors_rule->allowed_head_list, node) {
                printf("allowed_head_list: %s \n", sub_cors_rule->rule.data);
            }
            aos_list_for_each_entry(oss_sub_cors_rule_t, sub_cors_rule, &cors_rule->expose_head_list, node) {
                printf("expose_head_list: %s \n", sub_cors_rule->rule.data);
            }
        }
    } else {
        printf("get bucket cors failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete CORS rules

Run the following code to delete all CORS rules for the specified bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete the CORS rules. */
    resp_status = oss_delete_bucket_cors(oss_client_options, &bucket, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket cors succeeded\n");
    } else {
        printf("get bucket cors failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.7. Configure logging

You can enable logging to record access to a bucket in log objects, which are stored in a specified bucket.

The name of log objects is in the following format: `<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString`.

Enable logging

Run the following code to enable bucket access logging:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *target_bucket_name = "<yourTargetBucketName>";
const char *target_logging_prefix = "<yourTargetPrefix>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    oss_logging_config_content_t *content;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    content = oss_create_logging_rule_content(pool);
    aos_str_set(&content->target_bucket, target_bucket_name);
    aos_str_set(&content->prefix, target_logging_prefix);
    /* Enable logging for the bucket. */
    resp_status = oss_put_bucket_logging(oss_client_options, &bucket, content, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put symlink succeeded\n");
    } else {
        printf("put symlink failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

View logging configurations

Run the following code to view the access logging configurations for a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    oss_logging_config_content_t *content;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    content = oss_create_logging_rule_content(pool);
    /* View logging configurations of the bucket. */
    resp_status = oss_get_bucket_logging(oss_client_options, &bucket, content, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket logging succeeded\n");
    } else {
        printf("get bucket logging failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    printf("bucket:%s, prefix:%s\n", content->target_bucket.data, content->prefix.data);
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Disable logging

Run the following code to disable access logging for a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Disable logging. */
    resp_status = oss_delete_bucket_logging(oss_client_options, &bucket, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("delete bucket logging succeeded\n");
    } else {
        printf("delete bucket logging failed, code:%d, error_code:%s, error_msg:%s, request_id:%s\n",
            resp_status->code, resp_status->error_code, resp_status->error_msg, resp_status->req_id);
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.8. Configure hotlink protection

OSS provides hotlink protection to prevent unauthorized domain names from accessing your OSS data. You can configure the Referer field in the HTTP header.

To configure hotlink protection, you must configure the following parameters:

- Referer Whitelist: Only specified domain names are allowed to access OSS resources.
- Allow Empty Referer: If you do not allow empty Refer fields, a request is allowed to access OSS resources only if the request includes the Referer field configured in the HTTP or HTTPS header.

Configure hotlink protection

Run the following code to configure the referer whitelist:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_referer_config_t referer_config;
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&referer_config.referer_list);
    oss_create_and_add_refer(pool, &referer_config, "http://www.aliyun.com");
    oss_create_and_add_refer(pool, &referer_config, "https://www.aliyun.com");
    referer_config.allow_empty_referer = 0;
    /* Add a Referer whitelist. You can use an asterisk (*) or a question mark (?) as a wildcard character to set the Referer parameter. */
    resp_status = oss_put_bucket_referer(oss_client_options, &bucket, &referer_config, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put bucket referer succeeded\n");
    } else {
        printf("put bucket referer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Query hotlink protection information

Run the following code to obtain a referer whiltelist:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_referer_config_t referer_config;
    oss_referer_t *referer;
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&referer_config.referer_list);
    /* Query the Referer whitelist of the bucket. */
    resp_status = oss_get_bucket_referer(oss_client_options, &bucket, &referer_config, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket referer succeeded\n");
        aos_list_for_each_entry(oss_referer_t, referer, &referer_config.referer_list, node) {
            printf("get referer %s\n", referer->referer.data);
        }
    } else {
        printf("get bucket referer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete hotlink protection configurations

Run the following code to clear a referer whitelist:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_referer_config_t referer_config;
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&referer_config.referer_list);
    referer_config.allow_empty_referer = 1;
    /* You cannot clear hotlink protection configurations directly. To delete hotlink protection configurations, you must create a rule that allows a
n empty Referer field to replace the existing rule. */
    resp_status = oss_put_bucket_referer(oss_client_options, &bucket, &referer_config, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("delete bucket referer succeeded\n");
    } else {
        printf("delete bucket referer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.5.9. Configure static website hosting

You can set the static website hosting mode for your bucket. After configurations take effect, you can use the bucket domain name to access this static website and be redirected to a specified index page or error page.

Configure static website hosting

Run the following code to configure static website hosting:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_website_config_t website_config;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&website_config.suffix_str, "index.html");
    aos_str_set(&website_config.key_str, "error.html");
    /* Configure static web hosting: The index page is index.html and the error page is error.html. */
    resp_status = oss_put_bucket_website(oss_client_options, &bucket, &website_config, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put bucket website succeeded\n");
    } else {
        printf("put bucket website failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

View static website hosting configurations

Run the following code to view static website hosting configurations:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_website_config_t website_config;
    aos_str_set(&bucket, bucket_name);
    /* View the static website hosting configurations. */
    resp_status = oss_get_bucket_website(oss_client_options, &bucket, &website_config, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket website succeeded\n");
        printf("website_config: %s  %s \n", website_config.suffix_str.data, website_config.key_str.data);
    } else {
        printf("get bucket website failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete static website hosting configurations

Run the following code to delete static website hosting configurations:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete the static website hosting configurations. */
    resp_status = oss_delete_bucket_website(oss_client_options, &bucket, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("delete bucket website succeeded\n");
    } else {
        printf("delete bucket website failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.6. Upload objects

4.4.6.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for C provides a variety of methods to upload objects.

You can upload an object to OSS by using one of the following methods:

- Simple upload: includes stream upload and object upload. You can use this method to upload an object up to 5 GB in size.
- Append upload: uploads an object up to 5 GB in size.
- Resumable upload: supports concurrent upload and resumable upload. You can define the size of each part. This method is suitable for the upload of large objects. You can use this method to upload an object up to 48.8 TB in size.
- Multipart upload: uploads an object up to 48.8 TB in size. This method is suitable for the upload of large objects.

During object upload, you can configure object metadata and view upload progress by using a progress bar. For more information, see Use progress bars. After the object is uploaded, you can perform upload callbacks. For more information, see Upload callback.

4.4.6.2. Simple upload

This topic describes how to use simple upload.

For the complete code that is used to perform simple upload, visit GitHub.

Upload data from memory

The following code provides an example on how to upload data from memory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload the object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the object is uploaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Upload a local file

The following code provides an example on how to upload a local file to OSS:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Upload the object. */
    resp_status = oss_put_object_from_file(oss_client_options, &bucket, &object, &file, headers, &resp_headers);
    /* Determine whether the object is uploaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

FAQ: How can I upload objects whose names contain Chinese characters?

On Windows systems, you can use UTF-8 to encode the object names that contain Chinese characters and upload the object. The following code provides an example on how to upload objects whose names contain Chinese characters:

```
#include "oss_api.h"
#include "aos_http_io.h"
#include <wchar.h>
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename_ch = "c:\\测试.txt";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
```

```
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
char* multibyte_to_utf8(const char * ch, char *str, int size)
{
    if (! ch || ! str || size <= 0)
        return NULL;
    int chlen = strlen(ch);
    int multi_byte_cnt = 0;
    for (int i = 0; i < chlen - 1; i++) {
        if ((ch[i] & 0x80) && (ch[i + 1] & 0x80)) {
            i++;
            multi_byte_cnt++;
        }
    }
    if (multi_byte_cnt == 0)
        return ch;
    int len = MultiByteToWideChar(CP_ACP, 0, ch, -1, NULL, 0);
    if (len <= 0)
        return NULL;
    wchar_t *wch = malloc(sizeof(wchar_t) * (len + 1));
    if (! wch)
        return NULL;
    wmemset(wch, 0, len + 1);
    MultiByteToWideChar(CP_ACP, 0, ch, -1, wch, len);
    len = WideCharToMultiByte(CP_UTF8, 0, wch, -1, NULL, 0, NULL, NULL);
    if ((len <= 0) || (size < len + 1)) {
        free(wch);
        return NULL;
    }
    memset(str, 0, len + 1);
    WideCharToMultiByte(CP_UTF8, 0, wch, -1, str, len, NULL, NULL);
    free(wch);
    return str;
}
int main(int argc, char argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR lib
rary. */
    aos_pool_t pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Use UTF-8 to encode the object names that contain multibyte Chinese characters */
    char local_filename_buf[1024];
    char * local_filename = multibyte_to_utf8(local_filename_ch, local_filename_buf, 1024);
    aos_str_set(&file, local_filename);
    /* Upload the object. */
    resp_status = oss_put_object_from_file(oss_client_options, &bucket, &object, &file, headers, &resp_headers);
    /* Determine whether the object is uploaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.6.3. Append upload

You can append strings and files during append upload. You cannot perform CopyObject for append objects.

Note You cannot perform CopyObject for append objects.

Append data from memory

The following code provides an example on how to append data from memory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    int64_t position = 0;
    char *next_append_position = NULL;
    aos_buf_t *content = NULL;
    aos_table_t *headers1 = NULL;
    aos_table_t *headers2 = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers1 = aos_table_make(pool, 0);
    /* Obtain the position where the object is appended for the first time. */
    resp_status = oss_head_object(oss_client_options, &bucket, &object, headers1, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        next_append_position = (char*)(apr_table_get(resp_headers, "x-oss-next-append-position"));
        position = atoi(next_append_position);
    }
    /* Append the object. */
    headers2 = aos_table_make(pool, 0);
    aos_list_init(&buffer);
    content = aos_buf_pack(pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    resp_status = oss_append_object_from_buffer(oss_client_options, &bucket, &object, position, &buffer, headers2, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("append object from buffer succeeded\n");
    } else {
        printf("append object from buffer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Append data from a local file

The following code provides an example on how to append data from a local file:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    int64_t position = 0;
    char *next_append_position = NULL;
    aos_table_t *headers1 = NULL;
    aos_table_t *headers2 = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Obtain the position where the object is appended for the first time. */
    resp_status = oss_head_object(oss_client_options, &bucket, &object, headers1, &resp_headers);
    if(aos_status_is_ok(resp_status)) {
        next_append_position = (char*)(apr_table_get(resp_headers, "x-oss-next-append-position"));
        position = atoi(next_append_position);
    }
    /* Append the object. */
    resp_status = oss_append_object_from_file(oss_client_options, &bucket, &object, position, &file, headers2, &resp_headers);
    /* Determine whether the append upload is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("append object from file succeeded\n");
    } else {
        printf("append object from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.6.4. Resumable upload

If the system stops responding during multipart upload, you can resume the upload by using the ListMultipartUploads and ListParts operations to retrieve all multipart upload tasks on an object and list the uploaded parts in each task. This method allows an upload task to be resumed from the last uploaded part.

The progress of the current upload is recorded in the checkpoint file during the upload process. If the upload of a part fails during the process, the next upload attempt starts from the recorded position in the checkpoint file. The checkpoint file is deleted after the resumable upload task is completed.

You can use `oss_resumable_clt_params_t` to perform resumable upload. The following table describes common parameters used in this method:

Parameter	Description
thread_num	The number of concurrent threads. Default value: 1.

Parameter	Description
enable_checkpoint	Specifies whether to enable resumable upload. This feature is disabled by default.
partSize	The size of each part. The part size ranges from 100 KB to 5 GB.
checkpoint_path	The path of the checkpoint file. Default value: {upload_file_path}.cp.

The following code provides an example on how to perform resumable upload:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool
. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_list_t resp_body;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_resumable_clt_params_t *clt_params;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    aos_list_init(&resp_body);
    /* Start resumable upload. */
    clt_params = oss_create_resumable_clt_params_content(pool, 1024 * 100, 3, AOS_TRUE, NULL);
    resp_status = oss_resumable_upload_file(oss_client_options, &bucket, &object, &file, headers, NULL, clt_params, NULL, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("resumable upload succeeded\n");
    } else {
        printf("resumable upload failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.6.5. Multipart upload

OSS allows you to use multipart upload to split an object into several parts and upload them separately. After all parts are uploaded, you can combine the uploaded parts into a complete object to complete the upload.

To implement multipart upload, perform the following operations:

The following code provides an example on how to perform multipart upload:

```
#include "oss_api.h"
#include "aos_http_io.h"
#include <sys/stat.h>
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int64_t get_file_size(const char *file_path)
{
    int64_t filesize = -1;
    struct stat statbuff;
    if(stat(file_path, &statbuff) < 0){
        return filesize;
    } else {
        filesize = statbuff.st_size;
    }
    return filesize;
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    oss_upload_file_t *upload_file = NULL;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *complete_headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_null(&upload_id);
    headers = aos_table_make(p, 1);
    complete_headers = aos_table_make(p, 1);
    int part_num = 1;
    /* Initiate a multipart upload task and obtain an upload ID. */
    resp_status = oss_init_multipart_upload(oss_client_options, &bucket, &object, &upload_id, headers, &resp_headers);
    /* Determine whether the multipart upload task is initiated. */
    if (aos_status_is_ok(resp_status)) {
        printf("Init multipart upload succeeded, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    } else {
        printf("Init multipart upload failed, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    }
    /* Upload the parts. */
    int64_t file_length = 0;
    int64_t pos = 0;
    file_length = get_file_size(local_filename);
    while(pos < file_length) {
        upload_file = oss_create_upload_file(pool);
        aos_str_set(&upload_file->filename, local_filename);
        upload_file->file_pos = pos;
        pos += 100 * 1024;
        upload_file->file_last = pos < file_length ? pos : file_length;
        resp_status = oss_upload_part_from_file(oss_client_options, &bucket, &object, &upload_id, part_num++, upload_file, &resp_headers);
        if (aos_status_is_ok(resp_status)) {
```

```
        if (aos_status_is_ok(resp_status)) {
            printf("Multipart upload part from file succeeded\n");
        } else {
            printf("Multipart upload part from file failed\n");
        }
    }
    oss_list_upload_part_params_t *params = NULL;
    aos_list_t complete_part_list;
    /* Query the uploaded parts. */
    params = oss_create_list_upload_part_params(pool);
    params->max_ret = 1000;
    aos_list_init(&complete_part_list);
    resp_status = oss_list_upload_part(oss_client_options, &bucket, &object, &upload_id, params, &resp_headers);
    /* Determine whether all parts are listed. */
    if (aos_status_is_ok(resp_status)) {
        printf("List multipart succeeded\n");
    } else {
        printf("List multipart failed\n");
    }
    oss_complete_part_content_t *complete_part_content = NULL;
    oss_list_part_content_t *part_content = NULL;
    aos_list_for_each_entry(oss_list_part_content_t, part_content, &params->part_list, node) {
        complete_part_content = oss_create_complete_part_content(pool);
        aos_str_set(&complete_part_content->part_number, part_content->part_number.data);
        aos_str_set(&complete_part_content->etag, part_content->etag.data);
        aos_list_add_tail(&complete_part_content->node, &complete_part_list);
    }
    /* Complete the multipart upload task. */
    resp_status = oss_complete_multipart_upload(oss_client_options, &bucket, &object, &upload_id,
            &complete_part_list, complete_headers, &resp_headers);
    /* Determine whether the multipart upload task is complete. */
    if (aos_status_is_ok(resp_status)) {
        printf("Complete multipart upload from file succeeded, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    } else {
        printf("Complete multipart upload from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note Query the uploaded parts. You must provide the ETag value of each part to call the oss_complete_multipart_upload operation. You can use one of the following method to obtain the ETag values: First, the response of a part upload contains the ETag value of the part. You can save the values for future use. Second, you can call the oss_list_upload_part operation to query the ETag values of the uploaded parts. In this example, the second method is used.

Cancel a multipart upload task

The following code provides an example on how to cancel a multipart upload task:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_null(&upload_id);
    /* Initiate a multipart upload task and obtain an upload ID. */
    resp_status = oss_init_multipart_upload(oss_client_options, &bucket, &object, &upload_id, headers, &resp_headers);
    /* Determine whether the multipart upload task is initiated. */
    if (aos_status_is_ok(resp_status)) {
        printf("Init multipart upload succeeded, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    } else {
        printf("Init multipart upload failed, upload_id:%.*s\n",
               upload_id.len, upload_id.data);
    }
    /* Cancel the multipart upload task. */
    resp_status = oss_abort_multipart_upload(oss_client_options, &bucket, &object, &upload_id, &resp_headers);
    /* Determine whether the multipart upload task is canceled. */
    if (aos_status_is_ok(resp_status)) {
        printf("Abort multipart upload succeeded, upload_id::%.*s\n",
               upload_id.len, upload_id.data);
    } else {
        printf("Abort multipart upload failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note If a multipart upload task is canceled, the upload ID can no longer be used to upload any parts. The parts uploaded by using that upload ID are also deleted.

4.4.6.6. Use progress bars

Progress bars are used to indicate the progress of an upload or download.

For the complete code of upload progress bars, visit GitHub.

The following code uses PutObject as an example to show you how to view progress information:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
void percentage(int64_t consumed_bytes, int64_t total_bytes)
{
    assert(total_bytes >= consumed_bytes);
    printf("%%%" APR_INT64_T_FMT "\n", consumed_bytes * 100 / total_bytes);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_list_t resp_body;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* View the upload progress information when you upload an object. */
    resp_status = oss_do_put_object_from_file(oss_client_options, &bucket, &object, &file, NULL, NULL, percentage, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

You cannot use progress bars when you use oss_put_object_from_file and oss_put_object_from_buffer.

4.4.6.7. Upload callback

This topic describes how to use upload callback.

For the complete code used to implement upload callback, visit GitHub.

The following code provides an example on how to use upload callback:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
```

```
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    aos_pool_t *p = NULL;
    aos_status_t *s = NULL;
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *headers = NULL;
    oss_request_options_t *options = NULL;
    aos_table_t *resp_headers = NULL;
    aos_list_t resp_body;
    aos_list_t buffer;
    aos_buf_t *content;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    char b64_buf[1024];
    int b64_len;
    /* Use the JSON format. */
    /* Optional. Configure the value of the host field carried in the callback request header. */
    char *callback =  "{"
        "\"callbackUrl\":\"http://oss-demo.aliyuncs.com:23450\","
         "\"callbackHost\":\"yourCallbackHost\","
        "\"callbackBody\":\"bucket=${bucket}&object=${object}&size=${size}&mimeType=${mimeType}\","
        "\"callbackBodyType\":\"application/x-www-form-urlencoded\""
        "}";
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Initialize the parameters. */
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    init_options(options);
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&resp_body);
    aos_list_init(&buffer);
    content = aos_buf_pack(options->pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    /* Add callback to the header. */
    b64_len = aos_base64_encode((unsigned char*)callback, strlen(callback), b64_buf);
    b64_buf[b64_len] = '\0';
    headers = aos_table_make(p, 1);
    apr_table_set(headers, OSS_CALLBACK, b64_buf);
    /* Configure upload callback. */
    s = oss_do_put_object_from_buffer(options, &bucket, &object, &buffer,
        headers, NULL, NULL, &resp_headers, &resp_body);
    if (aos_status_is_ok(s)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Obtain the buffer length. */
    len = aos_buf_list_len(&resp_body);
    buf = (char *)aos_pcalloc(p, (apr_size_t)(len + 1));
    buf[len] = '\0';
    /* Copy the content from the buffer to the memory. */
    aos_list_for_each_entry(aos_buf_t, content, &resp_body, node) {
        size = aos_buf_size(content);
        memcpy(buf + pos, content->pos, (size_t)size);
        pos += size;
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.7. Download objects

4.4.7.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for C provides a variety of methods to upload objects.

For the complete object download code, visit GitHub.

You can use one of the following methods to download a single object:

- Download to local files
- Download to local memory
- Range download
- Resumable download

You can view the download progress by using the progress bar. For more information, see Use progress bars.

4.4.7.2. Download to local files

This topic describes how to download an object to a local file.

For the complete code of object download, visit GitHub.

The following code provides an example on how to download a specified object to a local file:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *params;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    params = aos_table_make(pool, 0);
    /* Download the object to your local file. If the name of the object is the same as that of the local file, the object replaces the local file. I
f the name of the object is different from that of the local file, the object is downloaded. */
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("Get object from file succeeded\n");
    } else {
        printf("Get object from file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note Compared with OSS SDK for C V1.0.0, this version adds the params parameter to the oss_get_object_to_file operation, and the values of the headers and params parameters can be NULL.

4.4.7.3. Download to local memory

This topic describes how to download an object to local memory.

For the complete code that is used to download objects, visit GitHub.

The following code provides an example on how to download a specified object to local memory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *params = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    /* Download the object to local memory. */
    resp_status = oss_get_object_to_buffer(oss_client_options, &bucket, &object,
                                headers, params, &buffer, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to buffer succeeded\n");
        /* Copy the downloaded content to the buffer. */
        len = aos_buf_list_len(&buffer);
        buf = aos_pcalloc(pool, len + 1);
        buf[len] = '\0';
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
        size = aos_buf_size(content);
            memcpy(buf + pos, content->pos, size);
            pos += size;
        }
    }
    else {
        printf("get object to buffer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.7.4. Range download

If you need only part of the data in an object, you can use range download to download data within the specified range.

The following code provides an example on how to perform range download:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *params = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *buf = NULL;
    int64_t len = 0;
    int64_t size = 0;
    int64_t pos = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    headers = aos_table_make(pool, 1);
    /* Set the range to byte 20 to byte 100. */
    apr_table_set(headers, "Range", "bytes=20-100");
    /* Download the object to local memory. */
    resp_status = oss_get_object_to_buffer(oss_client_options, &bucket, &object, headers, params, &buffer, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to buffer succeeded\n");
        /* Obtain the buffer length. */
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
            len += aos_buf_size(content);
        }
        buf = aos_pcalloc(pool, (apr_size_t)(len + 1));
        buf[len] = '\0';
        /* Copy the content in the buffer to the memory. */
        aos_list_for_each_entry(aos_buf_t, content, &buffer, node) {
            size = aos_buf_size(content);
            memcpy(buf + pos, content->pos, (size_t)size);
            pos += size;
        }
    }
    else {
        printf("get object to buffer failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.7.5. Resumable download

You may fail to download a large object if the network is unstable or other exceptions occur. In some cases, you may still fail to download the object even after multiple attempts. To solve this problem, OSS provides the resumable download feature.

You can use the `oss_resumable_clt_params_t` method to implement resumable download. The following table describes the parameters included in `oss_resumable_clt_params_t`.

Parameter	Description
thread_num	The number of concurrent threads. Default value: 1.
enable_checkpoint	Specifies whether to enable resumable upload. By default, resumable upload is disabled.
partSize	The size of each part. Valid values: 1 byte to 5 GB.
checkpoint_path	The path of the checkpoint file. Default value: {upload_file_path}.cp.

The `checkpoint` file stores information about the download progress.If you fail to download a part, the download continues based on the progress recorded in the `checkpoint` file. The `checkpoint` file is deleted after resumable download is complete.

Run the following code for resumable download:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_resumable_clt_params_t *clt_params;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* Use resumable download to download the object. */
    clt_params = oss_create_resumable_clt_params_content(pool, 1024 * 100, 3, AOS_TRUE, NULL);
    resp_status = oss_resumable_download_file(oss_client_options, &bucket, &object, &file, headers, NULL, clt_params, NULL, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("download succeeded\n");
    } else {
        printf("download failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.7.6. Use progress bars

Progress bars are used to indicate the progress of an upload or download.

For the complete code of download progress bars, visit GitHub.

The following code provides an example on how to view the progress information:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
const char *download_filename = "<yourDownloadFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
void percentage(int64_t consumed_bytes, int64_t total_bytes)
{
    assert(total_bytes >= consumed_bytes);
    printf("%%%" APR_INT64_T_FMT "\n", consumed_bytes * 100 / total_bytes);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *resp_headers = NULL;
    aos_list_t resp_body;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    /* View the upload progress while you upload an object. */
    resp_status = oss_do_put_object_from_file(oss_client_options, &bucket, &object, &file, NULL, NULL, percentage, &resp_headers, &resp_body);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from file succeeded\n");
    } else {
        printf("put object from file failed\n");
    }
    /* View the download progress while you download an object. */
    aos_pool_create(&pool, NULL);
    oss_client_options = oss_request_options_create(pool);
    init_options(oss_client_options);
    aos_str_set(&file, download_filename);
    resp_status = oss_do_get_object_to_file(oss_client_options, &bucket, &object, NULL, NULL, &file, percentage, NULL);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

oss_get_object_from_file and oss_get_object_from_buffer does not support the progress bar feature.

4.4.8. Manage objects

4.4.8.1. Overview

You can manage objects in a bucket by using a variety of operations including SetObjectAcl, GetObjectAcl, ListObjects, DeleteObject, and CopyObject.

By using the preceding operations, you can perform the following operations:

- Manage object ACLs
- Manage object metadata
- List objects
- Delete objects
- Copy objects

4.4.8.2. Manage object ACLs

This topic describes how to manage the access control list (ACL) of an object.

The following table describes the ACLs that can be configured for an object.

ACL	Description	Value
Inherited from the bucket	The ACL of an object is the same as that of the bucket in which the object is stored.	OSS_ACL_DEFAULT
Private	Only the owner or authorized users of the bucket can read and write the object.	OSS_ACL_PRIVATE
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	OSS_ACL_PUBLIC_READ
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	OSS_ACL_PUBLIC_READ_WRITE

The ACL of objects take precedence over that of buckets. For example, if the ACL of a bucket is private, while the object ACL is public read-write, all users can read and write the object. If an object is not configured with an ACL, its ACL is the same as that of its bucket by default.

You can run the following code to configure an ACL for an object:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    oss_acl_e oss_acl = OSS_ACL_PUBLIC_READ;
    /* Configure the object ACL. */
    resp_status = oss_put_object_acl(oss_client_options, &bucket, &object, oss_acl, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put object acl success! \n");
    } else {
        printf("put object acl failed! \n");
    }
    /* Query the ACL of the object. */
    aos_string_t oss_acl_string;
    resp_status = oss_get_object_acl(oss_client_options, &bucket, &object, &oss_acl_string, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object acl success! \n");
        printf("acl: %s \n", oss_acl_string.data);
    } else {
        printf("get object acl failed! \n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.8.3. Manage object metadata

After you upload an object to OSS or before you read an object from OSS, you can use the `oss_get_object_meta` operation to query object metadata such as the content length and object type.

The following code provides an example on how to configure and query object metadata:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret, i
s_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *headers;
    aos_list_t buffer;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_buf_t *content = NULL;
    char *content_length_str = NULL;
    char *object_type = NULL;
    char *object_author = NULL;
    int64_t content_length = 0;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    headers = aos_table_make(pool, 2);
    /* Configure user metadata. */
    apr_table_set(headers, "Expires", "Fri, 28 Feb 2032 05:38:42 GMT");
    apr_table_set(headers, "x-oss-meta-author", "oss");
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    /* Configure the object ACL. Upload object from the buffer. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object,
              &buffer, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer with md5 succeeded\n");
    } else {
        printf("put object from buffer with md5 failed\n");
    }
    /* Query the ACL of the object. */
    resp_status = oss_get_object_meta(oss_client_options, &bucket, &object, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        content_length_str = (char*)apr_table_get(resp_headers, OSS_CONTENT_LENGTH);
        if (content_length_str ! = NULL) {
            content_length = atol(content_length_str);
        }
        object_author = (char*)apr_table_get(resp_headers, OSS_AUTHORIZATION);
        object_type = (char*)apr_table_get(resp_headers, OSS_OBJECT_TYPE);
        printf("get object meta succeeded, object author:%s, object type:%s, content_length:%ld\n", object_author, object_type, content_length);
    } else {
        printf("req:%s, get object meta failed\n", resp_status->req_id);
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note For more information about HTTP headers, see RFC 2616.

4.4.8.4. List objects

This topic describes how to list the objects that are stored in a bucket.

For the complete code that is used to list objects, visit GitHub.

Objects are listed in alphabetical order. You can use the `oss_list_object` method to list objects in a bucket. The following table describes common parameters.

Parameter	Description
delimiter	The character used to group object names. Objects whose names contain the same string that ranges from the specified prefix to the next occurrence of the delimiter are grouped as a single result element in commonPrefixes.
prefix	The prefix that must be included in the names of returned objects.
max_ret	The maximum number of objects that can be listed at a time. Maximum value: 1000. Default value: 100.
marker	Lists the objects whose names are alphabetically after the object specified by marker.

List all objects

The following code provides an example on how to list all objects in a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note A maximum of 1,000 objects can be listed by default. If a bucket contains more than 1,000 objects, only the first 1,000 objects are returned, and the value of the truncated parameter in the response is true. Additionally, the next_marker parameter is returned as the start position for the next read operation. To list more than 1,000 objects at a time, you can modify the max_ret parameter. You can also use the marker parameter to perform multiple read operations.

List a specified number of objects

The following code provides an example on how to list a specified number of objects:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr_p
ool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    /* Set the maximum number of returned objects to 200. */
    params->max_ret = 200;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
    if (!aos_status_is_ok(resp_status))
    {
        printf("list object failed\n");
    }
    aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
        ++size;
        line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
            content->size.len, content->size.data,
            content->last_modified.len, content->last_modified.data);
        printf("%s", line);
    }
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is release
d. */
    aos_pool_destroy(pool);
    /* Before you exit the program, call aos_http_io_deinitialize to release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List objects whose names contain a specified prefix

The following code provides an example on how to list objects whose names contain a specified prefix:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr_p
ool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    /* List the objects whose names contain the specified prefix. */
    char *prefix = "<yourObjectPefix>";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is release
d. */
    aos_pool_destroy(pool);
    /* Before you exit the program, call aos_http_io_deinitialize to release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List the objects whose names are alphabetically after the object specified by marker

You can configure the marker parameter to specify the name of the object after which the list operation starts. The following code provides an example on how to list objects whose names are alphabetically after the object specified by marker:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr_p
ool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *prefix = "";
    /* List objects that come after the specified marker. */
    char *nextMarker = "<yourObjectMarker>";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is release
d. */
    aos_pool_destroy(pool);
    /* Before you exit the program, call aos_http_io_deinitialize to release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List objects by directory

OSS uses a flat structure for objects instead of a hierarchical structure to store objects. A directory is an object whose size is 0 and whose name ends with a forward slash (/). You can upload and download this object. By default, objects whose names end with a forward slash (/) are displayed as directories in the OSS console. For the complete sample code that is used to create directories, visit GitHub.

You can specify the delimiter and prefix parameters to list objects by directory.

- If you set prefix to a directory name in the request, objects and subdirectories whose names contain the prefix are listed.
- If you also set delimiter to a forward slash (/) in the request, the objects and subdirectories whose names start with the specified prefix in the directory are listed. Each subdirectory is listed as a single result element in commonPrefixes. The objects and directories in these subdirectories are not listed.

For more information about the complete sample code that is used to create a directory, visit GitHub.

Example: The following four objects are stored in a bucket: `oss.jpg` , `fun/test.jpg` , `fun/movie/001.avi` , and `fun/movie/007.avi` . The forward slash (/) is specified as the directory delimiter. The following code provides examples on how to list objects by directory.

- List all objects in a bucket

  The following code provides an example on how to list all objects in a bucket:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in
the APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- List all objects in a specified directory

  The following code provides an example on how to list all objects in a specified directory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr
_pool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    int size = 0;
    char *line = NULL;
    /* List the objects whose names contain the specified prefix. */
    char *prefix = "fun/";
    char *nextMarker = "";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is relea
sed. */
    aos_pool_destroy(pool);
    /* Before the project ends, call aos_http_io_deinitialize to release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

The following result is returned:

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
CommonPrefixes:
```

- List the objects and subdirectories in a specified directory

  The following code provides an example on how to list objects and subdirectories in a specified directory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr
_pool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    oss_list_object_common_prefix_t *commonPrefix = NULL;
    int size = 0;
    char *line = NULL;
    /* List the objects whose names contain the specified prefix. */
    char *prefix = "fun/";
    char *nextMarker = "";
    /* Set the delimiter to a forward slash (/). */
    char *delimiter = "/";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    aos_str_set(&params->delimiter, delimiter);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        aos_list_for_each_entry(oss_list_object_common_prefix_t, commonPrefix, &params->common_prefix_list, node) {
            line = apr_psprintf(pool, "%.*s", commonPrefix->prefix.len, commonPrefix->prefix.data);
            printf("%s", line);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
```

```
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is relea
sed. */
    aos_pool_destroy(pool);
    /* Before you exit the program, call aos_http_io_deinitialize to release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

The following result is returned:

```
Objects:
fun/test.jpg
CommonPrefixes:
fun/movie/
```

- List the sizes of objects in a specified directory

  The following code provides an example on how to list the sizes of objects in a specified directory:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int64_t calculateFolderLength (aos_string_t bucketName, char* folder)
{
    int64_t size = 0;
    oss_list_object_params_t *params = NULL;
    char *nextMarker = "";
    aos_status_t *resp_status = NULL;
    aos_pool_t *p = NULL;
    oss_request_options_t *options = NULL;
    oss_list_object_content_t* content = NULL;
    aos_pool_create(&p, NULL);
    options = oss_request_options_create(p);
    params = oss_create_list_object_params(p);
    params->max_ret = 10;
    aos_str_set(&params->prefix, folder);
    aos_str_set(&params->marker, nextMarker);
    /* List all objects. */
    do {
        resp_status = oss_list_object(options, &bucketName, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            size += content->size.len;
        }
        nextMarker = apr_psprintf(p, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is relea
sed. */
    aos_pool_destroy(p);
    return size;
}
int main(int argc, char *argv[])
{
    /* Call aos_http_io_initialize in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. The code used to create a memory pool is included in the APR library. aos_pool_t is equivalent to apr
_pool_t. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* The memory consumed by options is allocated by the memory pool and is released with the pool. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_status_t *resp_status = NULL;
    oss_list_object_params_t *params = NULL;
    oss_list_object_content_t *content = NULL;
    oss_list_object_common_prefix_t *commonPrefix = NULL;
    int size = 0;
    char *line = NULL;
    /* List the objects whose names contain the specified prefix. */
    char *prefix = "fun/";
    char *nextMarker = "";
    /* Set the delimiter to a forward slash (/). */
    char *delimiter = "/";
    aos_str_set(&bucket, bucket_name);
    params = oss_create_list_object_params(pool);
    params->max_ret = 100;
    aos_str_set(&params->prefix, prefix);
    aos_str_set(&params->marker, nextMarker);
    aos_str_set(&params->delimiter, delimiter);
    printf("Object\tSize\tLastModified\n");
    /* List all objects. */
    do {
        resp_status = oss_list_object(oss_client_options, &bucket, params, NULL);
        if (!aos_status_is_ok(resp_status))
        {
            printf("list object failed\n");
            break;
        }
        aos_list_for_each_entry(oss_list_object_content_t, content, &params->object_list, node) {
            ++size;
            line = apr_psprintf(pool, "%.*s\t%.*s\t%.*s\n", content->key.len, content->key.data,
                content->size.len, content->size.data,
                content->last_modified.len, content->last_modified.data);
            printf("%s", line);
        }
        aos_list_for_each_entry(oss_list_object_common_prefix_t, commonPrefix, &params->common_prefix_list, node) {
            line = apr_psprintf(pool, "%.*s", commonPrefix->prefix.len, commonPrefix->prefix.data);
            int64_t size = calculateFolderLength(bucket, commonPrefix->prefix.data);
            printf("Total %d\n", size);
        }
        nextMarker = apr_psprintf(pool, "%.*s", params->next_marker.len, params->next_marker.data);
        aos_str_set(&params->marker, nextMarker);
        aos_list_init(&params->object_list);
        aos_list_init(&params->common_prefix_list);
    } while (params->truncated == AOS_TRUE);
    printf("Total %d\n", size);
    /* Release the memory pool after the request is complete. In this case, the memory allocated to various resources used for the request is relea
sed. */
    aos_pool_destroy(pool);
    /* Before you exit the program, call aos_http_io_deinitialize to release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.8.5. Delete objects

This topic describes how to delete objects.

Warning Deleted objects cannot be recovered. Exercise caution when you delete objects.

For the complete code that is used to delete an object, visit GitHub.

Delete a single object

The following code provides an example on how to delete a single object:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign char* data to a bucket of the aos_string_t type. */
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Delete the object. */
    resp_status = oss_delete_object(oss_client_options, &bucket, &object, &resp_headers);
    /* Determine whether the object is deleted. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeed\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete multiple objects

The following code provides an example on how to delete multiple objects:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name1 = "<yourObjectName1>";
const char *object_name2 = "<yourObjectName2>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object1;
    aos_string_t object2;
    int is_quiet = 1;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object1, object_name1);
    aos_str_set(&object2, object_name2);
    /* Create a list of objects that you want to delete. */
    aos_list_t object_list;
    aos_list_t deleted_object_list;
    oss_object_key_t *content1;
    oss_object_key_t *content2;
    aos_list_init(&object_list);
    aos_list_init(&deleted_object_list);
    content1 = oss_create_oss_object_key(pool);
    aos_str_set(&content1->key, object_name1);
    aos_list_add_tail(&content1->node, &object_list);
    content2 = oss_create_oss_object_key(pool);
    aos_str_set(&content2->key, object_name2);
    aos_list_add_tail(&content2->node, &object_list);
    /* Delete the objects in the list. `is_quiet` specifies whether to return the deletion result. */
    resp_status = oss_delete_objects(oss_client_options, &bucket, &object_list, is_quiet, &resp_headers, &deleted_object_list);
    if (aos_status_is_ok(resp_status)) {
        printf("delete objects succeeded\n");
    } else {
        printf("delete objects failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete multiple objects whose names contain a specified prefix

You can delete multiple objects whose names contain a specified prefix to delete a folder. For example, if you delete objects whose names contain dir/, the dir folder is deleted. The following code provides an example on how to delete multiple objects whose names contain a specified prefix:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_prefix = "<yourObjectNamePrefix>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the APR li
brary. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t prefix;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&prefix, object_prefix);
    /* Delete the objects whose names contain the specified prefix. */
    resp_status = oss_delete_objects_by_prefix(oss_client_options, &bucket, &prefix);
    if (aos_status_is_ok(resp_status)) {
        printf("delete objects by prefix succeeded\n");
    } else {
        printf("delete objects by prefix failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Warning If you set the prefix to an empty string or NULL, all objects in the bucket are deleted. Exercise caution when you use this operation.

4.4.8.6. Copy objects

This topic describes how to copy an object.

Limits on object copying:

- You must have read permissions on the source object and write permissions on the destination bucket.
- You cannot copy an object across regions.

Copy a small object

The following code provides an example on how to copy an object smaller than or equal to 1 GB in size:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *source_bucket_name = "<yourSourceBucketName>";
const char *source_object_name = "<yourSourceObjectName>";
const char *dest_bucket_name = "<yourDestBucketName>";
const char *dest_object_name = "<yourDestObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, acces_key_secret, is_
cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t source_bucket;
    aos_string_t source_object;
    aos_string_t dest_bucket;
    aos_string_t dest_object;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&source_bucket, source_bucket_name);
    aos_str_set(&source_object, source_object_name);
    aos_str_set(&dest_bucket, dest_bucket_name);
    aos_str_set(&dest_object, dest_object_name);
    headers = aos_table_make(pool, 0);
    /* Copy the object. */
    resp_status = oss_copy_object(oss_client_options, &source_bucket, &source_object, &dest_bucket, &dest_object, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("copy object succeeded\n");
    } else {
        printf("copy object failed\n");
    }
    /* Release the memory pool. The memory allocated to various resources used for the request is released. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Copy a large object

We recommend that you use the oss_upload_part_copy operation to copy an object larger than 1 GB in size. The following code provides an example on how to perform multipart copy:

```
#include "oss_api.h"
#include "aos_http_io.h"
#include <sys/stat.h>
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *source_bucket_name = "<yourSourceBucketName>";
const char *source_object_name = "<yourSourceObjectName>";
const char *dest_bucket_name = "<yourDestBucketName>";
const char *dest_object_name = "<yourDestObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
```

```
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME to access OSS. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int64_t get_file_size(const char *file_path)
{
    int64_t filesize = -1;
    struct stat statbuff;
    if(stat(file_path, &statbuff) < 0){
        return filesize;
    } else {
        filesize = statbuff.st_size;
    }
    return filesize;
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR library
. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This parameter indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, acces_key_secret, is_
cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate a memory chunk in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Call oss_client_options to initialize the client options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t dest_bucket;
    aos_string_t dest_object;
    aos_string_t upload_id;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    oss_list_upload_part_params_t *list_upload_part_params;
    oss_upload_part_copy_params_t *upload_part_copy_params1;
    oss_upload_part_copy_params_t *upload_part_copy_params2;
    oss_list_part_content_t *part_content;
    aos_list_t complete_part_list;
    oss_complete_part_content_t *complete_content;
    aos_table_t *list_part_resp_headers = NULL;
    aos_table_t *complete_resp_headers = NULL;
    int part1 = 1;
    int part2 = 2;
    int64_t range_start1 = 0;
    int64_t range_end1 = 6000000;//not less than 5MB
    int64_t range_start2 = 6000001;
    int64_t range_end2;
    int max_ret = 1000;
    headers = aos_table_make(pool, 0);
    aos_str_set(&dest_bucket, dest_bucket_name);
    aos_str_set(&dest_object, dest_object_name);
    resp_status = oss_init_multipart_upload(oss_client_options, &dest_bucket, &dest_object, &upload_id, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("init multipart upload succeeded, upload_id is %s\n", upload_id.data);
    } else {
        printf("init multipart upload failed\n");
    }
    /* Copy the first part. */
    upload_part_copy_params1 = oss_create_upload_part_copy_params(pool);
    aos_str_set(&upload_part_copy_params1->source_bucket, source_bucket_name);
    aos_str_set(&upload_part_copy_params1->source_object, source_object_name);
    aos_str_set(&upload_part_copy_params1->dest_bucket, dest_bucket_name);
    aos_str_set(&upload_part_copy_params1->dest_object, dest_object_name);
    aos_str_set(&upload_part_copy_params1->upload_id, upload_id.data);
    upload_part_copy_params1->part_num = part1;
    upload_part_copy_params1->range_start = range_start1;
    upload_part_copy_params1->range_end = range_end1;
    headers = aos_table_make(pool, 0);
    resp_status = oss_upload_part_copy(oss_client_options, upload_part_copy_params1, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("upload part 1 copy succeeded\n");
    } else {
        printf("upload part 1 copy failed\n");
    }
```

```
    /* Copy the second part. */
    range_end2 = get_file_size(local_filename) - 1;
    upload_part_copy_params2 = oss_create_upload_part_copy_params(pool);
    aos_str_set(&upload_part_copy_params2->source_bucket, source_bucket_name);
    aos_str_set(&upload_part_copy_params2->source_object, source_object_name);
    aos_str_set(&upload_part_copy_params2->dest_bucket, dest_bucket_name);
    aos_str_set(&upload_part_copy_params2->dest_object, dest_object_name);
    aos_str_set(&upload_part_copy_params2->upload_id, upload_id.data);
    upload_part_copy_params2->part_num = part2;
    upload_part_copy_params2->range_start = range_start2;
    upload_part_copy_params2->range_end = range_end2;
    headers = aos_table_make(pool, 0);
    resp_status = oss_upload_part_copy(oss_client_options, upload_part_copy_params2, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("upload part 2 copy succeeded\n");
    } else {
        printf("upload part 2 copy failed\n");
    }
    /* List the copied parts. */
    headers = aos_table_make(pool, 0);
    list_upload_part_params = oss_create_list_upload_part_params(pool);
    list_upload_part_params->max_ret = max_ret;
    aos_list_init(&complete_part_list);
    resp_status = oss_list_upload_part(oss_client_options, &dest_bucket, &dest_object, &upload_id, list_upload_part_params, &list_part_resp_headers);
    aos_list_for_each_entry(oss_list_part_content_t, part_content, &list_upload_part_params->part_list, node) {
       complete_content = oss_create_complete_part_content(pool);
       aos_str_set(&complete_content->part_number, part_content->part_number.data);
       aos_str_set(&complete_content->etag, part_content->etag.data);
       aos_list_add_tail(&complete_content->node, &complete_part_list);
    }
    /* Complete the multipart copy task. */
    resp_status = oss_complete_multipart_upload(oss_client_options, &dest_bucket, &dest_object, &upload_id, &complete_part_list, headers, &complete_r
esp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("complete multipart upload succeeded\n");
    } else {
        printf("complete multipart upload failed\n");
    }
    /* Release the memory pool. The memory allocated to various resources used for the request is released. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.9. Authorized access

This topic describes how to authorize access to OSS by using STS or a signed URL.

Use STS for temporary access authorization

Alibaba Cloud STS is a web service that provides a temporary access token to a cloud computing user. You can use STS to grant a set of temporary access credentials that have a custom validity period and custom permissions to a third-party application or a RAM user managed by you.

STS provides the following benefits:

- You need only to generate an access token and send the access token to a third-party application. You do not need to expose your AccessKey pair to the third-party application. You can specify the access permissions and validity period of this token.
- The token automatically expires after the validity period. Therefore, you do not need to manually revoke the access permissions of a token.

The following sample code provides an example on how to upload a string to an object by using temporary access credentials obtained from STS.

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *sts_token = "<yourStsToken>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *object_content = "More than just cloud.";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    aos_str_set(&options->config->sts_token, sts_token);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in th
e APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret, is
_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    aos_buf_t *content = NULL;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign the char* data to the bucket. */
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, object_content, strlen(object_content));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload the object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the object is uploaded. */
    if (aos_status_is_ok(resp_status)) {
        printf("put object from buffer succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Use a signed URL for temporary access authorization

can generate a signed URL and provide the URL to a visitor for temporary access. When you generate a signed URL, you can specify the validity period of the URL to limit the period of time during which the visitor can access OSS.

- Use a signed URL to upload an object

  The following code provides an example on how to use a signed URL to upload an object:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in
the APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_http_request_t *req;
    apr_time_t now;
    char *url_str;
    aos_string_t url;
    int64_t expire_time;
    int one_hour = 3600;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    expire_time = now / 1000000 + one_hour;
    headers = aos_table_make(pool, 0);
    req = aos_http_request_create(pool);
    req->method = HTTP_PUT;
    now = apr_time_now(); /* Unit: microseconds */
    expire_time = now / 1000000 + one_hour;
    /* Generate a signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    aos_str_set(&url, url_str);
    printf ("The signed URL used to upload the object: %s\n", url_str);
    /* Use the signed URL to upload the object. */
    resp_status = oss_put_object_from_file_by_url(oss_client_options, &url, &file, headers, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put objects by signed url succeeded\n");
    } else {
        printf("put objects by signed url failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Use a signed URL to download an object

  The following code provides an example on how to use a signed URL to download an object:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
const char *local_filename = "<yourLocalFilename>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize the aos_string_t data type. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. The value 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as the timeout period. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code that is used to create a memory pool is included in
the APR library. */
    aos_pool_t *pool;
    /* Create a memory pool. The value of the second parameter is NULL. This value indicates that the pool does not inherit other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter includes global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_http_request_t *req;
    apr_time_t now;
    char *url_str;
    aos_string_t url;
    int64_t expire_time;
    int one_hour = 3600;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    aos_str_set(&file, local_filename);
    expire_time = now / 1000000 + one_hour;
    headers = aos_table_make(pool, 0);
    params = aos_table_make(pool, 0);
    req = aos_http_request_create(pool);
    req->method = HTTP_GET;
    now = apr_time_now(); /* Unit: microseconds. */
    expire_time = now / 1000000 + one_hour;
    /* Generate a signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    aos_str_set(&url, url_str);
    /* Use the signed URL to download the object. */
    resp_status = oss_get_object_to_file_by_url(oss_client_options, &url, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object succeeded\n");
    } else {
        printf("get object failed\n");
    }
    /* Release the memory pool. This operation releases the memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.10. IMG

OSS Image Processing (IMG) is a secure, cost-effective, and highly reliable image processing service that can process large amounts of data. After you upload source images to OSS, you can call RESTful API operations to process the images anytime, anywhere, and on any Internet device.

For the complete IMG code, visit GitHub.

Basic features

OSS IMG provides the following features:

- Query image information.
- Convert image formats.
- Resize, crop, and rotate images.
- Apply image effect.
- Add image, text, and text-and-image watermarks to images.
- Customize image processing styles. Log on to the OSS console. Click a bucket name to go to the bucket details page. Click the **Image Processing (IMG)** tab and click **Create Rule** to customize an image processing style.
- Use cascade processing to call multiple IMG features.

Usage

You can use standard `HTTP GET` requests to access IMG. All processing parameters are encoded in the query string of a URL.

- Anonymous access

  You can anonymously access a processed image by using a third-level domain in the following format:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
  ```

Parameter	Description
bucket	The name of the bucket.
endpoint	The domain name used to access the region.
object	The name of the image object.
image	The identifier reserved by IMG.
action	The operations performed on images, such as scaling, cropping, and rotating.
param	The parameters corresponding to the operations performed on the images.

  - Basic operations

    For example, you can scale an image to a width of 100 and adjust the height based on the ratio:

    ```
    http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
    ```

  - Customize an image style

    You can anonymously access IMG by using a third-level domain in the following format:

    ```
    http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
    ```

    - style: the identifier reserved by IMG for the custom style.
    - yourStyleName: the name of the custom style. The name is specified by the Rule Name parameter when you create the custom style in the console.

    Example:

    ```
    http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
    ```

  - Cascade processing

    You can use cascade processing to perform multiple operations in sequence on an image by using a URL in the following format:

    ```
    http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
    ```

    Example:

    ```
    http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
    ```

  - Access by using HTTPS

    IMG supports access by using HTTPS. Example:

    ```
    https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
    ```

- Authorized access

  If the ACL of an image object is private, you must have permissions on the image object before you can perform operations on the image object.

Bucket ACL	Object ACL
private	default
Any ACL	private

  The following code provides an example on how to generate a signed URL to access IMG:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code to create a memory pool is included in the APR libra
ry. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This value indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secret,
is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_table_t *params = NULL;
    aos_http_request_t *req;
    char *url_str;
    apr_time_t now;
    int64_t expire_time;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Process the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100");
    req = aos_http_request_create(pool);
    req->method = HTTP_GET;
    req->query_params = params;
    /* Specify the validity period (expire_time) in seconds. */
    now = apr_time_now();
    expire_time = now / 1000000 + 10 * 60;
    /* Generate a signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &object, expire_time, req);
    printf("url: %s\n", url_str);
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Access by using SDKs

  You can use OSS SDKs to access and process image objects of any ACLs.

  - Basic operations

    The following code provides an example on how to perform basic IMG operations including obtaining image information, converting image formats, resizing, cropping, rotating, watermarking, and applying effects:

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
```

```
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the AP
R library. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This value indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secre
t, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Scale the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-resize.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Crop the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/crop,w_100,h_100,x_100,y_100,r_1");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-crop.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Rotate the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/rotate,90");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-rotate.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Sharpen the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/sharpen,100");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-sharpen.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Add watermarks to the image. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-watermark.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
```

```
        printf("get object to file failed\n");
    }
    /* Convert the image format. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/format,png");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-format.png");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Obtain image information. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/info");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-info.txt");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Custom image styles

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the AP
R library. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This value indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secre
t, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Customize the image style. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "style/<yourCustomStyleName>");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-custom-style.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Cascade processing

```
#include "oss_api.h"
#include "aos_http_io.h"
const char *endpoint = "<yourEndpoint>";
const char *access_key_id = "<yourAccessKeyId>";
const char *access_key_secret = "<yourAccessKeySecret>";
const char *bucket_name = "<yourBucketName>";
const char *object_name = "<yourObjectName>";
void init_options(oss_request_options_t *options)
{
    options->config = oss_config_create(options->pool);
    /* Use a char* string to initialize aos_string_t. */
    aos_str_set(&options->config->endpoint, endpoint);
    aos_str_set(&options->config->access_key_id, access_key_id);
    aos_str_set(&options->config->access_key_secret, access_key_secret);
    /* Specify whether to use CNAME. A value of 0 indicates that CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters such as timeout periods. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the aos_http_io_initialize method in main() to initialize global resources such as networks and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Create a memory pool to manage memory. aos_pool_t is equivalent to apr_pool_t. The code used to create a memory pool is included in the AP
R library. */
    aos_pool_t *pool;
    /* Create a memory pool. The second parameter is NULL. This value indicates that the pool does not inherit any other memory pool. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter specifies global configuration information such as endpoint, access_key_id, access_key_secre
t, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate the memory resources in the memory pool to the options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize the parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t file;
    aos_table_t *headers = NULL;
    aos_table_t *params = NULL;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Perform cascade processing. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100,h_100/rotate,90");
    /* Download the processed image to a local file. */
    aos_str_set(&file, "example-cascade.jpg");
    resp_status = oss_get_object_to_file(oss_client_options, &bucket, &object, headers, params, &file, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get object to file succeeded\n");
    } else {
        printf("get object to file failed\n");
    }
    /* Release the memory pool, which releases memory resources allocated for the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4.11. Error handling

This topic describes how to handle errors when you use OSS SDK for C.

Exception handling

If an error occurs when you send a request by using OSS SDK for C, an corresponding error message is returned by aos_status_s. The following table describes the attributes of aos_status_s.

Error code	Description	Type
code	The HTTP status code of the error request.	Integer
error_code	The error code returned by OSS.	String
error_msg	The error message returned by OSS.	String
req_id	The UUID that identifies the request. You can provide req_id to OSS development engineers for help.	String

Timeout handling

If the code in the returned aos_status_t parameter is not 2xx and error_code indicates -992 or -995, the connection or request times out. You can send the request again.

You can use aos_should_retry(aos_status_t *) in the aos_status.h file to determine whether the error code returned requires a retry. If a value of 1 is returned, you can try again.

Common error codes

Error code	Error message	HTTP status code
AccessDenied	The error message returned because the access is denied.	403
BucketAlreadyExists	The error message returned because the bucket already exists.	409
BucketNotEmpty	The error message returned because the bucket is not empty.	409
EntityTooLarge	The error message returned because the entity exceeds the maximum allowed size.	400
EntityTooSmall	The error message returned because the entity is smaller than the minimum allowed size.	400
FileGroupTooLarge	The error message returned because the object group exceeds the maximum allowed size.	400
FilePartNotExist	The error message returned because the specified part does not exist.	400
FilePartStale	The error message returned because the part has expired.	400
InvalidArgument	The error message returned because the parameter format is invalid.	400
InvalidAccessKeyId	The error message returned because the specified Accesskey ID does not exist.	403
InvalidBucketName	The error message returned because the bucket name is invalid.	400
InvalidDigest	The error message returned because the digest is invalid.	400
InvalidObjectName	The error message returned because the object name is invalid.	400
InvalidPart	The error message returned because the part is invalid.	400
InvalidPartOrder	The error message returned because the part order is invalid.	400
InvalidTargetBucketForLogging	The error message returned because the target bucket specified for logging is invalid.	400
InternalError	The error message returned because an internal OSS error occurs.	500
MalformedXML	The error message returned because the XML format is invalid.	400
MethodNotAllowed	The error message returned because the method is not supported.	405
MissingArgument	The error message returned because the required parameters are not set.	411
MissingContentLength	The error message returned because the content length is not specified.	411
NoSuchBucket	The error message returned because the specified bucket does not exist.	404
NoSuchKey	The error message returned because the specified object does not exist.	404
NoSuchUpload	The error message returned because the specified multipart upload ID does not exist.	404
NotImplemented	The error message returned because the method cannot be implemented.	501
PreconditionFailed	The error message returned because a precondition error occurs.	412

Error code	Error message	HTTP status code
RequestTimeTooSkewed	The error message returned because the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes.	403
RequestTimeout	The error message returned because the request times out.	400
SignatureDoesNotMatch	The error message returned because the signature is invalid.	403
TooManyBuckets	The error message returned because the number of the buckets managed by a user exceeds the limit.	400

4.5. Go-SDK

4.5.1. Preface

This topic describes the sample code of OSS SDK for Go in various use cases.

SDK source code and API documents

For the source code of OSS SDK for Go, visit GitHub. For more information, see OSS SDK for Go API documents.

Sample programs

OSS SDK for Go provides a variety of sample programs for your reference or use. The following table describes the sample programs provided.

Sample file	Description
new_bucket.go	Shows you how to initialize a client.
put_object.go	Shows you how to use simple upload and resumable upload.
append_object.go	Shows you how to use append upload.
get_object.go	Shows you how to use streaming download, range download, and resumable download.
delete_object.go	Shows you how to delete a single object and multiple objects.
copy_object.go	Shows you how to use object copy and resumable copy.
list_objects.go	Shows you how to list objects, including list by default or specified parameter.
object_meta.go	Shows you how to configure and query object metadata.
object_acl.go	Shows you how to configure and query object ACL.
cname_sample.go	Shows you how to use CNAME.
create_bucket.go	Shows you how to create a bucket.
list_buckets.go	Shows you how to list buckets, including list by default or specified parameter.
bucket_acl.go	Shows you how to configure and query bucket ACL.
bucket_referer.go	Shows you how to configure, query, and delete the Referer whitelist for a bucket.
bucket_logging.go	Shows you how to configure, query, and delete access logs for a bucket.
bucket_lifecycle.go	Shows you how to configure, query, and delete lifecycle rules for objects in a bucket.
bucket_cors.go	Shows you how to configure, query, and delete the cross-origin resource sharing (CORS) rules for a bucket.

4.5.2. Installation

This topic describes how to install OSS SDK for Go.

Prerequisites

- Environment

  Go 1.6 or later is required.

  For more information about how to download and install the environment for Go, visit Installing Go from source. After Go is installed, create a new system variable GOPATH, and set the value to your code directory. For more information about GOPATH, run the `go help gopath` command.

- Check the Go version

  Run the `go version` command to view the version of Go.

Download OSS SDK for Go

- Download from GitHub
- Download previous versions

Install OSS SDK for Go

Run the following command to install OSS SDK for Go:

```
go get github.com/aliyun/aliyun-oss-go-sdk/oss
```

Note When you run the go get command to install the SDK, no prompts are displayed. If the installation times out, run the preceding command again.

View the version of OSS SDK for Go

Run the following code to view the version of OSS SDK for Go:

```
package main
import (
  "fmt"
  "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
  fmt.Println("OSS Go SDK Version: ", oss.Version)
}
```

4.5.3. Initialization

OSSClient serves as the client of OSS SDK for Go to manage OSS resources such as buckets and objects.

To create an OSSClient instance, you must specify an endpoint. For more information about endpoints, see Obtain an endpoint.

To create an OSSClient instance, you must specify an AccessKey pair. For more information about how to obtain an AccessKey pair, see Obtain an AccessKey pair.

The following table describes the parameters you can configure for this operation.

Parameter	Description	Default
UseCname	Specifies whether to use CNAME.	No
Timeout	The timeout period for request, including the connection timeout period and socket read/write timeout period. Unit: seconds.	The connection timeout period is 30 seconds. The read/write timeout period is 60 seconds.
SecurityToken	The temporary security token for users.	None
EnableMD5	Specifies whether to enable MD5 verification. We recommend that you enable CRC, which is more efficient than MD5 verification.	No
EnableCRC	Specifies whether to enable CRC.	Yes
Proxy	The proxy server. Example: `http://8.8.8.8:3128`.	None
AuthProxy	The proxy server that requires an account and a password to log on.	None

Create an OSSClient instance based on an OSS endpoint

The following code provides an example on how to create an OSSClient instance based on an OSS endpoint:

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
```

Create an OSSClient instance based on a custom domain name

For the complete code of custom domain names, visit GitHub.

The following code provides an example on how to create an OSSClient instance based on a custom domain name:

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
// oss.UseCname(true) indicates that CNAME is enabled. If CNAME is enabled, a custom domain name can be bound to a bucket.
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>", oss.UseCname(true))
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
```

Notice When you use CNAME, you cannot list buckets because a custom domain name is bound to a specified bucket.

Create an OSSClient instance based on STS

The following code provides an example on how to create an OSSClient instance based on Security Token Service (STS):

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>", oss.SecurityToken("<yourSecurityToken>"))
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
```

Configure the connection timeout period

The following code provides an example on how to configure the connection timeout period:

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
// oss.Timeout(10, 120) indicates that the HTTP connection timeout period is 10 seconds and the HTTP read/write timeout period is 120 seconds. The de
fault HTTP connection timeout period is 30 seconds. The default HTTP read/write timeout period is 60 seconds. A value of 0 indicates that the connect
ion never times out.
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>", oss.Timeout(10, 120))
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
```

Disable CRC

By default, CRC is enabled to ensure data integrity when you upload or download objects. The following code provides an example on how to disable CRC:

Warning We recommend that you do not disable CRC. If you disable CRC, data integrity may be affected during object upload and download.

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>", oss.EnableCRC(false))
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
```

4.5.4. Quick start

This topic describes how to use OSS SDK for Go to perform routine operations, such as creating buckets, uploading objects, and downloading objects.

Create buckets

A bucket is a global namespace in OSS. It is similar to a data container that stores files. The following code provides an example on how to create a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err != nil {
        handleError(err)
    }
    // Create the bucket.
    err = client.CreateBucket(bucketName)
    if err != nil {
        handleError(err)
    }
}
```

For more information about how to obtain an AccessKey ID and AccessKey secret, see Obtain an AccessKey pair.

For more information about how to obtain an endpoint, see Obtain an endpoint.

Upload objects

The following code provides an example on how to upload objects to OSS by using stream upload:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    // <yourObjectName> indicates the full path of the object you want to upload to OSS, and must include the extension of the object name. Example:
abc/efg/123.jpg.
    objectName := "<yourObjectName>"
    // <yourLocalFileName> consists of a local file path and a file name with extension. Example: /users/local/myfile.txt.
    localFileName := "<yourLocalFileName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err != nil {
        handleError(err)
    }
    // Query the bucket.
    bucket, err := client.Bucket(bucketName)
    if err != nil {
        handleError(err)
    }
    // Upload the object.
    err = bucket.PutObjectFromFile(objectName, localFileName)
    if err != nil {
        handleError(err)
    }
}
```

Download objects

The following code provides an example on how to download an object to your local device:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    // <yourObjectName> indicates the full path of the object that you want to download from OSS. <yourObjectName> must include the extension of the
object name. Example: abc/efg/123.jpg.
    objectName := "<yourObjectName>"
    downloadedFileName := "<yourDownloadedFileName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err != nil {
        handleError(err)
    }
    // Query the bucket.
    bucket, err := client.Bucket(bucketName)
    if err != nil {
        handleError(err)
    }
    // Download the object.
    err = bucket.GetObjectToFile(objectName, downloadedFileName)
    if err != nil {
        handleError(err)
    }
}
```

List objects

The following code provides an example on how to list objects in a specified bucket. By default, a maximum of 100 objects can be listed.

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err != nil {
        HandleError(err)
    }
    // Query the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err != nil {
        HandleError(err)
    }
    // List objects.
    marker := ""
    for {
        lsRes, err := bucket.ListObjects(oss.Marker(marker))
        if err != nil {
            HandleError(err)
        }
        // Display the listed objects. By default, a maximum of 100 objects are returned at a time.
        for _, object := range lsRes.Objects {
            fmt.Println("Bucket: ", object.Key)
        }
        if lsRes.IsTruncated {
            marker = lsRes.NextMarker
        } else {
            break
        }
    }
}
```

Delete objects

The following code provides an example on how to delete an object:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func handleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
    endpoint := "http://oss-cn-hangzhou.aliyuncs.com"
    accessKeyId := "<yourAccessKeyId>"
    accessKeySecret := "<yourAccessKeySecret>"
    bucketName := "<yourBucketName>"
    // <yourObjectName> indicates the full path of the object that you want to delete from OSS. The path must include the extension of the object nam
e. Example: abc/efg/123.jpg.
    objectName := "<yourObjectName>"
    // Create an OSSClient instance.
    client, err := oss.New(endpoint, accessKeyId, accessKeySecret)
    if err != nil {
        handleError(err)
    }
    // Query the bucket.
    bucket, err := client.Bucket(bucketName)
    if err != nil {
        handleError(err)
    }
    // Delete the object.
    err = bucket.DeleteObject(objectName)
    if err != nil {
        handleError(err)
    }
}
```

For more information about how to delete objects, see Delete objects.

4.5.5. Buckets

4.5.5.1. Create buckets

A bucket is a container used to store objects in Object Storage Service (OSS). Each object is contained in a bucket. This topic describes how to create a bucket.

For the complete code that is used to create a bucket, visit GitHub.

The following code provides an example on how to create a bucket:

```
package main
    import (
        "fmt"
        "os"
        "github.com/aliyun/aliyun-oss-go-sdk/oss"
    )
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Create the bucket. By default, the storage class of the bucket is Standard, and the ACL of the bucket is private.
    err = client.CreateBucket("<yourBucketName>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.5.2. List buckets

A bucket is a container that is used to store objects in Object Storage Service (OSS). All objects in OSS are contained in buckets. Bucket names are listed in alphabetical order. You can list buckets that belong to the current Alibaba Cloud account in all regions and meet the specific conditions.

For the complete code that is used to list buckets, visit GitHub.

List all buckets

Buckets are listed in alphabetical order. You can list all buckets or the buckets that meet the specified conditions.

The following sample code provides an example on how to list all buckets that belong to an Apsara Stack tenant account:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List the buckets.
    marker := ""
    for {
        lsRes, err := client.ListBuckets(oss.Marker(marker))
        if err != nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        // By default, a maximum of 100 buckets are listed at a time.
        for _, bucket := range lsRes.Buckets {
            fmt.Println("Bucket: ", bucket.Name)
        }
        if lsRes.IsTruncated {
            marker = lsRes.NextMarker
        } else {
            break
        }
    }
}
```

List the buckets whose names contain a specified prefix

The following code provides an example on how to list the buckets whose names contain a specified prefix:

```
// List the buckets whose names contain the specified prefix.
    lsRes, err = client.ListBuckets(oss.Prefix("<yourBucketPrefix>"))
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the list of buckets.
    fmt.Println("Buckets with prefix: ", lsRes.Buckets)
    for _, bucket := range lsRes.Buckets {
        fmt.Println("Bucket with prefix: ", bucket.Name)
    }
```

List the buckets whose names are alphabetically after the bucket specified by marker

The following code provides an example on how to list the buckets whose names are alphabetically after the bucket specified by the marker parameter:

```
    // List the buckets whose names are alphabetically after the bucket specified by marker.
    lsRes, err = client.ListBuckets(oss.Marker("<yourBucketMarker>"))
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the list of buckets.
    fmt.Println("My buckets with marker :", lsRes.Buckets)
    for _, bucket := range lsRes.Buckets {
        fmt.Println("Bucket with marker: ", bucket.Name)
    }
```

List a specified number of buckets

The following code provides an example on how to list up to 500 buckets:

```
    // Set the maximum number of buckets to list to 500. Maximum value: 1000. Default value: 100.
    lsRes, err = client.ListBuckets(oss.MaxKeys(500))
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the list of buckets.
    fmt.Println("My buckets max num:", lsRes.Buckets)
    for _, bucket := range lsRes.Buckets {
        fmt.Println("Bucket with maxKeys: ", bucket.Name)
    }
```

4.5.5.3. Manage bucket ACLs

A bucket is a container used to store objects in Object Storage Service (OSS). Every object is contained in a bucket. This topic describes how to configure and query the access control list (ACL) of a bucket.

For the complete code used to manage the ACL of a bucket, visit GitHub.

Configure ACL for a bucket

The following table describes the bucket ACLs.

ACL	Description	Value
Private	Only the owner or authorized users of the bucket can read and write the object.	oss.ACLPrivate
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	oss.ACLPublicRead
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	oss.ACLPublicReadWrite

The following code provides an example on how to configure the ACL of a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Set the ACL of the bucket to public read.
    err = client.SetBucketACL("<yourBucketName>", oss.ACLPublicRead)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Query the ACL of a bucket

The following code provides an example on how to query the ACL of a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Query the ACL of the bucket.
    aclRes, err := client.GetBucketACL("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Bucket ACL:", aclRes.ACL)
}
```

4.5.5.4. Delete buckets

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. This topic describes how to delete a bucket.

Before you delete a bucket, ensure that all objects, LiveChannels, and parts that are generated from multipart upload in the bucket are deleted. The following code provides an example on how to delete a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Delete the bucket.
    err = client.DeleteBucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.5.5. Manage lifecycle rules

Object Storage Service (OSS) allows you to configure lifecycle rules to delete expired objects and parts or convert the storage class of expired objects to Infrequent Access (IA) or Archive to reduce your storage costs. This topic describes how to manage lifecycle rules for buckets.

Background

Each lifecycle rule contains the following information:

- Policy: specifies the mode to match objects and parts.
  - Match by prefix: matches objects and parts by prefix. You can create multiple rules to configure different prefixes. Each prefix must be unique.
  - Match by bucket: matches all objects and parts contained in the bucket. You can create only one rule if you select this method.
- Object lifecycle policy: specifies the validity period or expiration data and the operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired objects are deleted.
  - Expiration date: specifies a date. Objects that are last modified before this date expire and are deleted.
- Part lifecycle policy: specifies the validity period or expiration time and the delete operation to perform on these objects after they expire.
  - Validity period: specifies the number of days within which parts can be retained after they are last modified. After the validity period, expired parts are deleted.
  - Expiration date: specifies a date. Parts that are last modified before this date expire and are deleted.

Lifecycle rules also apply to the parts uploaded by using uploadPart. In this case, the last modified time of an object is the time the multipart upload task is initiated.

Configure lifecycle rules

The following code provides an example on how to configure lifecycle rules:

```
package main
import (
    "fmt"
     "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // rule1 (id:"rule1", enable:true, prefix:"foo/", expiry:Days 3) indicates that objects whose names contain prefix foo/ expire three days after t
hey are last modified.
    rule1 := oss.BuildLifecycleRuleByDays("rule1", "foo/", true, 3)
    rules := []oss.LifecycleRule{rule1}
    // Configure the lifecycle rules.
    bucketName := "<yourBucketName>"
    err = client.SetBucketLifecycle(bucketName, rules)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

View lifecycle rules

The following code provides an example on how to query the lifecycle rules configured for the bucket named examplebucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // View the lifecycle rules.
    lcRes, err := client.GetBucketLifecycle(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Lifecycle Rules:", lcRes.Rules)
}
```

Delete lifecycle rules

The following code provides an example on how to delete lifecycle rules configured for the bucket named examplebucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Delete the lifecycle rules.
    err = client.DeleteBucketLifecycle(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
```

4.5.5.6. Hotlink protection

OSS provides hotlink protection to prevent unauthorized domain names from accessing your OSS data. You can configure the Referer field in the HTTP header.

To configure hotlink protection, you must configure the following parameters:

- Referer Whitelist: Only specified domain names are allowed to access OSS resources.
- Allow Empty Referer: If you do not allow empty Refer fields, a request is allowed to access OSS resources only if the request includes the Referer field configured in the HTTP or HTTPS header.

Configure hotlink protection

Run the following code to configure the referer whitelist:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Add a Referer whitelist. Specify that an empty Referer field is not allowed. You can use an asterisk (*) or a question mark (?) as a wildcard
to set the Referer parameter.
    referers := []string{"http://www.aliyun.com",
                         "http://www.???.aliyuncs.com",
                         "http://www. *.com"}
    err = client.SetBucketReferer(bucketName, referers, false)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Query hotlink protection configurations

Run the following code to obtain a referer whiltelist:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Query hotlink protection information.
    refRes, err := client.GetBucketReferer(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Referers: ", refRes.RefererList)
    fmt.Println("AllowEmptyReferer: ", refRes.AllowEmptyReferer)
}
```

Delete hotlink protection configurations

Run the following code to clear a referer whitelist:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Delete hotlink protection configurations. // You cannot delete hotlink protection configurations directly. To delete hotlink protection config
urations, you must create a rule that allows an empty Referer field to replace the existing rule.
    err = client.SetBucketReferer(bucketName, []string{}, true)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.5.7. Configure logging

You can enable logging to record access to a bucket in log objects. The log objects are stored in a specified bucket.

The log file format is as follows:

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

For the complete code of logging, visit GitHub.

Enable logging

Run the following code to enable bucket access logging:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    targetBucketName := "<yourTargetBucketName>"
    targetPrefix := "<yourTargetPrefix>"
    // Enable logging for the bucket. targetBucketName specifies the bucket that stores log objects. targetPrefix specifies the prefix of the stored
log objects. The log objects are stored in the targetPrefix folder.
    err = client.SetBucketLogging(bucketName, targetBucketName, targetPrefix, true)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

View logging configurations

Run the following code to view the access logging configurations for a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // View logging configurations of the bucket.
    logRes, err := client.GetBucketLogging(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Target Bucket: ", logRes.LoggingEnabled.TargetBucket)
    fmt.Println("Target Prefix: ", logRes.LoggingEnabled.TargetPrefix)
}
```

Disable logging

Run the following code to disable access logging for a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    // client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Disable logging.
    err = client.DeleteBucketLogging(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.5.8. Static website hosting

You can set the static website hosting mode for your bucket. After the configurations take effect, you can use the bucket domain to access this static website and be redirected to a specified index page or error page.

Configure static website hosting

Run the following code to configure static website hosting:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Configure static web hosting. Set the index page to index.html and the error page to error.html.
    err = client.SetBucketWebsite(bucketName, "index.html", "error.html")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

View static website hosting configurations

Run the following code to view static website hosting configurations:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // View the static website hosting configurations.
    wsRes, err := client.GetBucketWebsite(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("indexWebsite: ", wsRes.IndexDocument.Suffix)
    fmt.Println("errorWebsite: ", wsRes.ErrorDocument.Key)
}
```

Delete static website hosting configurations

Run the following code to delete static website hosting configurations:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Delete the static website hosting configurations.
    err = client.DeleteBucketWebsite(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.5.9. CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to different regions. OSS provides CORS operations to facilitate cross-origin access control.

For the complete CORS rule code, visit GitHub.

Configure CORS rules

Run the following code to configure CORS rules for the specified bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    rule1 := oss.CORSRule{
        AllowedOrigin: []string{"*"},
        AllowedMethod: []string{"PUT", "GET"},
        AllowedHeader: []string{},
        ExposeHeader:  []string{},
        MaxAgeSeconds: 200,
    }
    rule2 := oss.CORSRule{
        AllowedOrigin: []string{"http://www.a.com", "http://www.b.com"},
        AllowedMethod: []string{"POST"},
        AllowedHeader: []string{"Authorization"},
        ExposeHeader:  []string{"x-oss-test", "x-oss-test1"},
        MaxAgeSeconds: 100,
    }
    // Configure the CORS rules.
    err = client.SetBucketCORS(bucketName, []oss.CORSRule{rule1, rule2})
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Query CORS rules

Run the following code to obtain CORS rules:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Query the CORS rules.
    corsRes, err := client.GetBucketCORS(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Bucket CORS:", corsRes.CORSRules)
}
```

Delete CORS rules

Run the following code to delete all CORS rules for the specified bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Delete the CORS rules.
    err = client.DeleteBucketCORS(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.6. Upload objects

4.5.6.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for PHP provides the following object upload methods:

You can upload objects to OSS by using one of the following methods:

- Simple upload: uploads an object up to 5 GB in size.
- Append upload: uploads an object up to 5 GB in size.
- Resumable upload: supports concurrent upload and resumable upload. You can define the size of each part. This method is suitable for the upload of large objects. You can use this method to upload an object up to 48.8 TB in size.
- Multipart upload: uploads an object up to 48.8 TB in size. This method is suitable for the upload of large objects.

During object upload, you can configure object metadata and view upload progress by using progress bars. For more information, see Use a progress bar.

4.5.6.2. Simple upload

This topic describes how to use simple upload.

For the complete code used to perform simple upload, visit GitHub.

Upload a string

The following code provides an example on how to upload a string to the exampleobject.txt object in the exampledir directory of the examplebucket bucket:

```
package main
import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Set the storage class to Standard. The default storage class is Standard.
    storageType := oss.ObjectStorageClass(oss.StorageStandard)
    // Set the object ACL to public read. By default, the object inherits the bucket ACL.
    objectAcl := oss.ObjectACL(oss.ACLPublicRead)
    // Upload the string.
    err = bucket.PutObject("<yourObjectName>", strings.NewReader("yourObjectValue"), storageType, objectAcl)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Upload a byte array

The following code provides an example on how to upload a byte array:

```
package main
import (
    "fmt"
    "os"
    "bytes"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Upload the byte array.
    err = bucket.PutObject("<yourObjectName>", bytes.NewReader([]byte("yourObjectValueByteArrary")))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Upload a file stream

The following code provides an example on how to upload a file stream:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Read the local file.
    fd, err := os.Open("<yourLocalFile>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    defer fd.Close()
    // Upload the file stream.
    err = bucket.PutObject("<yourObjectName>", fd)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Upload a local file

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Upload the local file.
    err = bucket.PutObjectFromFile("<yourObjectName>", "<yourLocalFile>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.6.3. Append upload

This topic describes how to perform append upload.

You cannot perform the copyObject operation on append objects. If the object to append does not exist, you can call `AppendObject` to create an append object. If the object to append exists, you can call `AppendObject` to append content to the end of the object.

Run the following code to upload a file with append upload:

```
package main
import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    var nextPos int64 = 0
    // If the object is appended for the first time, the append position is 0. The returned value is the position for the next append. The position t
o start the next append is the total length of appended bytes and the append object.
    nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectAppendValue1"), nextPos)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Start the second append.
    nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectAppendValue2"), nextPos)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // You can append content to an object multiple times.
}
```

You can specify object metadata when you perform the first append from the position of 0. Object metadata cannot be specified during subsequent append operations.

```
    // Specify the metadata of a new object when you perform the first append.
    nextPos, err = bucket.AppendObject("<yourObjectName>", strings.NewReader("YourObjectValue"), 0, oss.Meta("MyProp", "MyPropVal"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
```

4.5.6.4. Resumable upload

If the system stops responding during multipart upload, you can resume the upload by using the ListMultipartUploads and ListParts operations to retrieve all multipart upload tasks on an object and list the uploaded parts in each task. This method allows an upload task to be resumed from the last uploaded part.

The progress of the current upload is recorded in the checkpoint file during the upload process. If the upload of a part fails during the process, the next upload attempt starts from the recorded position in the checkpoint file. The checkpoint file is deleted after the resumable upload task is complete.

Note

- The SDK records the upload progress in the checkpoint file. Make sure that you have write permissions on the checkpoint file. The checkpoint file contains a checksum. This file cannot be edited. If the checkpoint file is damaged, you must upload all parts again.
- If the object being uploaded is modified, you must upload the entire object again.

You can use `Bucket.UploadFile` to perform resumable upload. The following table describes the parameters of this operation.

Parameter	Description
objectKey	The name of the object to upload to OSS.
filePath	The local file path from which to upload the object.
partSize	The size of each part. Valid values: 100 KB to 5 GB.
options	The optional parameters, including: • Routines: the number of concurrent upload threads. The default value is 1, indicating that only one thread is used to upload the parts. • Checkpoint: specifies whether to enable resumable upload and the path of the checkpoint file. By default, resumable upload is disabled. For example, `oss.Checkpoint(true, "")` indicates that resumable upload is enabled and the checkpoint file is `file.cp` contained in the same directory as the source local file. `file` is a variable that indicates the local file name. You can also use `oss.Checkpoint(true, "your-cp-file.cp")` to specify the checkpoint file. • For other object metadata, see Manage object metadata.

The following code provides an example on how to perform resumable upload by using multipart upload:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // The part size is 100 KB. Start resumable upload. Upload the parts by using three concurrent threads.
    // <yourObjectName> is equivalent to objectKey and indicates the complete path of the object that you want to upload to OSS. The path must includ
e the extension of the object name. Example: abc/efg/123.jpg.
    // LocalFile is the filePath value. 100*1024 is the partSize value.
    err = bucket.UploadFile("<yourObjectName>", "LocalFile", 100*1024, oss.Routines(3), oss.Checkpoint(true, ""))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.6.5. Multipart upload

OSS allows you to use multipart upload to split an object into several parts and upload them separately. After all parts are uploaded, you can combine the uploaded parts into a complete object to complete the upload.

To implement multipart upload, perform the following operations:

The following code provides an example on how to perform multipart upload:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    locaFilename := "<yourLocalFilename>"
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    chunks, err := oss.SplitFileByPartNum(locaFilename, 3)
    fd, err := os.Open(locaFilename)
    defer fd.Close()
    // Set the storage class to Standard. The default storage class is Standard.
    storageType := oss.ObjectStorageClass(oss.StorageStandard)
    // Set the storage class to Archive.
    // storageType := oss.ObjectStorageClass(oss.StorageArchive)
    // Step 1: Initiate a multipart upload task. Set the storage class to Standard.
    imur, err := bucket.InitiateMultipartUpload(objectName, storageType)
    // Step 2: Upload the parts.
    var parts []oss.UploadPart
    for _, chunk := range chunks {
        fd.Seek(chunk.Offset, os.SEEK_SET)
        // Call the UploadPart method to upload each part.
        part, err := bucket.UploadPart(imur, fd, chunk.Size, chunk.Number)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        parts = append(parts, part)
    }
    // Set the object ACL to public read. By default, the object inherits the bucket ACL.
    objectAcl := oss.ObjectACL(oss.ACLPublicRead)
    // Step 3: Complete the multipart upload task. Set the object ACL to public read.
    cmur, err := bucket.CompleteMultipartUpload(imur, parts, objectAcl)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("cmur:", cmur)
}
```

Cancel a multipart upload task

You can call the bucket.AbortMultipartUpload method to cancel a multipart upload task. If you cancel a multipart upload task, you cannot use the upload ID to upload any parts. The uploaded parts are deleted.

The following code provides an example on how to cancel a multipart upload task:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Initiate a multipart upload task.
    imur, err := bucket.InitiateMultipartUpload("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Cancel the multipart upload task.
    err = bucket.AbortMultipartUpload(imur)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

List uploaded parts

The following code provides an example on how to list the parts that are uploaded during a specified multipart upload task:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    locaFilename := "<yourLocalFilename>"
    uploadID := "<yourUploadID>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Split the local file. In this example, the local file is split into three parts.
    chunks, err := oss.SplitFileByPartNum(locaFilename, 3)
    fd, err := os.Open(locaFilename)
    defer fd.Close()
    // Initiate a multipart upload task.
    imur, err := bucket.InitiateMultipartUpload(objectName)
    uploadID = imur.UploadID
    fmt.Println("InitiateMultipartUpload UploadID: ", uploadID)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Upload the parts.
    var parts []oss.UploadPart
    for _, chunk := range chunks {
        fd.Seek(chunk.Offset, os.SEEK_SET)
        // Call the UploadPart method to upload each part.
        part, err := bucket.UploadPart(imur, fd, chunk.Size, chunk.Number)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        fmt.Println("UploadPartNumber: ", part.PartNumber, ", ETag: ", part.ETag)
        parts = append(parts, part)
    }
    // List the uploaded parts based on the result obtained by using the InitiateMultipartUploadResult operation.
    lsRes, err := bucket.ListUploadedParts(imur)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the uploaded parts.
    fmt.Println("\nParts:", lsRes.UploadedParts)
    for _, upload := range lsRes.UploadedParts {
        fmt.Println("List PartNumber:  ", upload.PartNumber, ", ETag: " ,upload.ETag, ", LastModified: ", upload.LastModified)
    }
    // List all the uploaded parts based on the result that is obtained by using the InitiateMultipartUploadResult operation. The result is generated
based on the specified object name and upload ID. In this case, the object name and upload ID are known.
    var imur_with_uploadid oss.InitiateMultipartUploadResult
    imur_with_uploadid.Key = objectName
    imur_with_uploadid.UploadID = uploadID
    // List all the uploaded parts.
    lsRes, err = bucket.ListUploadedParts(imur_with_uploadid)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the uploaded parts.
    fmt.Println("\nListUploadedParts by UploadID: ", uploadID)
    for _, upload := range lsRes.UploadedParts {
        fmt.Println("List PartNumber:  ", upload.PartNumber, ", ETag: " ,upload.ETag, ", LastModified: ", upload.LastModified)
    }
    // Complete the multipart upload task.
    cmur, err := bucket.CompleteMultipartUpload(imur, parts)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("cmur:", cmur)
}
```

List multipart upload tasks

You can call the `Bucket.ListMultipartUploads` method to list all ongoing multipart upload tasks that are initiated but not completed or that are canceled. The following table describes the parameters you can configure.

Parameter	Description
Delimiter	The character used to group objects by name. Objects whose names contain the same string from the prefix and the next occurrence of the delimiter are grouped as a single result element in CommonPrefixes.
MaxUploads	The maximum number of multipart upload tasks to list this time. The default value is 1000. The maximum value is 1000.
KeyMarker	Specifies the key-marker value from which to start the list. Multipart upload tasks are listed in alphabetical order of their object names. You can use this parameter with the UploadIDMarker parameter to specify the start position to list the returned results.
Prefix	Specifies the prefix that returned object names must contain. Note that if you use a prefix for a query, the returned object names contain the prefix.

UploadIDMarker	The start position from which to list the returned results. This parameter is used together with the KeyMarker parameter.
	• This parameter is ignored if the KeyMarker parameter is not set.
	• If the KeyMarker parameter is set, the response includes the following tasks:
	  ◦ Specifies the key-marker value from which to start the list. Multipart upload tasks are listed in alphabetical order of their object names.
	  ◦ All multipart upload tasks in which the object names come after the KeyMarker value and the upload IDs are greater than the UploadIDMarker value.

- List multipart upload tasks by using default parameter values

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket to which to upload the object.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List all multipart upload tasks.
    keyMarker := ""
    uploadIdMarker := ""
    for {
        // By default, up to 1,000 records are listed each time.
        lsRes, err := bucket.ListMultipartUploads(oss.KeyMarker(keyMarker), oss.UploadIDMarker(uploadIdMarker))
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        // Display all multipart upload tasks.
        for _, upload := range lsRes.Uploads {
            fmt.Println("Upload: ", upload.Key, ", UploadID: ",upload.UploadID)
        }
        if lsRes.IsTruncated {
            keyMarker = lsRes.NextKeyMarker
            uploadIdMarker = lsRes.NextUploadIDMarker
        } else {
            break
        }
    }
}
```

- List multipart upload tasks by specifying a prefix

```
    lsRes, err := bucket.ListMultipartUploads(oss.Prefix("<yourObjectNamePrefix>"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Uploads:", lsRes.Uploads)
```

- List up to 100 multipart upload tasks

```
    lsRes, err := bucket.ListMultipartUploads(oss.MaxUploads(100))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Uploads:", lsRes.Uploads)
```

- List multipart upload tasks by specifying a prefix and the maximum number of records to returned

```
    lsRes, err := bucket.ListMultipartUploads(oss.Prefix("<yourObjectNamePrefix>"), oss.MaxUploads(100))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Uploads:", lsRes.Uploads)
```

4.5.6.6. Use a progress bar

Progress bars are used to indicate the progress of an upload or download.

The following code provides an example on how to use a progress bar when you call Bucket.PutObjectFromFile to upload an object:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
// Define a progress bar listener.
type OssProgressListener struct {
}
// Define the progress event handler.
func (listener *OssProgressListener) ProgressChanged(event *oss.ProgressEvent) {
    switch event.EventType {
    case oss.TransferStartedEvent:
        fmt.Printf("Transfer Started, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    case oss.TransferDataEvent:
        fmt.Printf("\rTransfer Data, ConsumedBytes: %d, TotalBytes %d, %d%%.",
            event.ConsumedBytes, event.TotalBytes, event.ConsumedBytes*100/event.TotalBytes)
    case oss.TransferCompletedEvent:
        fmt.Printf("\nTransfer Completed, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    case oss.TransferFailedEvent:
        fmt.Printf("\nTransfer Failed, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    default:
    }
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localFile := "<yourLocalFile>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // View the upload progress information while you upload an object.
    err = bucket.PutObjectFromFile(objectName, localFile, oss.Progress(&OssProgressListener{}))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.7. Download objects

4.5.7.1. Overview

Objects are the basic unit for data operations in OSS. OSS SDK for Go provides a variety of object download methods.

For the complete code that is used to download objects, visit GitHub.

You can use one of the following methods to download objects:

- Streaming download

- Range download
- Download to local files
- Resumable download
- Conditional download

You can view the download progress by using progress bars. For more information, see Use a progress bar.

4.5.7.2. Streaming download

This topic describes how to use streaming download.

Download an object to a stream

The following code provides an example on how to download an object to a stream:

```
package main
import (
    "fmt"
    "os"
    "io/ioutil"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Download the object to a stream.
    body, err := bucket.GetObject("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // If you do not close the connection after it is used, connection leaks may occur. Consequently, no connections are available and the program ca
nnot run properly.
    defer body.Close()
    data, err := ioutil.ReadAll(body)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("data:", string(data))
}
```

Download an object to the buffer

The following code provides an example on how to download an object to the buffer:

```
package main
import (
    "fmt"
    "os"
    "io"
    "bytes"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Download the object to the buffer.
    body, err := bucket.GetObject("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    defer body.Close()
    buf := new(bytes.Buffer)
    io.Copy(buf, body)
    fmt.Println("buf:", buf)
}
```

Download an object to a local file stream

The following code provides an example on how to download a specified object to a local file stream:

```
package main
import (
    "fmt"
    "os"
    "io"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Download the object to a local file stream.
    body, err := bucket.GetObject("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    defer body.Close()
    fd, err := os.OpenFile("LocalFile", os.O_WRONLY|os.O_CREATE, 0660)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    defer fd.Close()
    io.Copy(fd, body)
}
```

4.5.7.3. Range download

If you need only part of the data in an object, you can use range download to download data within the specified range.

Specify a valid range

The following code provides an example on how to specify a valid range to download data:

```
package main
import (
  "fmt"
  "os"
  "io/ioutil"
  "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
  // Create an OSSClient instance.
  client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
  if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
  }
  // Specify the name of the bucket.
  bucket, err := client.Bucket("<yourBucketName>")
  if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
  }
  // Query the data that is within the range from byte 15 to byte 35, which includes a total of 21 bytes.
  If the specified range is invalid, for example, the specified range includes a negative number, or the specified value is larger than the object si
ze, the entire object is downloaded.
  body, err := bucket.GetObject("<yourObjectName>", oss.Range(15, 35))
  if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
  }
  // If you do not close the connection after it is used, connection leaks may occur. Consequently, no connections are available and the program cann
ot run properly.
  defer body.Close()
  data, err := ioutil.ReadAll(body)
  if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
  }
  fmt.Println("data:", string(data))
}
```

Specify an invalid range

The valid range for an object whose size is 1,000 bytes is from byte 0 to byte 999. If the specified range is not within the range, the range does not take effect. Then, OSS returns the 200 HTTP status code and the data of the entire object. The following list provides examples of invalid requests and the returned results:

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns the 200 HTTP status code and the data of the entire object.

Compatible download

If you add x-oss-range-behavior:standard to the request header, the download behavior is modified when the specified range is not within the valid range. Example: Use range download to download an object whose size is 1,000 bytes.

- If you set Range: bytes to 500-2000, the value at the end of the range is invalid. Then, OSS returns HTTP status code 206 and the data that is within the range from byte 500 to byte 999.
- If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. Then, OSS returns HTTP status code 416 and error code InvalidRange.

The following code provides an example on how to specify standard behaviors to download data by range:

```
package main
import (
    "fmt"
    "io/ioutil"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Upload 1,000 bytes.
    strContent := ""
    for i := 0; i < 100; i++ {
        strContent += "abcdefghij"
    }
    fmt.Printf("content len:%d\n", len(strContent))
    // Upload a string.
    err = bucket.PutObject("<yourObjectName>", strings.NewReader(strContent))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // If you set Range: bytes to 500-2000, the value at the end of the range is invalid. OSS returns the 206 HTTP status code and the data that is w
ithin the range from byte 500 to byte 999.
    // If you set Range: bytes to 1000-2000, the value at the start of the range is invalid. OSS returns the 416 HTTP status code and the error code
InvalidRange.
    body, err := bucket.GetObject("<yourObjectName>", oss.Range(500, 2000), oss.RangeBehavior("standard"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // If you do not close the connection after it is used, connection leaks may occur. Consequently, no connections are available and the program ca
nnot run properly.
    defer body.Close()
    data, err := ioutil.ReadAll(body)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    if len(data) ! = 500 {
        fmt.Println("read data error, len:%d", len(data))
        os.Exit(-1)
    }
    fmt.Println("data:", string(data))
}
```

4.5.7.4. Download to local files

This topic describes how to download an object to a local file.

The following code provides an example on how to download a specified object to a local file:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Download the object to the local file.
    err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.7.5. Resumable download

You may fail to download a large object if the network is unstable or if the program stops responding. In some cases, you may still fail to download the object even after multiple attempts. To solve this problem, OSS provides the resumable download feature.

To enable resumable download, you need to split the object you want to download into multiple parts and download them separately. After you have downloaded all parts, these parts will be combined into a complete object.

During the download, the download progress is recorded in a checkpoint file. If the download of a part fails, the next download starts from the position recorded in the checkpoint file. The checkpoint file is deleted after resumable download is complete.

Note

- The SDK records the download progress in the checkpoint file. Ensure that you have write permissions on the checkpoint file. The checkpoint file contains a checksum. This file cannot be edited. If the checkpoint file is damaged, you must download the entire object again.
- If the object being downloaded is modified (the ETag value of the object changes) or some parts are missing or modified, you must download the entire object again.

You can use `Bucket.DownloadFile` to download an object by using resumable download. The following table describes the parameters of this operation.

Parameter	Description
objectKey	The name of the object to download.
filePath	The local file path to which to download the object.
partSize	The size of each part. The part size ranges from 1 byte to 5 GB.
options	The optional parameters, including: • Routines: the number of concurrent download threads. The default value is 1, indicating that only one thread is used to download parts. • Checkpoint: specifies whether resumable download is enabled and the path of the checkpoint file. By default, resumable download is disabled. For example, `oss.Checkpoint(true, "")` indicates that resumable download is enabled and the checkpoint file is the `file.cp` file contained in the same directory as the destination local file. `file` is a variable that indicates the local file name. You can also use `oss.Checkpoint(true, "your-cp-file.cp")` to specify the checkpoint file. • Conditions: conditions for multipart download. For more information about the conditions, see Manage object metadata.

Run the following code for resumable download:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Start resumable download. Download the parts by using three threads concurrently.
    // yourObjectName is equivalent to objectKey and indicates the complete path of the object that you want to download from OSS. The path must incl
ude the extension of the object name. Example: abc/efg/123.jpg.
    // LocalFile is the filePath value. 100*1024 is the partSize value.
    err = bucket.DownloadFile("<yourObjectName>", "LocalFile", 100*1024, oss.Routines(3), oss.Checkpoint(true, ""))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.7.6. Conditional download

OSS allows you to configure one or more conditions for object downloads. If the specified conditions are met, the object can be downloaded. If the specified conditions are not met, an error is returned and the object cannot be downloaded.

The following table describes the download conditions.

Parameter	Description	Configuration method
IfModifiedSince	If the specified time is earlier than the time the object is modified, the object can be downloaded. Otherwise, error code 304 Not modified is returned.	oss.IfModifiedSince
IfUnmodifiedSince	If the specified time is later than or equal to the time the object is modified, the object can be downloaded. Otherwise, error code 412 Precondition failed is returned.	oss.IfUnmodifiedSince
IfMatch	If the specified ETag matches that of an object, the object can be downloaded. Otherwise, error code 412 Precondition failed is returned.	oss.IfMatch
IfNoneMatch	If the specified ETag does not match that of an object, the object can be downloaded. Otherwise, error code 304 Not modified is returned.	oss.IfNoneMatch

You can use If-Modified-Since and If-Unmodified-Since as object download conditions at the same time. You can use If-Match and If-None-Match as object download conditions at the same time. To obtain the ETag value of an object, use the Bucket.GetObjectDetailedMeta method.

You can use Bucket.GetObjectDetailedMeta to Obtain the ETag.

```
package main
import (
    "fmt"
    "os"
    "time"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Configure the download conditions.
    date := time.Date(2015, time.November, 10, 23, 0, 0, 0, time.UTC)
    // The object is not downloaded if it does not meet the specified conditions.
    err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile", oss.IfModifiedSince(date))
    if err == nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // The object is downloaded if it meets the specified conditions.
    err = bucket.GetObjectToFile("<yourObjectName>", "LocalFile", oss.IfUnmodifiedSince(date))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

4.5.7.7. Use a progress bar

Progress bars are used to indicate the progress of an upload or download.

The following code provides an example on how to use a progress bar when you use Bucket.GetObjectToFile to download an object:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
// Define the progress bar listener.
type OssProgressListener struct {
}
// Define the progress event handler.
func (listener *OssProgressListener) ProgressChanged(event *oss.ProgressEvent) {
    switch event.EventType {
    case oss.TransferStartedEvent:
        fmt.Printf("Transfer Started, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    case oss.TransferDataEvent:
        fmt.Printf("\rTransfer Data, ConsumedBytes: %d, TotalBytes %d, %d%%.",
            event.ConsumedBytes, event.TotalBytes, event.ConsumedBytes*100/event.TotalBytes)
    case oss.TransferCompletedEvent:
        fmt.Printf("\nTransfer Completed, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    case oss.TransferFailedEvent:
        fmt.Printf("\nTransfer Failed, ConsumedBytes: %d, TotalBytes %d.\n",
            event.ConsumedBytes, event.TotalBytes)
    default:
    }
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localFile := "<yourLocalFile>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // View the download progress while you download an object.
    err = bucket.GetObjectToFile(objectName, localFile, oss.Progress(&OssProgressListener{}))
    if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
    }
    fmt.Println("Transfer Completed.")
}
```

4.5.8. Manage objects

4.5.8.1. Overview

You can manage objects in a bucket by using a variety of operations including SetObjectAcl, GetObjectAcl, ListObjects, DeleteObject, and CopyObject.

By using the preceding operations, you can perform the following operations:

- Determine whether objects exist
- Manage object ACLs
- Manage object metadata
- List objects
- Delete objects
- Copy objects

4.5.8.2. Determine whether an object exists

This topic describes how to determine whether an object exists.

The following code provides an example on how to determine whether an object exists:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket from which you want to delete the objects.
    bucket, err := client.Bucket("<yourBucketName>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Determine whether the object exists.
    isExist, err := bucket.IsObjectExist("<yourObjectName>")
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Exist:", isExist)
}
```

4.5.8.3. Manage object ACLs

This topic describes how to manage the access control list (ACL) of an object.

For the complete code used to manage the ACL of an object, visit GitHub.

The following table describes the ACLs that can be configured for objects.

ACL	Description	Value
Inherited from bucket	The ACL of an object is the same as that of the bucket in which the object is stored.	oss.ACLDefault
Private	Only the owner or authorized users of the bucket can read and write the object.	oss.ACLPrivate
Public read	Only the owner or authorized users of this bucket can write the object. Other users, including anonymous users can only read the object. Exercise caution when you use this ACL.	oss.ACLPublicRead
Public read/write	Any users, including anonymous users can read and write the object. Exercise caution when you use this ACL.	oss.PublicReadWrite

The ACL privileges of objects take precedence over that of buckets. For example, if the ACL of a bucket is private, while the object ACL is public read-write, all users can read and write the object. If an object is not configured with an ACL, its ACL is the same as that of its bucket by default.

The following code provides a complete example on how to configure and query the ACL of an object:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Configure ACL for the object.
    err = bucket.SetObjectACL("<yourObjectName>", oss.ACLPublicReadWrite)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Query the ACL of the object.
    aclRes, err := bucket.GetObjectACL("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object ACL:", aclRes.ACL)
}
```

4.5.8.4. Manage object metadata

This topic describes how to use OSS SDK for Go to configure, modify and query object metadata.

For the complete code used to manage object metadata, visit GitHub.

Configure object metadata

You can configure the metadata of an object when you upload the object. The following table describes the metadata that you can configure.

Parameter	Description
CacheControl	Specifies the caching behavior of a web page when the object is downloaded.
ContentDisposition	Specifies the name of the object when it is downloaded.
ContentEncoding	Specifies the encoding format that is used when the object is downloaded.
Expires	Specifies the expiration time of the cache in GMT.
ServerSideEncryption	Specifies the server-side encryption algorithm to use when OSS creates the object. Valid value: AES256.
ObjectACL	Specifies the ACL of the created object.
Meta	Specifies the user metadata of the object. The parameters of user metadata are prefixed with `X-Oss-Meta-`.

The following code provides an example on how to configure object metadata:

```
package main
import (
    "fmt"
    "os"
    "time"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Configure object metadata: Set the expiration time to 23:00:00, 10 January 2049 in GMT, ACL to public read, and MyProp to MyPropVal as user me
tadata.
    expires := time.Date(2049, time.January, 10, 23, 0, 0, 0, time.UTC)
    options := []oss.Option{
        oss.Expires(expires),
        oss.ObjectACL(oss.ACLPublicRead),
        oss.Meta("MyProp", "MyPropVal"),
    }
    // Use a data stream to upload the object.
    err = bucket.PutObject("<yourObjectName>", strings.NewReader("MyObjectValue"), options...)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Query the object metadata.
    props, err := bucket.GetObjectDetailedMeta("<yourObjectName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

Modify object metadata

The following code provides an example on how to modify a single or multiple object metadata parameters at a time:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Modify a single object metadata parameter at a time.
    err = bucket.SetObjectMeta(objectName, oss.Meta("MyMeta", "MyMetaValue1"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Modify multiple object metadata parameters at a time.
    options := []oss.Option{
        oss.Meta("MyMeta", "MyMetaValue2"),
        oss.Meta("MyObjectLocation", "HangZhou"),
    }
    err = bucket.SetObjectMeta(objectName, options...)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Query the object metadata.
    props, err := bucket.GetObjectDetailedMeta(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

Obtain object metadata

The following code provides an example on how to obtain object metadata:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Query the object metadata.
    props, err := bucket.GetObjectDetailedMeta(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Object Meta:", props)
}
```

4.5.8.5. List objects

This topic describes how to list the objects that are contained in a bucket.

Objects in a bucket are sorted alphabetically. You can use `Bucket.ListObjects` to list the objects in a bucket. The following table describes the parameters for Bucket.ListObjects.

Parameter	Description
delimiter	Groups objects by name. Objects whose names contain the same string from the prefix and the next occurrence of the delimiter are grouped as a single result element in commonPrefixes.
prefix	Specifies the prefix that returned object names must contain.
maxKeys	Specifies the maximum number of objects that can be listed each time. The default value is 100. The maximum value is 1000.
marker	Specifies the name of the object after which the list begins.

For the complete code that is used to list objects, visit GitHub.

Simple list

You can use the default parameters to query the list of objects in a bucket. By default, a maximum of 100 objects are listed. The following code provides an example on how to perform simple list:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        HandleError(err)
    }
    // Specify the name of the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // List all objects.
    marker := ""
    for {
        lsRes, err := bucket.ListObjects(oss.Marker(marker))
        if err ! = nil {
            HandleError(err)
        }
        // Display the listed objects. By default, a maximum of 100 records are returned each time.
        for _, object := range lsRes.Objects {
            fmt.Println("Bucket: ", object.Key)
        }
        if lsRes.IsTruncated {
            marker = lsRes.NextMarker
        } else {
            break
        }
    }
}
```

List a specified number of objects

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Set the maximum number of objects to list and list the objects.
    lsRes, err := bucket.ListObjects(oss.MaxKeys(200))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the listed objects.
    fmt.Println("Objects:", lsRes.Objects)
    for _, object := range lsRes.Objects {
        fmt.Println("Object:", object.Key)
    }
}
```

List objects whose names contain a specified prefix

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List the objects whose names contain the specified prefix. By default, a maximum of 100 objects are listed.
    lsRes, err := bucket.ListObjects(oss.Prefix("my-object-"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the listed objects.
    fmt.Println("Objects:", lsRes.Objects)
    for _, object := range lsRes.Objects {
        fmt.Println("Object:", object.Key)
    }
}
```

List objects whose names come after a specified marker

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List the objects whose names come after the specified marker. By default, a maximum of 100 objects are listed.
    lsRes, err := bucket.ListObjects(oss.Marker("my-object-xx"))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Display the listed objects.
    fmt.Println("Objects:", lsRes.Objects)
    for _, object := range lsRes.Objects {
        fmt.Println("Object:", object.Key)
    }
}
```

List all objects by page

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List all objects by page, with up to 200 objects on each page.
    marker := oss.Marker("")
    for {
        lsRes, err := bucket.ListObjects(oss.MaxKeys(200), marker)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        marker = oss.Marker(lsRes.NextMarker)
        fmt.Println("Objects:", lsRes.Objects)
        if ! lsRes.IsTruncated {
            break
        }
    }
}
```

List objects whose names come after a specified marker by page

The following code provides an example on how to list the objects that follow a specified marker by page. The maximum number of objects that can be listed on each page is specified by MaxKeys.

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List the objects whose names come after the specified marker by page, with up to 50 objects on each page.
    marker = oss.Marker("my-object-xx")
    for {
        lsRes, err := bucket.ListObjects(oss.MaxKeys(50), marker)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        marker = oss.Marker(lsRes.NextMarker)
        fmt.Println("Objects:", lsRes.Objects)
        if ! lsRes.IsTruncated {
            break
        }
    }
}
```

List objects whose names contain a specified prefix by page

The following code provides an example on how to list the objects whose names contain a specified prefix by page. The maximum number of objects to list on each page is specified by maxKeys.

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List the objects whose names contain the specified prefix by page, with up to 80 objects on each page.
    prefix := oss.Prefix("my-object-")
    marker := oss.Marker("")
    for {
        lsRes, err := bucket.ListObjects(oss.MaxKeys(80), marker, prefix)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        prefix = oss.Prefix(lsRes.Prefix)
        marker = oss.Marker(lsRes.NextMarker)
        fmt.Println("Objects:", lsRes.Objects)
        if ! lsRes.IsTruncated {
            break
        }
    }
}
```

List the information about the objects whose names contain a specified prefix

The following code provides an example on how to list the information about the objects whose names contain a specified prefix. Such information includes the object size, the time when the object was last modified, and the object name.

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Traverse the objects.
    marker := oss.Marker("")
    prefix := oss.Prefix("<yourObjectPrefix>")
    for {
        lor, err := bucket.ListObjects(marker, prefix)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        for _, object := range lor.Objects {
            fmt.Printf("%s %d %s\n", object.LastModified, object.Size, object.Key)
        }
        prefix = oss.Prefix(lor.Prefix)
        marker = oss.Marker(lor.NextMarker)
        if ! lor.IsTruncated {
            break
        }
    }
}
```

List the information about the subfolders whose names contain a specified prefix

The following code provides an example on how to list the information about the subfolders whose names contain a specified prefix:

```
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    marker := oss.Marker("")
    prefix := oss.Prefix("<yourDirPrefix>")
    for {
        lor, err := bucket.ListObjects(marker, prefix, oss.Delimiter("/"))
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        for _, dirName := range lor.CommonPrefixes {
            fmt.Println(dirName)
        }
        prefix = oss.Prefix(lor.Prefix)
        marker = oss.Marker(lor.NextMarker)
        if ! lor.IsTruncated {
            break
        }
    }
}
```

4.5.8.6. Delete objects

This topic describes how to delete objects.

Warning Deleted objects cannot be recovered. Exercise caution when you delete objects.

For the complete code used to delete objects, visit GitHub.

Delete a single object

The following code provides an example on how to delete an object named *exampleobject.txt* from a bucket named examplebucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Delete a single object. objectName indicates the full path that includes the extension of the object. Example: abc/efg/123.jpg.
    // To delete a folder, set objectName to the folder name. If the folder is not empty, you must delete all objects from the folder before you can
delete it.
    err = bucket.DeleteObject(objectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Delete multiple objects

You can call the `Bucket.DeleteObjects` operation to delete multiple objects and specify the `DeleteObjectsQuiet` parameter to specify whether to return results of delete operations. By default, information of deleted objects is returned.

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify that information of deleted objects is returned.
    delRes, err := bucket.DeleteObjects([]string{"my-object-1", "my-object-2"})
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    fmt.Println("Deleted Objects:", delRes.DeletedObjects)
    // Specify that results of delete operations are not returned.
    _, err = bucket.DeleteObjects([]string{"my-object-3", "my-object-4"},
    oss.DeleteObjectsQuiet(true))
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Delete objects whose names contain a specified prefix

The following code provides an example on how to delete objects whose names contain a specified prefix:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify the name of the bucket.
    bucket, err := client.Bucket("<yourBucketName>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // List and delete objects whose names contain the specified prefix.
    marker := oss.Marker("")
    prefix := oss.Prefix("<yourObjectPrefix>")
    count := 0
    for {
        lor, err := bucket.ListObjects(marker, prefix)
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        objects := []string{}
        for _, object := range lor.Objects {
            objects = append(objects, object.Key)
        }
        delRes, err := bucket.DeleteObjects(objects, oss.DeleteObjectsQuiet(true))
        if err ! = nil {
            fmt.Println("Error:", err)
            os.Exit(-1)
        }
        if len(delRes.DeletedObjects) > 0 {
            fmt.Println("these objects deleted failure,", delRes.DeletedObjects)
            os.Exit(-1)
        }
        count += len(objects)
        prefix = oss.Prefix(lor.Prefix)
        marker = oss.Marker(lor.NextMarker)
        if ! lor.IsTruncated {
            break
        }
    }
    fmt.Printf("success,total delete object count:%d\n", count)
}
```

4.5.8.7. Copy objects

This topic describes how to copy objects.

For the complete code that is used to copy objects, visit GitHub.

Copy objects within a bucket

The following code provides an example on how to copy an object within a bucket:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    destObjectName := "<yourDestObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Copy the object to another object within the bucket.
    _, err = bucket.CopyObject(objectName, destObjectName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Copy objects across buckets

You can copy objects from other buckets to the current bucket or copy objects from the current bucket to other buckets. The following code provides an example on how to copy objects across buckets:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    srcBucketName := "<yourSrcBucketName>"
    dstBucketName := "<yourDstBucketName>"
    srcObjectName := "<yourSrcObjectName>"
    dstObjectName := "<yourDstObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Copy srcObjectName from srcBucketName to the current bucket.
    bucket.CopyObjectFrom(srcBucketName, srcObjectName, dstObjectName)
    if err ! = nil {
        fmt.Println("CopyObjectFrom Error:", err)
        os.Exit(-1)
    }
    // Copy srcObjectName from the current bucket to dstBucketName.
    bucket.CopyObjectTo(dstBucketName, dstObjectName, srcObjectName)
    if err ! = nil {
        fmt.Println("CopyObjectTo Error:", err)
        os.Exit(-1)
    }
}
```

Process object metadata during object copy

You can use MetadataDirective to process object metadata during object copy. Valid values of MetadataDirective:

- oss.MetaCopy: This is the default value. Copy metadata of the source object. The metadata of the destination object is the same as the metadata of the source object.
- oss.MetaReplace: overwrites the source object metadata by using the specified object metadata.

Notice All metadata of the source object is overwritten if you use oss.MetaReplace. The metadata of the source object cannot be recovered. Exercise caution when you perform this operation.

The following table describes the parameter you can configure when you set MetadataDirective to oss.MetaReplace.

Parameter	Description
CacheControl	Specifies the caching behavior of a web page when the destination object is downloaded.
ContentDisposition	Specifies the name of the destination object when it is downloaded.
ContentEncoding	Specifies the encoding format that is used when the destination object is downloaded.
Expires	Specifies the expiration time of the cache in GMT.
ServerSideEncryption	Specifies the server-side encryption algorithm that is used when OSS creates the destination object. Valid value: AES256.
ObjectACL	Specifies the ACL of the destination object.
Meta	Specifies the user metadata of the destination object. The parameters of user metadata are prefixed with `X-Oss-Meta-` .

The following code provides an example on how to process object metadata during object copy:

```
package main
import (
    "fmt"
    "os"
    "time"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    destObjectName := "<yourDestObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // Specify metadata for the destination object.
    expires := time.Date(2049, time.January, 10, 23, 0, 0, 0, time.UTC)
    options := []oss.Option{
        oss.MetadataDirective(oss.MetaReplace),
        oss.Expires(expires),
        oss.ObjectACL(oss.ACLPublicRead),
        oss.Meta("MyMeta", "MyMetaValue")}
    // Overwrite the metadata of the source object by using the specified metadata.
    _, err = bucket.CopyObject(objectName, destObjectName, options...)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
}
```

Conditional copy

OSS allows you to configure one or more conditions for object copy. If the specified conditions are met, the object can be copied. If the specified conditions are not met, an error code is returned and the object cannot be copied. The following table describes the conditions that you can configure in object copy.

Parameter	Description
CopySourceIfMatch	If the ETag of the source object is the same as that of the destination object, the object is copied. Otherwise, an error is returned.
CopySourceIfNoneMatch	If the ETag of the source object is different from that of the destination object, the object is copied. Otherwise, an error is returned.
CopySourceIfModifiedSince	If the specified time is earlier than the actual modified time of the object, the object is copied. Otherwise, an error is returned.
CopySourceIfUnmodifiedSince	If the specified time is equal to or later than the actual modified time of the object, the object is copied. Otherwise, an error is returned.

You can set CopySourceIfMatch and CopySourceIfNoneMatch at the same time. You can set CopySourceIfModifiedSince and CopySourceIfUnmodifiedSince at the same time.

The following code provides an example on how to perform conditional copy:

```
package main
import (
    "fmt"
    "os"
    "time"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    // bucketName := "<yourBucketName>"
    // objectName := "<yourObjectName>"
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    destMatchObjectName := "<yourMatchDestObjectName>"
    destUnMatchObjectName := "<yourUnmatchDestObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        fmt.Println("Error:", err)
        os.Exit(-1)
    }
    date := time.Date(2011, time.November, 10, 23, 0, 0, 0, time.UTC)
    // Copy the objects that meet the specified condition.
    _, err = bucket.CopyObject(objectName, destMatchObjectName, oss.CopySourceIfModifiedSince(date))
    if err ! = nil {
        fmt.Println("CopyObject CopySourceIfModifiedSince Error:", err)
    }
    // Do not copy the objects that cannot meet the specified condition.
    _, err = bucket.CopyObject(objectName, destUnMatchObjectName, oss.CopySourceIfUnmodifiedSince(date))
    if err ! = nil {
        fmt.Println("CopyObject CopySourceIfUnmodifiedSince Error:", err)
    }
}
```

4.5.9. Authorized access

This topic describes how to use STS and a signed URL to authorize access.

Use STS to authorize temporary access

Alibaba Cloud Security Token Service (STS) is a web service that provides a temporary access token to a cloud computing user. You can use STS to grant a third-party application or your RAM user an access credential with a customized validity period and permissions.

STS provides the following benefits:

- You need only to generate an access token and send the access token to a third-party application. You do not need to expose your AccessKey pair to the third-party application. You can specify the access permissions and validity period of this token.
- The token automatically expires after the validity period. Therefore, you do not need to manually revoke the access permissions of a token.

The following code provides an example on how to create a signature request by using STS:

```
import "github.com/aliyun/aliyun-oss-go-sdk/oss"
// After you obtain a temporary STS credential for your OSSClient, an OSSClient instance is created based on the security token and temporary AccessK
ey pair (AccessKey ID and AccessKey secret) that are contained in the credential.
// Create an OSSClient instance.
client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>", oss.SecurityToken("<yourSecurityToken>"))
if err ! = nil {
    fmt.Println("Error:", err)
    os.Exit(-1)
// Perform an operation in OSS.
}
```

Use a signed URL to authorize temporary access

For the complete code of using a signed URL to authorize temporary access, visit GitHub.

- Use a signed URL to upload an object

```
package main
import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        HandleError(err)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localFilename := "<yourLocalFilename>"
    // Specify the name of the bucket to which to upload the object.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // Upload the object by using the signed URL.
    signedURL, err := bucket.SignURL(objectName, oss.HTTPPut, 60)
    if err ! = nil {
        HandleError(err)
    }
    var val = "Go with Alibaba Cloud"
    err = bucket.PutObjectWithURL(signedURL, strings.NewReader(val))
    if err ! = nil {
        HandleError(err)
    }
    // Upload the object by using the signed URL with optional parameters configured.
    options := []oss.Option{
        oss.Meta("myprop", "mypropval"),
        oss.ContentType("image/tiff"),
    }
    signedURL, err = bucket.SignURL(objectName, oss.HTTPPut, 60, options...)
    if err ! = nil {
        HandleError(err)
    }
    err = bucket.PutObjectFromFileWithURL(signedURL, localFilename, options...)
    if err ! = nil {
        HandleError(err)
    }
}
```

Note For more information, see Manage object metadata.

- Use a signed URL to download an object

```
package main
import (
    "fmt"
    "os"
    "io/ioutil"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        HandleError(err)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    localDownloadedFilename := "<yourDownloadedFilename>"
    // Specify the name of the bucket from which to download the object.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // Download the object to a stream by using the signed URL.
    signedURL, err := bucket.SignURL(objectName, oss.HTTPGet, 60)
    if err ! = nil {
        HandleError(err)
    }
    body, err := bucket.GetObjectWithURL(signedURL)
    if err ! = nil {
        HandleError(err)
    }
    // Read the object content.
    data, err := ioutil.ReadAll(body)
    body.Close()
    data = data // use data
    // Download the object to a local file by using the signed URL.
    err = bucket.GetObjectToFileWithURL(signedURL, localDownloadedFilename)
    if err ! = nil {
        HandleError(err)
    }
}
```

4.5.10. IMG

OSS Image Processing (IMG) is a secure, cost-effective, and highly reliable image processing service that can process large amounts of data. After you upload source images to OSS, you can call RESTful API operations to process the images anytime, anywhere, and on any Internet device.

Basic features

OSS IMG provides the following features:

- Query image information.
- Convert image formats.
- Resize, crop, and rotate images.
- Apply image effect.
- Add image, text, and text-and-image watermarks to images.
- Customize image processing styles. Log on to the OSS console. Click a bucket name to go to the bucket details page. Click the **Image Processing (IMG)** tab and click **Create Rule** to customize an image processing style.
- Use cascade processing to call multiple IMG features.

Usage

You can use standard `HTTP GET` requests to access IMG. All processing parameters are encoded in the query string of a URL.

- Anonymous access

  You can anonymously access a processed image by using a third-level domain in the following format:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
```

Parameter	Description
bucket	The name of the bucket.
endpoint	The domain name used to access the region.
object	The name of the image object.
image	The identifier reserved by IMG.
action	The operations performed on images, such as scaling, cropping, and rotating.

Parameter	Description
param	The parameters corresponding to the operations performed on the images.

- Basic operations

  For example, you can scale an image to a width of 100 and adjust the height based on the ratio:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- Customize an image style

  You can anonymously access IMG by using a third-level domain in the following format:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
  ```

  - style: the identifier reserved by IMG for the custom style.
  - yourStyleName: the name of the custom style. The name is specified by the Rule Name parameter when you create the custom style in the console.

  Example:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
  ```

- Cascade processing

  You can use cascade processing to perform multiple operations in sequence on an image by using a URL in the following format:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction1>,<yourParamValue1>/<yourAction2>,<yourParamValue2>/...
  ```

  Example:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100/rotate,90
  ```

- Access by using HTTPS

  IMG supports access by using HTTPS. Example:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
  ```

- Authorized access

  If the ACL of an image object is private, you must have permissions on the image object before you can perform operations on the image object.

Bucket ACL	Object ACL
private	default
Any ACL	private

The following code provides an example on how to generate a signed URL to access IMG:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
    HandleError(err)
    }
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
    HandleError(err)
    }
    ossImageName := "<yourImageName>"
    signedURL, err := bucket.SignURL(ossImageName, oss.HTTPGet, 600, oss.Process("image/format,png"))
    if err ! = nil {
    HandleError(err)
    } else {
    fmt.Println(signedURL)
    }
}
```

- Access by using SDKs

  You can use OSS SDKs to access and process image objects of any ACLs.

- Basic operations

  The following code provides an example on how to perform basic IMG operations including obtaining image information, converting image formats, resizing, cropping, rotating, watermarking, and applying effects:

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
    HandleError(err)
    }
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
    HandleError(err)
    }
    // Resize the image.
    sourceImageName := "example.png"
    targetImageName := "example-resize.png"
    style := "image/resize,m_fixed,w_100,h_100"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
    // Crop the image.
    targetImageName = "example-crop.png"
    style = "image/crop,w_100,h_100,x_100,y_100,r_1"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
    // Rotate the image.
    targetImageName = "example-rotate.png"
    style = "image/rotate,90"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
    // Sharpen the image.
    targetImageName = "example-sharpen.png"
    style = "image/sharpen,100"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
    // Add watermarks to the image.
    targetImageName = "example-watermark.png"
    style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
    // Convert the image format.
    targetImageName = "example-formatconvert.jpg"
    style = "image/format,jpg"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
}
```

- Custom image styles

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
    HandleError(err)
    }
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
     HandleError(err)
    }
    // Customize the image style.
    sourceImageName := "example.png"
    targetImageName := "example-custom.png"
    style := "style/<yourCustomStyleName>"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
    HandleError(err)
    }
}
```

- Cascade processing

```
package main
import (
    "fmt"
    "os"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
    HandleError(err)
    }
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
    HandleError(err)
     }
    // Customize the image style.
    sourceImageName := "example.png"
    targetImageName := "example-cascade.png"
    style := "image/resize,m_fixed,w_100,h_100/rotate,90"
    err = bucket.GetObjectToFile(sourceImageName, targetImageName, oss.Process(style))
    if err ! = nil {
     HandleError(err)
    }
}
```

Persistent storage of processed images

```
package main
import (
    "fmt"
    "os"
    "encoding/base64"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
    HandleError(err)
    }
```

```
    // Obtain the bucket.
    bucketName := "<yourBucketName>"
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
     HandleError(err)
    }
    // Scale the image and store the processed image. By default, the processed image is stored in the same bucket as the original image.
    sourceImageName := "example.png"
    targetImageName := "example-resize.png"
    style := "image/resize,m_fixed,w_100,h_100"
    process := fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err := bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
    // Scale the image and store the processed image. The processed image is stored in the different bucket from the original image.
    targetBucketName := "<yourTargetBucketName>"
    targetImageName = "example-crop-different-bucket.png"
    style = "image/resize,m_fixed,w_100,h_100"
    process = fmt.Sprintf("%s|sys/saveas,o_%v,b_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)), base64.URLEncoding.EncodeToSt
ring([]byte(targetBucketName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
    // Crop the image and store the processed image.
    targetImageName = "example-crop.png"
    style = "image/crop,w_100,h_100,x_100,y_100,r_1"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
    // Rotate the image and store the processed image.
    targetImageName = "example-rotate.png"
    style = "image/rotate,90"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
    // Sharpen the image and store the processed image.
    targetImageName = "example-sharpen.png"
    style = "image/sharpen,100"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
       fmt.Println(result)
    }
    // Add watermarks to the image and store the processed image.
    targetImageName = "example-watermark.png"
    style = "image/watermark,text_SGVsbG8g5Zu-54mH5pyN5YqhIQ"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
    // Convert the image format and store the processed image.
    targetImageName = "example-formatconvert.jpg"
    style = "image/format,jpg"
    process = fmt.Sprintf("%s|sys/saveas,o_%v", style, base64.URLEncoding.EncodeToString([]byte(targetImageName)))
    result, err = bucket.ProcessObject(sourceImageName, process)
    if err ! = nil {
    HandleError(err)
    } else {
        fmt.Println(result)
    }
}
```

4.5.11. Error handling

This topic describes how to handle errors when you use OSS SDK for Go.

Client errors

If errors occur when you use OSS SDK for Go, a unified operation used to handle errors is returned. The following code defines the error operation:

```
type error interface {
    Error() string
}
```

Other specific errors are further defined based on this error operation. The following code provides an example of the operation used to handle an HTTP request error:

```
//net.Error
type Error interface {
    error
    Timeout() bool   // Is the error a timeout
    Temporary() bool // Is the error temporary
}
```

Server errors

If request errors occur when you use OSS SDK for Go, corresponding errors are returned. In case of HTTP request errors and I/O errors, errors defined in OSS SDK for Go are returned. If an error occurs when the OSS server processes a request, the following error is returned, which is an implementation of the error operation:

```
type ServiceError struct {
    Code       string    // The error code returned by OSS.
    Message    string    // The error details provided by OSS.
    RequestId  string    // The UUID used to identify the request.
    HostId     string    // The host ID used to identify the accessed OSS cluster.
    StatusCode int       // The HTTP status code.
}
```

If the HTTP status code returned by OSS is not as expected, the following error is returned, which is an implementation of the error operation:

```
type UnexpectedStatusCodeError struct {
    allowed []int    // The expected HTTP status code returned by OSS.
    got     int      // The actual HTTP status code returned by OSS.
}
```

Example

The following code provides an example on how to handle errors:

```
package main
import (
    "fmt"
    "os"
    "strings"
    "github.com/aliyun/aliyun-oss-go-sdk/oss"
)
// The function used to handle the error.
func HandleError(err error) {
    fmt.Println("Error:", err)
    os.Exit(-1)
}
func main() {
    // Create an OSSClient instance.
    client, err := oss.New("<yourEndpoint>", "<yourAccessKeyId>", "<yourAccessKeySecret>")
    if err ! = nil {
        HandleError(err)
    }
    bucketName := "<yourBucketName>"
    objectName := "<yourObjectName>"
    // Specify the name of the bucket.
    bucket, err := client.Bucket(bucketName)
    if err ! = nil {
        HandleError(err)
    }
    // Use simple upload to upload strings.
    err = bucket.PutObject(objectName, strings.NewReader("yourObjectValueContentString"))
    if err ! = nil {
        HandleError(err)
}
```

OSS error codes

The following table describes the OSS error codes.

Error code	Error message	HTTP status code
AccessDenied	The error message returned because access is denied.	403
BucketAlreadyExists	The error message returned because the bucket already exists.	409

Error code	Error message	HTTP status code
BucketNotEmpty	The error message returned because the bucket is not empty.	409
EntityTooLarge	The error message returned because the entity exceeds the maximum allowed size.	400
EntityTooSmall	The error message returned because the entity is smaller than the minimum allowed size.	400
FileGroupTooLarge	The error message returned because the object group exceeds the maximum allowed size.	400
InvalidLinkName	The error message returned because the symbolic link points to an object that has the same name as the symbolic link.	400
LinkPartNotExist	The error message returned because the symbolic link points to an object that does not exist.	400
ObjectLinkTooLarge	The error message returned because the number of objects the symbolic link points to exceeds the limit.	400
FieldItemTooLong	The error message returned because the total length of the form fields in the POST request exceeds the limit.	400
FilePartInterity	The error message returned because the part is modified.	400
FilePartNotExist	The error message returned because the specified part does not exist.	400
FilePartStale	The error message returned because the part expires.	400
IncorrectNumberOfFilesInPOSTRequest	The error message returned because the number of objects in the POST request is invalid.	400
InvalidArgument	The error message returned because the parameter format is invalid.	400
InvalidAccessKeyId	The error message returned because the specified Accesskey ID does not exist.	403
InvalidBucketName	The error message returned because the bucket name is invalid.	400
InvalidDigest	The error message returned because the digest is invalid.	400
InvalidEncryptionAlgorithmError	The error message returned because the specified encryption algorithm based on entropy encoding is invalid.	400
InvalidObjectName	The error message returned because the object name is invalid.	400
InvalidPart	The error message returned because the part is invalid.	400
InvalidPartOrder	The error message returned because the part sequence is invalid.	400
InvalidPolicyDocument	The error message returned because the policy document is invalid.	400
InvalidTargetBucketForLogging	The error message returned because the destination bucket specified for logging is invalid.	400
InternalError	The error message returned when an internal OSS error occurs.	500
MalformedXML	The error message returned because the XML format is invalid.	400
MalformedPOSTRequest	The error message returned because the format of the POST request body is invalid.	400
MaxPOSTPreDataLengthExceededError	The error message returned because the size of the POST request body, excluding the object content to upload, exceeds the limit.	400
MethodNotAllowed	The error message returned because the method is not supported.	405
MissingArgument	The error message returned because the required parameters are not set.	411

Error code	Error message	HTTP status code
MissingContentLength	The error message returned because the content length is not specified.	411
NoSuchBucket	The error message returned because the specified bucket does not exist.	404
NoSuchKey	The error message returned because the specified object does not exist.	404
NoSuchUpload	The error message returned because the specified multipart upload ID does not exist.	404
NotImplemented	The error message returned because the method cannot be implemented.	501
PreconditionFailed	The error message returned because a precondition error occurs.	412
RequestTimeTooSkewed	The error message returned because the difference between the time on the client when the request is sent and the time on the server when the request is received is greater than 15 minutes.	403
RequestTimeout	The error message returned because the request times out.	400
RequestIsNotMultiPartContent	The error message returned because the content-type value specified in the POST request is invalid.	400
SignatureDoesNotMatch	The error message returned because the signature is invalid.	403
TooManyBuckets	The error message returned because the number of the buckets managed by a user exceeds the limit.	400
InvalidEncryptionAlgorithmError	The error message returned because the specified encryption algorithm based on entropy encoding is invalid.	400

Note In the preceding table, the error code is specified by the OssServiceError.Code parameter. The HTTP status code is specified by the OssServiceError.StatusCode parameter.

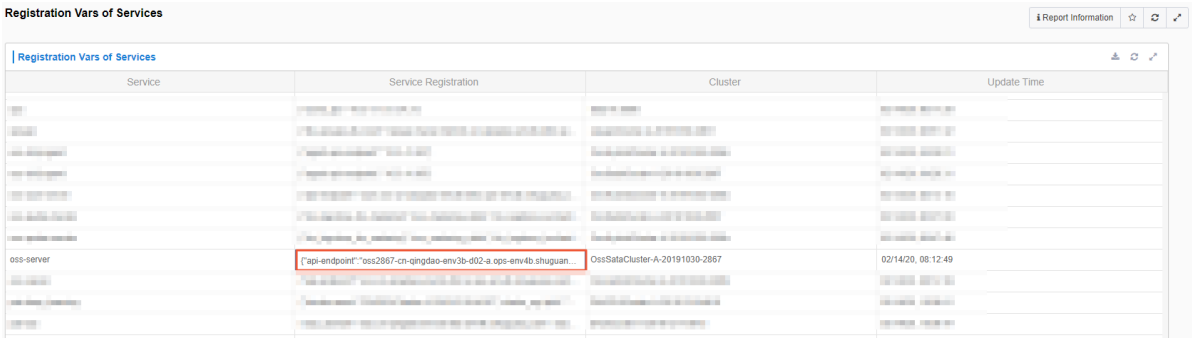
5.Obtain an endpoint

You can obtain an endpoint of a service in the Apsara Uni-manager Operations Console.

Procedure

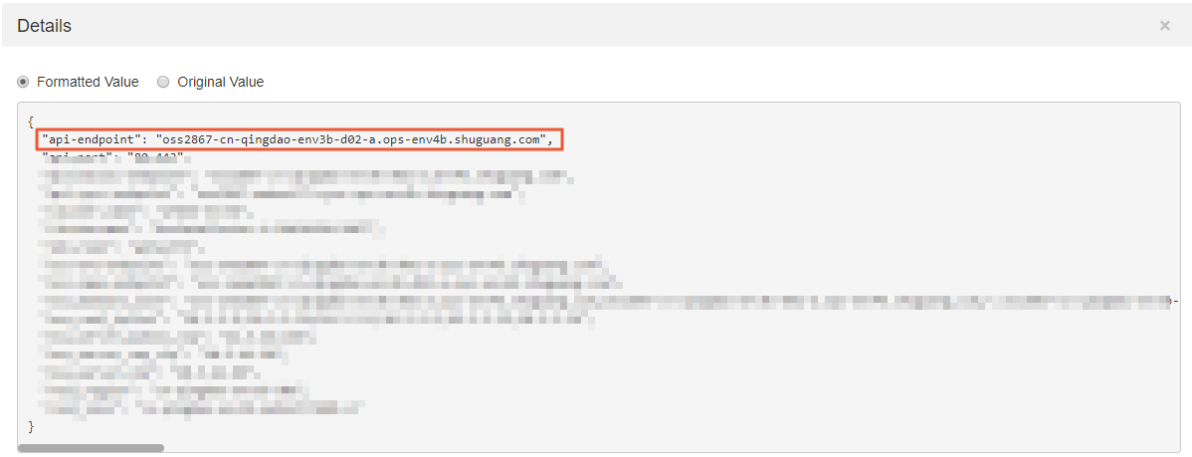
1. Log on to the Apsara Uni-manager Operations Console

   For more information about how to log on to the Apsara Uni-manager Operations console, see *Log on to the Apsara Uni-manager Operations console* in *Operations and Maintenance Guide*.
2. In the top navigation bar, choose **Products > Platforms > Apsara Infrastructure Management Framework**.
3. In the left-side navigation pane, click **Reports**.
4. On the All Reports page, search for **Registration Vars of Services**. In the report list that appears, click the name of the report.
5. On the **Registration Vars of Services** page, find the **oss-server** row. Right-click the **Service Registration** column and choose **Show More** from the shortcut menu.



6. Click **Formatted Value**.

   The value of **api-endpoint** is the OSS endpoint.



6.Obtain an AccessKey pair

An AccessKey pair consists of an AccessKey ID and an AccessKey secret. AccessKey pairs are used to implement symmetric encryption to verify the identity of requesters. The AccessKey ID is used to identify a user. The AccessKey secret is used to encrypt a signature string. This topic describes how to obtain an AccessKey pair.

Prerequisites

Only the operations administrator or level-1 organization administrator can obtain the AccessKey pair of an organization.

Obtain the AccessKey pair of an organization

To obtain the AccessKey pair of an organization, perform the following operations:

1. Log on to the Apsara Uni-manager Management Console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click an organization name.
5. In the Current Organization section, click **Obtain an accesskey**.
6. In the AccessKey message, view the AccessKey pair of the organization.

> **Note** An AccessKey pair is automatically allocated to each level-1 organization. Subordinate organizations use the same AccessKey pair of their level-1 organization.

Obtain the AccessKey pair of a personal account

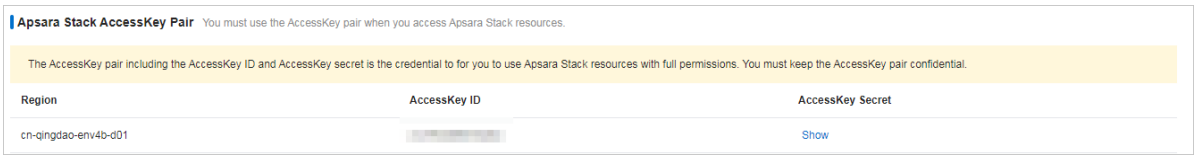
To obtain the AccessKey pair of a personal account, perform the following steps:

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you can use to log on to the console from the operations administrator.

   > **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, which include ! @ # $ %

3. Click **Login**.
4. In the upper-right corner of the homepage, move the pointer over the profile picture and select **User Information** from the shortcut menu.
5. In the **Apsara Stack AccessKey Pair** section, view your AccessKey pair.



> **Note** An AccessKey pair consists of an AccessKey ID and an AccessKey secret. These credentials provide you with full permissions on Apsara Stack resources. You must keep the AccessKey pair confidential.