Alibaba Cloud
Object Storage Service

SDK Reference

Issue: 20181116

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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 **Note:** You can use **Ctrl** + **A** to select all files.
>	Multi-level menu cascade.	**Settings** > **Network** > **Set network type**
Bold	It is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Java

1.1 Installation

This topic describes how to install OSS Java SDK.

Preparation

- Environment requirements

  You must use Java 1.6 or later versions.

- Check the Java version

  Run `java -version` to check Java version.

Download SDK

- *Direct download*
- *Download from GitHub*
- *Download previous versions*

Install SDK

- Method One: Add dependencies to Maven (recommended)

  To use OSS Java SDK in Maven, you only need to add the corresponding dependencies to the pom.xml file. This example uses OSS Java SDK 2.8.3. Add the following content to <dependencies>:

  ```
  <dependency>
      <groupId>com.aliyun.oss</groupId>
      <artifactId>aliyun-sdk-oss</artifactId>
      <version>2.8.3</version>
  </dependency>
  ```

- Method Two: Import jar files into Eclipse

  Take OSS Java SDK 2.8.3 as an example. To import jar files into Eclipse, follow these steps:

  1. Download the *Java SDK*.
  2. Decompress the Java SDK.
  3. Copy the *aliyun-sdk-oss-2.8.3.jar* file and all files from the lib folder to your project.
  4. Right-click your project name in Eclipse. Select **Properties** > **Java Build Path** > **Add JARs**.
  5. Select all jar files you have copied in step 3 and import them to Libraries.

- Method Three: Import jar files into Intellij IDEA

Take OSS Java SDK 2.8.3 as an example. To import jar files into Intellij IDEA, follow these steps:

1. Download the *Java SDK*.
2. Decompress the Java SDK.
3. Copy the `aliyun-sdk-oss-2.8.3.jar` file and all jar files from the lib folder to your project.
4. Right-click your project name in Intellij IDEA. Select **File** > **Project Structure** > **Modules** > **Dependencies** > **+** > **JARs or directories**.
5. Select all jar files you have copied in step 3 and import them to External Libraries.

1.2 Quick start

This topic describes how to use OSS Java SDK to perform routine operations such as bucket creation, object uploads, and object downloads.

Sample project

OSS Java SDK provides Maven-based 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 the compilation and running method of the projects, see the README.md in the project directory.

- Maven sample project: *aliyun-oss-java-sdk-demo-mvn.zip*.
- Ant sample project: *aliyun-oss-java-sdk-demo-ant.zip*.

Create a bucket

A bucket is a global namespace in OSS. It is similar to a data container that stores files. Run the following code to create a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);
```

```
// Create a bucket.
ossClient.createBucket(bucketName);

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information about bucket naming rules, see naming conventions in *Basic concepts*. For more information about how to create a bucket, see *Manage a bucket*.

Upload objects

Run the following code to upload a file to OSS:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Upload content to the specified bucket and save it as the specified
 object name.
String content = "Hello OSS";
ossClient.putObject(bucketName, objectName, new ByteArrayInputStream(
content.getBytes()));

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *Upload objects*.

Download objects

Run the following code to obtain object content:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Call ossClient.getObject to return an OSSObject instance. The
OSSObject instance contains the object content and Object Meta.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
// Call ossObject.getObjectContent to obtain object InputStream. Read
the InputStream to obtain object content.
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);
    }
    // If you do not close the reader after the data is read,
connection leaks may occur. Consequently, no available connections are
 left and an exception occurs.
    content.close();
}

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *Download objects*.

List objects

Use the following code to list objects in a bucket. You can list a maximum of 100 objects by default .

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Use ossClient.listObjects to return an ObjectListing instance. The
ObjectListing instance contains the response from listObject.
ObjectListing objectListing = ossClient.listObjects(bucketName);
// Use objectListing.getObjectSummaries to obtain the descriptions of
all objects.
for (OSSObjectSummary objectSummary : objectListing.getObjectSummaries
()) {
    System.out.println(" - " + objectSummary.getKey() + "  " +
            "(size = " + objectSummary.getSize() + ")");
}
```

```
// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *List objects* in *Manage objects*.

Delete objects

For the complete code of deleting objects, see *GitHub*.

Use the following code to delete a specified object:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Delete objects.
ossClient.deleteObject(bucketName, objectName);

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *Delete objects* in *Manage objects*.

1.3 Initialization

OSSClient serves as the OSS Java client to manage OSS resources such as buckets and objects. To initiate an OSS request with Java SDK, you need to initiate an OSSClient instance first and then modify the default configuration items of ClientConfiguration.

Create an OSSClient instance

To create an OSSClient instance, you need to specify an endpoint. For more information about endpoints, see *Regions and endpoints* and *Bind a custom domain name*.

- Use an OSS domain to create an OSSClient instance

  Use the following code to create an OSSClient instance with a domain assigned by OSS:

  ```
  // This example uses endpoint China (Hangzhou). Specify the actual
  endpoint based on your requirements.
  ```

```
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Close your OSSClient instance.
ossClient.shutdown();
```

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

- Use a custom domain (CNAME) to create an OSSClient instance

  Run the following code to create an OSSClient instance with CNAME:

```
String endpoint = "<yourEndpoint>";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create a ClientConfiguration instance. Modify parameters as
required.
ClientConfiguration conf = new ClientConfiguration();
// Enable CNAME. CNAME indicates a custom domain bound to a bucket.
conf.setSupportCname(true);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

  Note:

  ossClient.listBuckets is not available when CNAME is used.

- Create an OSSClient instance for Apsara Stack

  Run the following code to create an OSSClient instance for Apsara Stack:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
```

```
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create a ClientConfiguration instance. Modify parameters as
required.
ClientConfiguration conf = new ClientConfiguration();
// Disable CNAME.
conf.setSupportCname(false);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

- Use an IP address to create an OSSClient instance

  Run the following code to create an OSSClient instance with an IP address:

```
// In some special cases, use the IP address as an endpoint. Specify
 the actual IP address based on your requirements.
String endpoint = "http://10.10.10.10";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create ClientConfiguration.
ClientConfiguration conf = new ClientConfiguration();
// Enable OSS access with the level-2 domain. Access with the level-
2 domain is disabled by default. You need to configure this function
 for OSS Java SDK versions 2.1.2 or earlier because OSS Java SDK
versions later than 2.1.2 automatically detect IP addresses.
conf.setSLDEnabled(true);

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret, conf);

// Close your OSSClient instance.
ossClient.shutdown();
```

- Use STS to create an OSSClient instance

  Run the following code to create an OSSClient instance with Security Token Service (STS):

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
```

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret, securityToken);

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *What is RAM and STS* and *Authorized access*.

Configure an OSSClient instance

ClientConfiguration is a configuration class of OSSClient. ClientConfiguration is used to configure parameters such as user agents, host proxies, connection timeout, and the maximum number of connections. You can configure the following parameters.

Parameter	Description	Configuration method
MaxConnections	Specifies the maximum number of HTTP connections that can be enabled. The default value is 1024.	ClientConfiguration.setMaxConnections
SocketTimeout	Specifies the timeout time in milliseconds for data transmission at the sockets layer. The default value is 50,000.	ClientConfiguration.setSocketTimeout
ConnectionTimeout	Specifies the timeout time in milliseconds when connections are established. The default value is 50,000 milliseconds.	ClientConfiguration.setConnectionTimeout
ConnectionRequestTimeout	Specifies the timeout time in milliseconds for obtaining connections from the connection pool. No timeout is configured by default. No timeout is configured by default.	ClientConfiguration.setConnectionRequestTimeout
IdleConnectionTime	Specifies the idle connection timeout time. If the idle connection time in milliseconds exceeds the specified value,	ClientConfiguration.setIdleConnectionTime

Parameter	Description	Configuration method
	the connection is closed. The default value is 60,000.	
MaxErrorRetry	Specifies the maximum number of retry attempts in the case of a request error. The default value is 3.	ClientConfiguration. setMaxErrorRetry
SupportCname	Specifies whether CNAME can be used as an endpoint. You can use CNAME as an endpoint by default.	ClientConfiguration.setSupport Cname
SLDEnabled	Specifies whether access with the level-2 domain is enabled. Access with the level-2 domain is disabled by default.	ClientConfiguration. setSLDEnabled
Protocol	Specifies the protocol (HTTP or HTTPS) used to connect to OSS. The default value is HTTP.	ClientConfiguration.setProtocol
UserAgent	Specifies the user agent (the user agent in the HTTP header). The default value is “aliyun-sdk-java.”	ClientConfiguration. setUserAgent
ProxyHost	Specifies the IP address to access the proxy host.	ClientConfiguration. setProxyHost
ProxyPort	Specifies the port for the proxy host.	ClientConfiguration. setProxyPort
ProxyUsername	Specifies the username verified by the proxy host.	ClientConfiguration. setProxyUsername
ProxyPassword	Specifies the password verified by the proxy host.	ClientConfiguration. setProxyPassword

Run the following code to configure parameters for an OSSClient instance with ClientConf iguration:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
```

```
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create ClientConfiguration. ClientConfiguration is a configurat
ion class of OSSClient. You can use ClientConfiguration to configure
parameters such as user agents, host proxies, connection timeout, and
the maximum number of connections.
ClientConfiguration conf = new ClientConfiguration();

// Configure the maximum HTTP connections allowed by an OSSClient
instance. The default value is 1024.
conf.setMaxConnections(200);
// Configure the timeout time in milliseconds for data transmission at
 SSL. The default value is 50,000. conf.setSocketTimeout(10000);
conf.setSocketTimeout(10000);
// Configure the timeout time in milliseconds when connections are
established. The default value is 50,000.
conf.setConnectionTimeout(10000);
// Configure the timeout time in milliseconds for retrieving
connections from the connection pool. This function is disabled by
default.
conf.setConnectionRequestTimeout(1000);
// Configure idle connection timeout. If the idle connection time in
 milliseconds exceeds the specified value, the connection is closed.
The default value is 60,000.
conf.setIdleConnectionTime(10000);
// Configure the maximum number of retry attempts in the case of a
request error. The default value is 3.
conf.setMaxErrorRetry(5);
// Check whether CNAME can be used as an endpoint. You can use CNAME
as an endpoint by default.
conf.setSupportCname(true);
// Check whether access with the level-2 domain is enabled. Access
with the level-2 domain is disabled by default.
conf.setSLDEnabled(true);
// Configure the protocol (HTTP or HTTPS) used to connect to OSS. HTTP
 is used by default.
conf.setProtocol(Protocol.HTTP);
// Configure the user agent (the user agent in the HTTP header). The
default value is "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.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret, conf);

// Close your OSSClient instance.
```

```
ossClient.shutdown();
```

1.4 Manage a bucket

A bucket serves as a container that stores objects. Objects belong to a bucket.

Create a bucket

Run the following code to create a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Create a bucket.
String bucketName = "<yourBucketName>";
// The default storage class and ACL of the created bucket are set to
standard and private read respectively.
ossClient.createBucket(bucketName);

// Close your OSSClient.
ossClient.shutdown();
```

For more information about bucket naming rules, see *naming conventions* in *Basic concepts*.

You can specify the ACL and the storage class when you create a bucket, as shown in *Bucket-level permissions* and *Storage class*. To create Infrequent Access or archive buckets, use Java SDK versions 2.6.0 or later. The sample code is as follows:

```
CreateBucketRequest createBucketRequest = new CreateBucketRequest(
bucketName);
// Set the ACL to public read for a bucket. The default value is
Private.
createBucketRequest.setCannedACL(CannedAccessControlList.PublicRead);
// Set the storage class to Infrequent Access for a bucket. The
default value is Standard.
createBucketRequest.setStorageClass(StorageClass.IA);
ossClient.createBucket(createBucketRequest);
```

List buckets

Buckets are listed alphabetically. You can list all buckets or buckets that meet the specified conditions. To list Infrequent Access or archive buckets, use Java SDK versions 2.6.0 or later.

- List all buckets

  Run the following code to list all buckets:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// List buckets.
List<Bucket> buckets = ossClient.listBuckets();
for (Bucket bucket : buckets) {
    System.out.println(" - " + bucket.getName());
}

// Close your OSSClient.
ossClient.shutdown();
```

- List buckets with a specified prefix

  Run the following code to list buckets with a specified prefix:

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// List buckets by a specified prefix. If no prefix is specified,
list all buckets.
listBucketsRequest.setPrefix("<yourBucketPrefix>");
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

- List buckets by marker

  The value of the marker parameter indicates the name of a bucket. Run the following code to list buckets after the specified bucket (the value of marker):

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// List buckets after the specified marker. If the marker is not
specified, list all buckets.
listBucketsRequest.setMarker("<yourBucketMarker>");
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

- List a specified number of buckets

Run the following code to list a specified number of buckets:

```
ListBucketsRequest listBucketsRequest = new ListBucketsRequest();
// Set the number of buckets that can be listed to 500. The default
value is 100. The maximum value is 1,000.
listBucketsRequest.setMaxKeys(500);
BucketList bucketList = ossClient.listBuckets(listBucketsRequest);
for (Bucket bucket : bucketList.getBucketList()) {
    System.out.println(" - " + bucket.getName());
}
```

Determine whether a bucket exists

Run the following code to determine whether a specified bucket exists:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

boolean exists = ossClient.doesBucketExist("<yourBucketName>");

// Close your OSSClient.
ossClient.shutdown();
```

Configure an ACL for a bucket

The ACL of a bucket includes the following permissions.

Permission	Description	Value in Java SDK
Private	The bucket owner and the authorized users can read and write objects in the bucket. Other users cannot perform any operation on the objects.	CannedAccessControlList.Private
Public read	The bucket owner and the authorized users can read and write objects in the bucket. Other users can only read the objects in the bucket.	CannedAccessControlList.PublicRead

Permission	Description	Value in Java SDK
	Authorize this permission with caution.	
Public read-write	All users can read and write objects in the bucket. Authorize this permission with caution.	CannedAccessControlList.PublicReadWrite

Run the following code to configure the ACL for a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Set the ACL for a bucket to private read and write.
ossClient.setBucketAcl("<yourBucketName>", CannedAccessControlList.
Private);

// Close your OSSClient.
ossClient.shutdown();
```

Obtain the ACL for a bucket

Run the following code to obtain the ACL for a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);
// Obtain the access ACL for the bucket
AccessControlList acl = ossClient.getBucketAcl("<yourBucketName>");
System.out.println(acl.toString());

// Close your OSSClient.
```

```
ossClient.shutdown();
```

Obtain the region of a bucket

Run the following code to obtain the region for a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

String location = ossClient.getBucketLocation("<yourBucketName>");
System.out.println(location);

// Close your OSSClient.
ossClient.shutdown();
```

For more information about regions, see *Region* in *Basic concepts*.

Obtain bucket information

Run the following code to obtain bucket information:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We strongly recommend that you create a RAM account and use it for
API access and daily O&M. Log on to https://ram.console.aliyun.com to
create a RAM account.
String accessKeyId = "<accessKeyId>";
String accessKeySecret = "<accessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

Bucket information includes regions (region or location), creation
dates (CreationDate), owners (Owner), and permissions (Grants).
BucketInfo info = ossClient.getBucketInfo("<yourBucketName>");
// Obtain region information.
info.getBucket().getLocation();
// Obtain the creation date.
info.getBucket().getCreationDate();
// Obtain owner information.
info.getBucket().getOwner();
// Obtain permission information.
info.getGrants();
```

```
// Close your OSSClient.
ossClient.shutdown();
```

Delete a bucket

Before a bucket is deleted, ensure that all objects in the bucket and fragments that are generated from multipart upload are deleted.

Note:
To delete the fragments that are generated from multipart upload, use *Bucket.ListMultipartUploads* to list all fragments, and then use *Bucket.AbortMultipartUpload* to delete the fragments.

Run the following code to delete a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

/// Delete the bucket.
ossClient.deleteBucket("<yourBucketName>");

// Close your OSSClient.
ossClient.shutdown();
```

1.5 Upload objects

1.5.1 Simple upload

Simple upload includes streaming upload and file upload. Streaming upload uses InputStream as the source object. File upload uses a local file as the source object.

For the complete simple upload code, see *GitHub*.

Streaming upload

Use ossClient.putObject to upload a data stream to OSS.

- Upload a string

Run the following code to upload a string:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Upload a string.
String content = "Hello OSS";
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new
ByteArrayInputStream(content.getBytes()));

// Close your OSSClient instance.
ossClient.shutdown();
```

- Upload a byte array

  Run the following code to upload a byte array:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,accessKeyS
ecret);

// Upload a byte array.
byte[] content = "Hello OSS".getBytes();
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new
ByteArrayInputStream(content));

// Close your OSSClient instance.
ossClient.shutdown();
```

- Upload a network stream

  Run the following code to upload a network stream:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Upload a network stream.
InputStream inputStream = new URL("https://www.aliyun.com/").
openStream();
ossClient.putObject("<yourBucketName>", "<yourObjectName>",
inputStream);

// Close your OSSClient instance.
ossClient.shutdown();
```

- Upload a file stream

  Run the following code to upload a file stream:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in
OSS. To ensure cloud security, we recommend that you follow best
practices of Resource Access Management and log on as a RAM user to
access APIs or perform routine operations and maintenance. To create
 a RAM account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Upload a file stream.
InputStream inputStream = new FileInputStream("<yourlocalFile>");
ossClient.putObject("<yourBucketName>", "<yourObjectName>",
inputStream);

// Close your OSSClient instance.
ossClient.shutdown();
```

File upload

Run the following code to upload a local file:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
```

```
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Upload a file. <yourLocalFile> consists of a local file path and a
file name that includes an extension, for example, /users/local/myfile
.txt.
ossClient.putObject("<yourBucketName>", "<yourObjectName>", new File
("<yourLocalFile>"));

// Close your OSSClient instance.
ossClient.shutdown();
```

1.5.2 Form upload

Form upload uses an HTML form to upload a file to a specified bucket. The maximum size of files is 5 GB for each form upload.

For more information about the application scenarios of form upload, see *Form upload* in OSS Developer Guide.

Run the following code for form upload:

```
public class PostObjectSample {
    // Upload a file.
    private String localFilePath = "<yourLocalFile>";
    // This example uses endpoint China (Hangzhou). Specify the actual
 endpoint based on your requirements.
    private String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    // It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, 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 file 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 Map for the form.
        Map<String, String> formFields = new LinkedHashMap<String,
String>();
        // Specify the file name.
        formFields.put("key", this.objectName);
```

```
        // Configure Content-Disposition.
        formFields.put("Content-Disposition", "attachment;filename="
                + localFilePath);
        // Configure the callback parameter.
        Callback callback = new Callback();
        // Configure the IP address of the server you want to send the
 callback request to, for example, http://oss-demo.aliyuncs.com:23450
or http://127.0.0.1:9090.
        callback.setCallbackUrl("<yourCallbackServerUrl>");
        // Configure the host field value in the callback request
header, such as oss-cn-hangzhou.aliyuncs.com.
        callback.setCallbackHost("<yourCallbackServerHost>");
        // Configure the body field value 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 for sending 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");
        // Configure the callback parameter for Map in the form.
        setCallBack(formFields, callback);
        // Configure 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()));
        // Configure the policy.
        formFields.put("policy", encodePolicy);
        // Generate the URL for signature.
        String signaturecom = com.aliyun.oss.common.auth.ServiceSig
nature.create().computeSignature(accessKeySecret, encodePolicy);
        // Configure 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)");
            // Configure the MD5 value. The MD5 value 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.getOutputS
tream());
            // Recursively read all Map data in the form and write the
 data to the OutputStream.
            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 file and write it to the OutputStream.
            File file = new File(localFile);
            String filename = file.getName();
            String contentType = new MimetypesFileTypeMap().getContent
Type(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 FileInputS
tream(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 InputStrea
mReader(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();
    }
}
```

1.5.3 Append upload

This topic describes how to use append upload.

You cannot perform copyObject for appended objects.

Run the following code for append upload:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log 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.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

ObjectMetadata meta = new ObjectMetadata();
// Specify the content type of the object you want to upload.
meta.setContentType("text/plain");

// Configure parameters with AppendObjectRequest.
AppendObjectRequest appendObjectRequest = new AppendObjectRequest
("<yourBucketName>", "<yourObjectName>", new ByteArrayInputStream(
content1.getBytes()),meta);

// Configure a single parameter with AppendObjectRequest.
// Specify the bucket name.
//appendObjectRequest.setBucketName("<yourBucketName>");
// Specify the object name.
//appendObjectRequest.setKey("<yourObjectName>");
// Configure the type of content you want to append. Two types of
content are available: InputStream and File. Select InputStream.
//appendObjectRequest.setInputStream(new ByteArrayInputStream(content1
.getBytes()));
// Configure the type of content you want to append. Two types of
content are available: InputStream and File. Select File.
//appendObjectRequest.setFile(new File("<yourLocalFile>"));
// Specify object meta, because the specified object meta 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. This value is calculated based on ECMA-182
criteria.
System.out.println(appendObjectResult.getObjectCRC());

// Start the second append.
// nextPosition indicates the position provided for the next request,
namely, 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);

// Close your OSSClient.
ossClient.shutdown();
```

For more information about append upload, see *Append object* in OSS Developer Guide. For the complete code of append upload, see *GitHub*.

1.5.4 Multipart upload

This topic describes how to use multipart upload.

For the complete code of multipart upload, see *GitHub*.

To enable multipart upload, perform the following steps:

1. Initiate a multipart upload event.

   You can call ossClient.initiateMultipartUpload to return the globally unique uploadId created in OSS.

2. Upload parts.

   You can call ossClient.uploadPart to upload part data.

   Note:

   - For parts with a same uploadId, parts are sequenced by their part numbers. If you have uploaded a part and use the same part number to upload another part, the later part will replace the former part.
   - OSS places the MD5 value of part data in ETag and returns the MD5 value to the user.
   - SDK automatically configures Content-MD5. OSS calculates the MD5 value of uploaded data and compares it with the MD5 value calculated by SDK. If the two values vary, the error code of InvalidDigest is returned.

3. Complete multipart upload.

   After you have uploaded all parts, call partossClient.completeMultipartUpload to combine these parts into a complete object.

The following code is used as a complete example that describes the process of multipart upload:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

/* Step 1: Initiate a multipart upload event.
*/
InitiateMultipartUploadRequest request = new InitiateMultipartUpl
oadRequest(bucketName, objectName);
InitiateMultipartUploadResult result = ossClient.initiateMultipartUpl
oad(request);
// An uploadId is returned. It is the unique identifier for a part
upload event. You can initiate related operations (such as part upload
 cancelation and query) based on the uploadId.
String uploadId = result.getUploadId();

/* Step 2: Upload parts.
*/
// 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.
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 simultaneously 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 parts that have been uploaded.
    instream.skip(startPos);
    UploadPartRequest uploadPartRequest = new UploadPartRequest();
    uploadPartRequest.setBucketName(bucketName);
    uploadPartRequest.setKey(objectName);
    uploadPartRequest.setUploadId(uploadId);
    uploadPartRequest.setInputStream(instream);
    // Configure the size available for each part. Aside from the last
 part, all parts are sized 100 KB at least.
    uploadPartRequest.setPartSize(curPartSize);
    // Configure part numbers. Each part is configured with a part
number. The value can be from 1 to 10,000. If you configure a number
beyond the range, OSS returns an InvalidArgument error code.
```

```
    uploadPartRequest.setPartNumber( i + 1);
 // Do not upload each part sequentially. They can be uploaded from
 different OSSClients. Then they are sequenced and combined into a
complete object based on part numbers.
    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());
}

/* Step 3: Complete multipart upload.
*/
// Sequence parts. partETags must be sequenced by part numbers in
ascending order.
Collections.sort(partETags, new Comparator<PartETag>() {
    public int compare(PartETag p1, PartETag p2) {
        return p1.getPartNumber() - p2.getPartNumber();
    }
});
// You must provide all valid PartETags when you perform this
operation. OSS verifies the validity of all parts one by one after
 it receives PartETags. After part verification is successful, OSS
combines these parts into a complete object.
CompleteMultipartUploadRequest completeMultipartUploadRequest =
        new CompleteMultipartUploadRequest(bucketName, objectName,
uploadId, partETags);
ossClient.completeMultipartUpload(completeMultipartUploadRequest);

// Close your OSSClient.
ossClient.shutdown();
```

Cancel a multipart upload event

You can call ossClient.abortMultipartUpload to cancel a multipart upload event. If you cancel a multipart upload event, you are not allowed to perform any other operations with this uploadId anymore. The uploaded parts will be deleted.

Run the following code to cancel a multipart upload event:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Cancel multipart upload. The uploadId is from InitiateMultipartUpl
oad.
AbortMultipartUploadRequest abortMultipartUploadRequest =
```

```
        new AbortMultipartUploadRequest("<yourBucketName>", "<
yourObjectName>", "<uploadId>");
ossClient.abortMultipartUpload(abortMultipartUploadRequest);

// Close your OSSClient.
ossClient.shutdown();
```

List uploaded parts

Call ossClient.listParts to list all uploaded parts with a specified uploadId.

- Simple list

  Run the following code for simple list:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// List uploaded parts. The uploadId is returned from InitiateMu
ltipartUpload.
ListPartsRequest listPartsRequest = new ListPartsRequest("<
yourBucketName>", "<yourObjectName>", "<uploadId>");
 // Configure uploadId.
 //listPartsRequest.setUploadId(uploadId);
 // Set the maximum number of parts that can be displayed on each
page to 100. The default number is 1,000.
 listPartsRequest.setMaxParts(100);
 // Specify the initial position in the list. Only the parts with
part numbers greater than this parameter value are listed.
 listPartsRequest.setPartNumberMarker(2);
PartListing partListing = ossClient.listParts(listPartsRequest);

for (PartSummary part : partListing.getParts()) {
    // Obtain part numbers.
    part.getPartNumber();
    // Obtain the size of part data.
    part.getSize();
    // Obtain ETag.
    part.getETag();
    // Obtain the latest time parts are modified.
    part.getLastModified();
}

// Close your OSSClient.
ossClient.shutdown();
```

- List all uploaded parts

By default, listParts can only list a maximum of 1,000 parts at a time. To list more than 1,000 uploaded parts, run the following code:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(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()) {
        // Obtain part numbers.
        part.getPartNumber();
        // Obtain the part size.
        part.getSize();
        // Obtain ETag.
        part.getETag();
        // Obtain the latest time parts are modified.
        part.getLastModified();
    }
// Specify the initial position for the list. Only the parts with
part numbers greater than the initial value are listed.
listPartsRequest.setPartNumberMarker(partListing.getNextPar
tNumberMarker());
} while (partListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown()
```

- List all uploaded parts on one or more pages

  Run the following code to specify the maximum number of parts displayed on each page:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// List all uploaded parts on one or more pages.
PartListing partListing;
ListPartsRequest listPartsRequest = new ListPartsRequest("<
yourBucketName>", "<yourObjectName>", "<uploadId>");
// Set the maximum number of parts displayed on each page to 100.
listPartsRequest.setMaxParts(100);

do {
    partListing = ossClient.listParts(listPartsRequest);

    for (PartSummary part : partListing.getParts()) {
        // Obtain part numbers.
        part.getPartNumber();
        // Obtain the part size.
        part.getSize();
        // Gets the etag of the slice.
        // Obtain ETag.
        // Obtain the latest time parts are modified.
        part.getLastModified();
    }

    listPartsRequest.setPartNumberMarker(partListing.getNextPar
tNumberMarker());

} while (partListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

List all part upload events for a bucket

Call ossClient.listMultipartUploads to list all ongoing part upload events (events that have been initiated but not completed or have been canceled). You can configure the following parameters:

Parameter	Description	Configuration method
prefix	Specifies the prefix that must be included in the returned object name. Note that if you use a prefix for query, the returned object name will contain the prefix.	ListMultipartUploadsRequest.setPrefix(String prefix)
delimiter	Specifies a delimiter of a forward slash (/) used to group object names. The object between the specified prefix and the first occurrence of a delimiter of a forward slash (/) is commonPrefixes.	ListMultipartUploadsRequest.setDelimiter(String delimiter)

Parameter	Description	Configuration method
maxUploads	Specifies the maximum number of part upload events. The maximum value (also default value) you can set is 1,000.	ListMultipartUploadsRequest.setMaxUploads(Integer maxUploads)
keyMarker	Lists all part upload events with the object whose names start with a letter that comes after the keyMarker value in the alphabetical order. You can use this parameter with the uploadIdMarker parameter to specify the initial position for the specified returned result.	ListMultipartUploadsRequest.setKeyMarker(String keyMarker)
uploadIdMarker	You can use this parameter with the keyMarker parameter to specify the initial position for the specified returned result. If you do not configure keyMarker, the uploadIdMarker parameter is invalid. If you configure keyMarker, the query result contains: • All objects whose names start with a letter that comes after the keyMarker value. • All objects whose names start with a letter that is the same as the keyMarker value in the alphabetical order and the value of uploadId greater than that of uploadIdMarker.	ListMultipartUploadsRequest.setUploadIdMarker(String uploadIdMarker)

- Simple list

  Run the following code to list multipart upload events:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Lists multipart upload events. You can list 1,000 parts by
default.
ListMultipartUploadsRequest listMultipartUploadsRequest = new
ListMultipartUploadsRequest(bucketName);
MultipartUploadListing multipartUploadListing = ossClient.listMultip
artUploads(listMultipartUploadsRequest);

for (MultipartUpload multipartUpload : multipartUploadListing.
getMultipartUploads()) {
    // Obtain uploadId.
    multipartUpload.getUploadId();
    // Obtain object names.
    multipartUpload.getKey();
    // Obtain the time part upload events are initiated.
    multipartUpload.getInitiated();
}

// Close your OSSClient.
ossClient.shutdown();
```

If the returned result shows that the value of isTruncated is false, the values of nextKeyMarker and nextUploadIdMarker are returned and used as the initial position for the next object reading . If you fail to obtain all part upload events at a time, list them by pagination.

- List all part upload events

  By default, listMultipartUploads can only list a maximum of 1,000 parts at a time. To list more than 1,000 parts, run the following code to list all parts:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Lists multipart upload events.
```

```
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new
ListMultipartUploadsRequest(bucketName);

do {
    multipartUploadListing = ossClient.listMultipartUploads(
listMultipartUploadsRequest);

    for (MultipartUpload multipartUpload : multipartUploadListing.
getMultipartUploads()) {
        // Obtain uploadId.
        multipartUpload.getUploadId();
        // Obtain object names.
        multipartUpload.getKey();
        // Obtain the time part upload events are initiated.
        multipartUpload.getInitiated();
    }

    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.
getNextKeyMarker());

    listMultipartUploadsRequest.setUploadIdMarker(multipartU
ploadListing.getNextUploadIdMarker());
} while (multipartUploadListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

- List part upload events on one or more pages

  Run the following code to list part upload events on one or more pages:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Lists multipart upload events.
MultipartUploadListing multipartUploadListing;
ListMultipartUploadsRequest listMultipartUploadsRequest = new
ListMultipartUploadsRequest(bucketName);
// Configure the number of part upload events that can be displayed
on each page.
listMultipartUploadsRequest.setMaxUploads(50);

do {
    multipartUploadListing = ossClient.listMultipartUploads(
listMultipartUploadsRequest);
```

```
    for (MultipartUpload multipartUpload : multipartUploadListing.
getMultipartUploads()) {
        // Obtain uploadId.
        multipartUpload.getUploadId();
        // Obtain object names.
        multipartUpload.getKey();
        // Obtain the time part upload events are initiated.
        multipartUpload.getInitiated();
    }

    listMultipartUploadsRequest.setKeyMarker(multipartUploadListing.
getNextKeyMarker());
    listMultipartUploadsRequest.setUploadIdMarker(multipartU
ploadListing.getNextUploadIdMarker());

} while (multipartUploadListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

1.5.5 Upload progress bars

You can use progress bars to indicate the upload or download progress.

The following code is used as an example to describe how to view progress information with ossClient.putObject:

```
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
 + "/" + this.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.bytesWritt
en + " bytes have been transferred in total");
            break;
        case TRANSFER_FAILED_EVENT:
            System.out.println("Failed to upload, " + this.bytesWritt
en + " bytes have been transferred");
            break;
        default:
            break;
        }
    }

    public boolean isSucceed() {
        return succeed;
    }

    public static void main(String[] args) {
        // This example uses endpoint China (Hangzhou). Specify the
actual endpoint based on your requirements.
        String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
        // It is highly risky to log on with AccessKey of an Alibaba
 Cloud account because the account has permissions on all the APIs
in OSS. We recommend that you log on as a RAM user to access APIs or
perform routine operations and maintenance. To create a RAM account,
log on to https://ram.console.aliyun.com.
        String accessKeyId = "<yourAccessKeyId>";
        String accessKeySecret = "<yourAccessKeySecret>";
        String bucketName = "<yourBucketName>";
        String objectName = "<yourObjectName>";

        // Create an OSSClient instance.
        OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

        try {
            // Upload with a progress bar displayed.
            ossClient.putObject(new PutObjectRequest(bucketName,
objectName, new FileInputStream("<yourLocalFile>")).
                    <PutObjectRequest>withProgressListener(new
PutObjectProgressListener()));

        } catch (Exception e) {
            e.printStackTrace();
        }
        // Close your OSSClient instance.
        ossClient.shutdown();
    }
}
```

You can view progress information with ossClient.putObject, ossClient.getObject, or ossClient.uploadPart. However, you cannot view progress information with ossClient.uploadFile or ossClient.downloadFile.

For the complete code of upload progress bar, see *GitHub*.

1.5.6 Upload callback

This topic describes how to use

For the complete code of upload callback, see *GitHub*.

Run the following code for upload callback:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
// Specify the IP address of the server you want to send the callback
request to, for example, http://oss-demo.aliyuncs.com:23450 or http://
127.0.0.1:9090.
String callbackUrl = "<yourCallbackServerUrl>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

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);
// Configure the value of the host field carried in the callback
request header, such as oss-cn-hangzhou.aliyuncs.com.
callback.setCallbackHost("oss-cn-hangzhou.aliyuncs.com");
// Configure the value of the body field carried in the callback
request.
callback.setCallbackBody("{\\\"mimeType\\\":${mimeType},\\\"size\\\":
${size}}");
// Configure Content-Type for the callback request.
callback.setCalbackBodyType(CallbackBodyType.JSON);
// Configure the custom parameters used to initiate a 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 message returned from upload callback.
byte[] buffer = new byte[1024];
putObjectResult.getCallbackResponseBody().read(buffer);
// If you do not close the reader after the data is read, connection
leaks may occur. Consequently, no available connections are left and
an exception occurs.
```

```
putObjectResult.getCallbackResponseBody().close();

// Close your OSSClient instance.
ossClient.shutdown();
```

For more information, see *Upload callback* in OSS Developer Guide.

1.6 Download objects

1.6.1 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 each time until you have downloaded the entire object.

If you do not close OssObject after it is used, connection leaks may occur. Consequently, no connection is available and the program cannot run properly. Run the following code to close OssObject:

```
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
ossObject.close();
```

The following code is used for streaming downloads:

```
// This example uses the endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// ossObject includes the bucket name, key (object name), object meta
 (meta information), and InputStream.
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);
```

```
}
// If you do not close the reader after the data is read, connection
 leaks may occur. Consequently, no connection is available and the
program cannot run properly.
reader.close();

//Close your OSSClient.
ossClient.shutdown();
```

For the complete code of streaming download, see *GitHub*.

1.6.2 Download to your local file

This topic describes how to download an object from OSS to a local file.

You can run the following code to download a specified object from OSSClient to your local file:

```
//This example uses the endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Download an object to your local file. If the name of the object
is the same as that of your local file, the object will replace the
local file. If the name of the object is different from that of your
local file, the object will be downloaded and a new file will be added
 to your local device.
ossClient.getObject(new GetObjectRequest(bucketName, objectName), new
File("<yourLocalFile>"));

//Close your OSSClient.
ossClient.shutdown();
```

1.6.3 Download progress bars

You can use progress bars to indicate the upload or download progress.

The following code is used as an example to describe how to view progress information with 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) {
    // This example uses endpoint China (Hangzhou). Specify the actual
 endpoint based on your requirements.
    String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
    // It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
    String accessKeyId = "<yourAccessKeyId>";
    String accessKeySecret = "<yourAccessKeySecret>";
    String bucketName = "<yourBucketName>";
    String objectName = "<yourObjectName>";
    OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);
    try {
            // Download with a progress bar displayed.
```

```
            ossClient.getObject(new GetObjectRequest(bucketName,
objectName).
                    <GetObjectRequest>withProgressListener(new
GetObjectProgressListener()),
                    new File("<yourLocalFile>"));
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Close your OSSClient instance.
    ossClient.shutdown();
}
```

You can view progress information with ossClient.putObject, ossClient.getObject, or ossClient.uploadPart. However, you cannot view progress information with ossClient.uploadFile or ossClient.downloadFile.

For the complete code of download progress bar, see *GitHub*.

1.7 Manage objects

1.7.1 Determine whether a specified object exists

This topic describes how to determine whether a specified object exists

Run the following code to determine whether a specified object exists:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Determine whether the specified object exists.
boolean found = ossClient.doesObjectExist("<yourBucketName>", "<
yourObjectName>");
System.out.println(found);

// Close your OSSClient.
```

```
ossClient.shutdown();
```

1.7.2 Manage ACL for an object

This topic describes how to manage ACL for an object.

The following table describes the permissions included in the Access Control List (ACL) for an object.

Permission	Description	Value
default	The ACL of an object is the same with that of its bucket.	CannedAccessControlList.Default
Private	Only the object owner and authorized users can read and write the object.	CannedAccessControlList.Private
Public read	Only the object owner and authorized users can read and write the object. Other users can only read the object. Authorize this permission with caution.	CannedAccessControlList.PublicRead
Public read-write	All users can read and write the object. Authorize this permission with caution.	CannedAccessControlList.PublicReadWrite

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.

Configure an ACL for an object

You can run the following code to configure an ACL for an object:

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);
```

```
// Configure the ACL for the object to public read.
ossClient.setObjectAcl("<yourBucketName>", "<yourObjectName>",
CannedAccessControlList.PublicRead);

// Close your OSSClient.
ossClient.shutdown();
```

Obtain the ACL for an object

You can run the following code to obtain the ACL for an object:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Obtain the ACL for an object.
ObjectAcl objectAcl = ossClient.getObjectAcl("<yourBucketName>", "<
yourObjectName>");
System.out.println(objectAcl.getPermission().toString());

// Close your OSSClient.
ossClient.shutdown();
```

1.7.3 List objects

This topic describes how to list objects.

Objects are listed alphabetically. You can use ossClient.listObjects to list objects in a bucket. The following parameters can be used for listObjects:

- ObjectListing listObjects(String bucketName): lists objects in a bucket. A maximum of 100 objects can be listed.
- ObjectListing listObjects(String bucketName, String prefix): lists objects with the specified prefix in a bucket. A maximum of 100 objects can be listed.
- ObjectListing listObjects(ListObjectsRequest listObjectsRequest): provides multiple filtering functions to flexibly query objects.

The following table describes parameters for ObjectListing.

Parameter	Description	Configuration method
objectSummaries	Specifies the returned Object Meta.	List<OSSObjectSummary> getObjectSummaries()
prefix	Specifies the prefix you configure to list required objects.	String getPrefix()
delimiter	Specifies a delimiter used to group objects.	String getDelimiter()
marker	Specifies the initial object in the list.	String getMarker()
maxKeys	Specifies the maximum number of objects that can be listed.	int getMaxKeys()
nextMarker	Specifies the initial position for the next object.	String getNextMarker()
isTruncated	Specifies whether all results are returned.	boolean isTruncated()
commonPrefixes	Specifies a set of objects whose names share a prefix and end with a forward slash (/) delimiter.	List<String> getCommonPrefixes()
encodingType	Specifies the encoding type used in the response.	String getEncodingType()

Simple list

Run the following code to list objects in a specified bucket. A maximum of 100 objects can be listed by default.

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String KeyPrefix = "<yourKeyPrefix>";

// Create an OSSClient instance.
```

```
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// List objects. If you do not configure KeyPrifex, all objects in
 the bucket are listed. If you configure KeyPrifex, objects with a
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());
}

// Close your OSSClient.
ossClient.shutdown();
```

List objects with ListObjectsRequest

You can configure parameters for ListObjectsRequest to flexibly query objects. The following table describes parameters for ListObjectsRequest.

Parameter	Description	Configuration method
prefix	Specifies the prefix that must be included in the returned objects.	setPrefix(String prefix)
delimiter	Specifies a delimiter of a forward slash (/) used to group objects based on their names. The substring between the specified prefix and the first occurrence of the delimiter of a forward slash (/) is commonPrefixes.	setDelimiter(String delimiter)
marker	Lists the objects after the value of marker.	setMarker(String marker)
maxKeys	Specifies the maximum number of objects that can be listed. The default value is 100. The maximum value is 1,000.	setMaxKeys(Integer maxKeys)
encodingType	Specifies the encoding type of an object name in the response body. Only the URL encoding type is supported.	setEncodingType(String encodingType)

- List a specified number of objects

Run the following code to list a specified number of objects:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Configure the maximum number.
final int maxKeys = 200;
// List objects.
ObjectListing objectListing = ossClient.listObjects(new ListObject
sRequest(bucketName).withMaxKeys(maxKeys));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- List objects by a specified prefix

  Run the following code to list objects with a specified prefix:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Specify a prefix.
final String keyPrefix = "<yourkeyPrefix>";

// List objects with a specified prefix. By default, a maximum of
100 objects can be listed.
ObjectListing objectListing = ossClient.listObjects(new ListObject
sRequest(bucketName).withPrefix(keyPrefix));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
```

```
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- List objects by marker

  The marker parameter indicates the name of an object from which the listing begins. Run the following code to specify an object (specified with marker) after which the listing begins:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);
final String marker = "<yourMarker>";

// List objects placed after the object (specified with marker). By
default, a maximum of 100 objects can be listed.
ObjectListing objectListing = ossClient.listObjects(new ListObject
sRequest(bucketName).withMarker(marker));
List<OSSObjectSummary> sums = objectListing.getObjectSummaries();
for (OSSObjectSummary s : sums) {
    System.out.println("\t" + s.getKey());
}

// Close your OSSClient.
ossClient.shutdown();
```

- List all objects on one or more pages

  Use the following code to list all objects on one or more pages. The maximum number of objects listed on each page is specified with maxKeys.

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
```

```
OSSClient ossClient = new OSSClient(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());

// Close your OSSClient.
ossClient.shutdown();
```

- List objects by a specified prefix on one or more pages

  Run the following code to list objects by a specified prefix on one or more pages:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// A maximum of 200 objects can be listed 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());

// Close your OSSClient.
ossClient.shutdown();
```

- Specify the name of an object for encoding.

  You must encode the name of an object if the name contains the following special characters.

  OSS supports encoding with URL.

  — Single quotation marks (‘)

  — Double quotations marks (“)

  — ampersands (&)

  — angle brackets (< >)

  — Pause marker (、)

  — Chinese characters

  Run the following code to encode the name of a specified object:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(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 name of an object for encoding.
    listObjectsRequest.setEncodingType("url");

    objectListing = ossClient.listObjects(listObjectsRequest);
```

```
    // Decode the object.
    for (OSSObjectSummary objectSummary: objectListing.getObjectS
ummaries()) {
        System.out.println("Key:" + URLDecoder.decode(objectSummary.
getKey(), "UTF-8"));
    }

    // Decode the commonPrefixes parameter.
    for (String commonPrefixes: objectListing.getCommonPrefixes()) {
        System.out.println("CommonPrefixes:" + URLDecoder.decode(
commonPrefixes, "UTF-8"));
    }

    // Decode the nextMarker parameter.
    if (objectListing.getNextMarker() ! = null) {
        nextMarker = URLDecoder.decode(objectListing.getNextMarker
(), "UTF-8");
    }
} while (objectListing.isTruncated());

// Close your OSSClient.
ossClient.shutdown();
```

Folder simulation

OSS does not use folders. All elements are stored as objects. To simulate a folder on the OSS console, you actually create a 0 MB object whose name ends with a forward slash (/). This object can be uploaded and downloaded. By default, the OSS console displays an object whose name ends with a forward slash (/) as a folder.

The delimiter and prefix parameters can be used to simulate folder functions.

- Prefix: specifies the prefix as the name of a folder. The folder is used to list all files (in the folder) and subfolders (directories in the folder) that start with this prefix. These files and subfolders are included in the Objects list.
- Delimiter: If the delimiter is specified as a forward slash (/), only the files and subfolders (directories) are displayed. Subfolders (directories) are included in the CommonPrefixes list, while the files and folders in subfolders are not displayed.

For more information about folders, see *Folder simulation*. For the complete code of creating an object, see *GitHub*.

Assume that the following objects are stored in a bucket: oss.jpg, fun/test.jpg, fun/movie/001.avi , and fun/movie/007.avi. Forward slashes (/) are used as delimiters for folders. The subsequent examples show how to simulate folder functions.

- List all objects in a bucket

Run the following code to list all objects in a bucket:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(
bucketName);

// List objects.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries())
 {
    System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("CommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}

// Disable the OSSClient.
// Close your OSSClient.
```

The returned results are as follows:

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg
oss.jpg

CommonPrefixes:
```

- List all objects in a specified folder

  Run the following code to list all objects in a specified folder:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(
bucketName);
// Configure the Prefix parameter to obtain all objects in the fun
folder.
listObjectsRequest.setPrefix("fun/");

// Recursively list all objects in the fun folder.
ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
for (OSSObjectSummary objectSummary : listing.getObjectSummaries())
 {
    System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("\nCommonPrefixes:");
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}

// Close your OSSClient.
ossClient.shutdown();
```

The returned results are as follows:

```
Objects:
fun/movie/001.avi
fun/movie/007.avi
fun/test.jpg

CommonPrefixes:
```

- List objects and subfolders in a folder

  Use the following code to list objects and subfolders in a folder:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
```

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Create ListObjectsRequest.
ListObjectsRequest listObjectsRequest = new ListObjectsRequest(
bucketName);

// Set forward slashes (/) as the delimiter for folders.
listObjectsRequest.setDelimiter("/");

// List all objects and folders in the fun folder.
listObjectsRequest.setPrefix("fun/");

ObjectListing listing = ossClient.listObjects(listObjectsRequest);

// Print all objects.
System.out.println("Objects:");
// List objects in the fun folder in a objectSummaries list.
for (OSSObjectSummary objectSummary : listing.getObjectSummaries())
 {
    System.out.println(objectSummary.getKey());
}

// Print all commonPrefixes.
System.out.println("\nCommonPrefixes:");
// commonPrefixs lists all subfolders in the fun folder. The fun/
movie/001.avi and fun/movie/007.avi objects are not listed, because
 they are objects in the movie folder (a subfolder of the fun folder
).
for (String commonPrefix : listing.getCommonPrefixes()) {
    System.out.println(commonPrefix);
}

// Close your OSSClient.
ossClient.shutdown();
```

The returned results are as follows:

```
Objects:
fun/test.jpg

CommonPrefixes:
fun/movie/
```

1.7.4 Delete objects

This topic describes how to delete objects.

Warning:

Delete objects with caution because deleted objects cannot be recovered.

Delete an object

Run the following code to delete a single object:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Delete an object.
ossClient.deleteObject(bucketName, objectName);

// Close your OSSClient.
ossClient.shutdown();
```

Delete multiple objects

You can delete a maximum of 1,000 objects simultaneously. Objects can be deleted in two modes : detail (verbose) and simple (quiet) modes.

- verbose: returns the list of objects that you have deleted successfully. The default mode is verbose.
- quiet: returns the list of objects that 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 a mode that the deletion result uses. True indicates quiet. False indicates verbose. The default mode is verbose.	setQuiet(boolean)
encodingType	Encodes the name of the returned objects. Only URL is supported.	setEncodingType(String)

The following table describes the parameters for DeleteObjectsResult.

Parameter	Description	Configuration method
deletedObjects	Specifies the deletion result . verbose indicates the list of objects you have deleted successfully. quiet indicates the list of objects you failed to delete.	List<String> getDeletedObjects ()
encodingType	Encodes the name of objects in deletedObjects.	getEncodingType()

Run the following code to delete multiple objects simultaneously:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Delete objects.
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();

// Close your OSSClient.
ossClient.shutdown();
```

For the complete code of deleting multiple objects simultaneously, see *GitHub*.

1.7.5 Restore an archive object

This topic describes how to restore an archive object.

For the complete code of restoring an archive object, see *GitHub*.

You must restore an archive object before you read it. Do not call restoreObject for non-archive objects.

The state conversion process of an archive object is as follows:

1. An archive object is in the frozen state.
2. After you submit it for restoration, the server restores the object. The object is in the restoring state.
3. You can read the object after it is restored. The restored state of the object lasts one day by default. You can prolong this period to a maximum of seven days. Once this period expires, the object returns to the frozen state.

Run the following code to restore an object:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

ObjectMetadata objectMetadata = ossClient.getObjectMetadata(bucketName
, key);

// Verify whether the object is an archive object.// Verify that the
file is an archive file.
StorageClass storageClass = objectMetadata.getObjectStorageClass();
if (storageClass == StorageClass.Archive) {
    // Restore the object.
    ossClient.restoreObject(bucketName, objectName);

    // Wait until the object is restored.
    do {
        Thread.sleep(1000);
        objectMetadata = ossClient.getObjectMetadata(bucketName,
objectName);
    } while (! objectMetadata.isRestoreCompleted());
}

// Obtain the restored object.
OSSObject ossObject = ossClient.getObject(bucketName, objectName);
ossObject.getObjectContent().close();

// Close your OSSClient.
```

```
ossClient.shutdown();
```

For more information about the archive storage classes, see *Introduction to storage classes*.

1.7.6 Manage a symbolic link

This topic describes how to manage a symbolic link.

Create a symbolic link

A symbolic link is a special object that maps to an object similar to a shortcut used in Windows.

You can configure user-defined Object Meta for symbolic links.

Run the following code to create a symbolic link:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String symLink = "<yourSymLink>";
String destinationObjectName = "<yourDestinationObjectName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Create Object Meta.
ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("text/plain");
// Set property to property-value for user-defined metadata.
metadata.addUserMetadata("property", "property-value");

// Create CreateSymlinkRequest.
CreateSymlinkRequest createSymlinkRequest = new CreateSymlinkRequest(
bucketName, symLink, destinationObjectName);

// Configure Object Meta.
createSymlinkRequest.setMetadata(metadata);

// Create symbolic links.
// Create a symbolic link.

// Close your OSSClient.
ossClient.shutdown();
```

For more information about symbolic links, see *PutSymlink*.

Obtain object content for a symbolic link

To obtain a symbolic link, you must have the read permission on it. Run the following code to obtain the object content for a symbolic link:

```
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String symLink = "<yourSymLink>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Obtain a symbolic link.
OSSSymlink symbolicLink = ossClient.getSymlink(bucketName, symLink);
// Print the content of the object that the symbolic link directs to.
System.out.println(symbolicLink.getSymlink());
System.out.println(symbolicLink.getTarget());
System.out.println(symbolicLink.getRequestId());

// Close your OSSClient.
ossClient.shutdown();
```

For more information about symbolic links, see *GetSymlink*.

1.8 Authorized access

This topic describes how to authorize access to users.

Use STS for temporary access authorization

OSS supports Alibaba Cloud Security Token Service (STS) for temporary access authorization. STS is a web service that provides a temporary access token to a cloud computing user. Through the STS, you can assign a third-party application or a RAM user (you can manage the user ID) an access credential with a custom validity period and permissions. For more information about STS, see *STS introduction*.

STS advantages:

- Your long-term key (AccessKey) is not exposed to a third-party application. You only need to generate an access token and send the access token to the third-party application. You can customize access permissions and the validity of this token.

- You do not need to keep track of permission revocation issues. The access token automatically becomes invalid when it expires.

For more information about the process of access to OSS with STS, see *RAM and STS scenario practices* in OSS Developer Guide.

Run the following code to create a signature request with STS:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";

// After a user obtains a temporary STS credential, the OSSClient is
generated with the security token and temporary access key (AccessKeyI
d and AccessKeySecret).
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret, securityToken);

// Perform operations on OSS.

// Close your OSSClient.
ossClient.shutdown();
```

Sign a URL to authorize temporary access

- Sign a URL

  You can provide a signed URL to a visitor for temporary access. When you sign a URL, you can specify the expiration time for a URL to restrict the period of access from visitors.

- Sign a URL for access with HTTP GET

  Use the following code to sign a URL that allows access with HTTP GET:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on
to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Set the expiration time of a URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
// Generate the URL that allows access with HTTP GET. Visitors can
use a browser to access relevant content.
URL url = ossClient.generatePresignedUrl(bucketName, objectName,
expiration);

// Close your OSSClient.
ossClient.shutdown();
```

- Sign a URL for access with other HTTP methods

  A URL needs to be signed for temporary access from a visitor to perform other operations such as file upload and deletion. For example:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on
to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String securityToken = "<yourSecurityToken>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// After a user obtains a temporary STS credential, the OSSClient
 is generated with the security token and temporary access key (
AccessKeyId and AccessKeySecret).
OSSClient ossClient = new OSSClient(endpoint, accessKeyId,
accessKeySecret, securityToken);

GeneratePresignedUrlRequest request = new GeneratePresignedUrl
Request(bucketName, objectName, HttpMethod.PUT);
// Set the expiration time of a URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
request.setExpiration(expiration);
// Set ContentType.
request.setContentType(DEFAULT_OBJECT_CONTENT_TYPE);
// Configure custom Object Meta.
request.addUserMetadata("author", "aliy");

// Sign a URL that allows access with HTTP PUT.
URL signedUrl = ossClient.generatePresignedUrl(request);

Map<String, String> requestHeaders = new HashMap<String, String>();
requestHeaders.put(HttpHeaders.CONTENT_TYPE, DEFAULT_OBJECT_CONTE
NT_TYPE);
requestHeaders.put(OSS_USER_METADATA_PREFIX + "author", "aliy");

// Upload a file with the signed URL.
ossClient.putObject(signedUrl, new ByteArrayInputStream("Hello OSS".
getBytes()), -1, requestHeaders, true);
```

```
// Close your OSSClient.
ossClient.shutdown();
```

Visitors can use a signed URL to upload a file by passing in the HttpMethod.PUT parameter.

- Add specified parameters to a URL

  Run the following code to add specified parameters to a URL:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on
to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient  = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Create a request.
GeneratePresignedUrlRequest generatePresignedUrlRequest = new
GeneratePresignedUrlRequest(bucketName, objectName);
// Set HttpMethod to PUT.
generatePresignedUrlRequest.setMethod(HttpMethod.PUT);
// Add custom Object Meta.
generatePresignedUrlRequest.addUserMetadata("author", "baymax");
// Add Content-Type.
generatePresignedUrlRequest.setContentType("application/octet-stream
");
// Set the expiration time of a URL to one hour.
Date expiration = new Date(new Date().getTime() + 3600 * 1000);
generatePresignedUrlRequest.setExpiration(expiration);
// Generate the signed URL.
URL url = ossClient.generatePresignedUrl(generatePresignedUrlRequest
);

// Close your OSSClient.
ossClient.shutdown();
```

- Use a signed URL to obtain or upload an object

  — Use a signed URL to obtain an object

  Run the following code to obtain a specified object through a signed URL:

```
// This example uses endpoint China (Hangzhou). Specify the actual
 endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba
Cloud account because the account has permissions on all APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
```

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient  = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

Date expiration = DateUtil.parseRfc822Date("Wed, 18 Mar 2022 14:20
:00 GMT");
GeneratePresignedUrlRequest request = new GeneratePresignedUrl
Request(bucketName, objectName, HttpMethod.GET);
// Configure the expiration time.
request.setExpiration(expiration);
// Generate a signed URL that allows HTTP GET access.
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>();
// Add a request header to GetObject.
customHeaders.put("Range", "bytes=100-1000");
OSSObject object = ossClient.getObject(signedUrl,customHeaders);

// Close your OSSClient.
ossClient.shutdown();
```

- Use a signed URL to upload a file

  Run the following code to upload a file with a signed URL:

```
// This example uses endpoint China (Hangzhou). Specify the actual
 endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba
Cloud account because the account has permissions on all APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";
String objectName = "<yourObjectName>";

// Create an OSSClient instance.
OSSClient ossClient  = new OSSClient(endpoint, accessKeyId,
accessKeySecret);

// Generate the signed URL.
Date expiration = DateUtil.parseRfc822Date("Thu, 19 Mar 2019 18:00
:00 GMT");
GeneratePresignedUrlRequest request = new GeneratePresignedUrl
Request(bucketName, objectName, HttpMethod.PUT);
// Configure the expiration time.
request.setExpiration(expiration);
// Configure Content-Type.
request.setContentType("application/octet-stream");
// Configure custom Object Meta.
request.addUserMetadata("author", "aliy");
// Generate a signed URL that allows access with HTTP PUT.
```

```
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);
// Add a request header to 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);

// Close your OSSClient.
ossClient.shutdown();
```

1.9 CORS

This topic describes how to perform CORS.

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to another region. OSS provides CORS APIs for convenient cross-origin access control.

For more information, see *Cross-origin resource sharing* and *PutBucketcors* in OSS Developer Guide.

Configure CORS rules

Run the following code to configure CORS rules for the specified bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

SetBucketCORSRequest request = new SetBucketCORSRequest(bucketName);

// Create a CORS rule. A maximum of 10 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 source of the cross-origin access request.
allowedOrigin.add( "http://www.b.com");
```

```
ArrayList<String> allowedMethod = new ArrayList<String>();
// Specify the cross-region request methods (GET, PUT, DELETE, POST,
and HEAD) that are allowed.
allowedMethod.add("GET");

ArrayList<String> allowedHeader = new ArrayList<String>();
// Specify whether the header specified in Access-Control-Request-
Headers of pre-flight request (OPTIONS) is allowed.
allowedHeader.add("x-oss-test");

ArrayList<String> exposedHeader = new ArrayList<String>();
// Specify the response header that allows user access from applicatio
ns.
exposedHeader.add("x-oss-test1");
// AllowedOrigins and AllowedMethods allow only one wildcard asterisk
 (*). Wildcard asterisks (*) indicate that all sources of the cross-
origin requests and operations are allowed.
corRule.setAllowedMethods(allowedMethod);
corRule.setAllowedOrigins(allowedOrigin);
// AllowedHeaders and ExposeHeaders do not allow wildcard asterisks
 (*).
corRule.setAllowedHeaders(allowedHeader);
corRule.setExposeHeaders(exposedHeader);
// Specify the cache time (seconds) for the response of browser pre-
flight (OPTIONS) requests to a specific resource.
corRule.setMaxAgeSeconds(10);

// A maximum of 10 rules is allowed.
putCorsRules.add(corRule);
// The existing rules will be replaced.
request.setCorsRules(putCorsRules);
ossClient.setBucketCORS(request);

// Close your OSSClient.
ossClient.shutdown();
```

Obtain CORS rules

Run the following code to obtain CORS rules:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

ArrayList<CORSRule> corsRules;
// Obtain the CORS rule list.
corsRules =  (ArrayList<CORSRule>) ossClient.getBucketCORSRules(
bucketName);
```

```
for (CORSRule rule : corsRules) {
    for (String allowedOrigin1 : rule.getAllowedOrigins()) {
     // Obtain allowed sources of cross-origin requests.
     System.out.println(allowedOrigin1);
    }

    for (String allowedMethod1 : rule.getAllowedMethods()) {
     // Obtain the allowed cross-origin request method.
     System.out.println(allowedMethod1);
    }

    if (rule.getAllowedHeaders().size() > 0){
     for (String allowedHeader1 : rule.getAllowedHeaders()) {
         // Obtain the header list for allowed cross-origin requests.
         System.out.println(allowedHeader1);
     }
    }

    if (rule.getExposeHeaders().size() > 0) {
        for (String exposeHeader : rule.getExposeHeaders()) {
         // Obtain the header for an allowed cross-origin request.
         System.out.println(exposeHeader);
        }
    }

    if ( null ! = rule.getMaxAgeSeconds()) {
        System.out.println(rule.getMaxAgeSeconds());
    }
}

// Close your OSSClient.
ossClient.shutdown();
```

Delete CORS rules

Run the following code to delete all CORS rules for the specified bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Delete all CORS rules for a specified bucket.
ossClient.deleteBucketCORSRules(bucketName);

// Close your OSSClient.
```

```
ossClient.shutdown();
```

1.10 Set logging

You can enable access logs to record bucket access to log files, which are stored in a specified bucket.

The log file format is as follows:

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

For more information about access log files, see *Set access logging*.

Enable access logging

Run the following code to enable bucket access logging:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

SetBucketLoggingRequest request = new SetBucketLoggingRequest("<
yourSourceBucketName>");
// Configure the bucket that stores log files.
request.setTargetBucket("<yourTargetBucketName>");
// Configure the log file storage directory.
request.setTargetPrefix("<yourTargetPrefix>");
ossClient.setBucketLogging(request);

// Close your OSSClient.
ossClient.shutdown();
```

View access logging configurations

Run the following code to view the access logging configurations for a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
```

```
// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

BucketLoggingResult result = ossClient.getBucketLogging("<yourSource
BucketName>");
System.out.println(result.getTargetBucket());
System.out.println(result.getTargetPrefix());

// Close your OSSClient.
ossClient.shutdown();
```

Disable access logging

Run the following code to disable access logging for a bucket:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

SetBucketLoggingRequest request = new SetBucketLoggingRequest("<
yourSourceBucketName>");
request.setTargetBucket(null);
request.setTargetPrefix(null);
ossClient.setBucketLogging(request);

// Close your OSSClient.
ossClient.shutdown();
```

1.11 Static website hosting

You can set your bucket configuration to the static website hosting mode. After the configuration takes effect, you can access this static website with the bucket domain and be redirected to a specified index page or error page.

For more information about static website hosting, see *Static website hosting*.

Configure static website hosting

Run the following code to configure static website hosting:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

SetBucketWebsiteRequest request = new SetBucketWebsiteRequest("<
yourBucketName>");
request.setIndexDocument("index.html");
request.setErrorDocument("error.html");
ossClient.setBucketWebsite(request);

// Close your OSSClient.
ossClient.shutdown();
```

View static website hosting configurations

Run the following code to view static website hosting configurations:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

BucketWebsiteResult result = ossClient.getBucketWebsite("<yourBucket
Name>");
System.out.println(result.getIndexDocument());
System.out.println(result.getErrorDocument());

// Close your OSSClient.
ossClient.shutdown();
```

Delete static website hosting configurations

Run the following code to delete static website hosting configurations:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
```

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

ossClient.deleteBucketWebsite("<yourBucketName>");

// Close your OSSClient.
ossClient.shutdown();
```

1.12 Anti-leech

This topic describes how to use anti-leech.

To prevent your data on OSS from being leeched, OSS supports anti-leeching through the referer field settings in the HTTP header, including the following parameters:

- Referer whitelist: Used to allow access only for specified domains to OSS data.
- Empty referer: Determines whether the referer can be empty. If it is not allowed, only requests with the referer filed in their HTTP or HTTPS headers can access OSS data.

For more information about anti-leaching, see *Anti-leeching settings*.

Configure the referer whitelist

Run the following code to configure the referer whitelist:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

List<String> refererList = new ArrayList<String>();
// Add the referer field. The referer field allows question marks (?)
and asterisks (*) for wildcard use.
refererList.add("http://www.aliyun.com");
refererList.add("http://www. *.com");
refererList.add("http://www.?.aliyuncs.com");
// Configure the referer list for a bucket. Set the parameter to true
, which allows the referer field to be empty.
BucketReferer br = new BucketReferer(true, refererList);
ossClient.setBucketReferer(bucketName, br);
```

```
// Close your OSSClient.
ossClient.shutdown();
```

Obtain a referer whiltelist

Run the following code to obtain a referer whiltelist:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// Obtain the referer list for a bucket.
BucketReferer br = ossClient.getBucketReferer(bucketName);
List<String> refererList = br.getRefererList();
for (String referer : refererList) {
 System.out.println(referer);
}

// Close your OSSClient.
ossClient.shutdown();
```

Clear a referer whitelist

Run the following code to clear a referer whitelist:

```
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String bucketName = "<yourBucketName>";

// Create an OSSClient instance.
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeyS
ecret);

// You cannot clear a referer whitelist directly. To clear a referer
 whitelist, you need to create the rule that allows an empty referer
field and replace the original rule with the new rule.
BucketReferer br = new BucketReferer();
ossClient.setBucketReferer(bucketName, br);

// Close your OSSClient.
```

```
ossClient.shutdown();
```

1.13 Exception handling

OSS Java SDK has two types of exceptions: ClientException and OSSException. Both share the properties of RuntimeException.

Example for handling exceptions

Run the following code to handle exceptions:

```
try {
    // Perform operations on OSS, for example, object uploads.
    ossClient.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 (ossClient ! = null) {
        ossClient.shutdown();
    }
}
```

ClientException

The ClientException exception occurs when the client attempts to send requests and when the data is transmitted to OSS. For example, when a request is sent, ClientException may occur due to unavailable network connections. ClientException may occur during file uploads due to IO exceptions.

OSSException

ServiceException indicates a server exception. The exception occurs because a server error message is parsed. OSSException includes the error code and information returned from OSS. It is used to locate and handle problems.

The following table describes common OSSException information.

Parameter	Description
Code	Specifies the error code returned by OSS.
Message	Specifies the detailed error information returned by OSS.
RequestId	Specifies the UUID. It is unique and used to identify a request. If a problem persists, send the RequestId to OSS developers for help.
HostId	Identifies an OSS cluster. Ensure that the value of HostId is consistent with the Host (endpoint to access a bucket) used in the request.

Common OSS error codes

Error Code	Description	HTTP status code
AccessDenied	Access denied	403
BucketAlreadyExists	The bucket already exists.	409
BucketNotEmpty	The bucket is not empty.	409
EntityTooLarge	The entity size exceeds the maximum limit.	400
EntityTooSmall	The entity size is below the minimum limit.	400
FileGroupTooLarge	The total file group size exceeds the maximum limit.	400
FilePartNotExist	No such part exists.	400
FilePartStale	The part has expired.	400
InvalidArgument	The parameter format is invalid .	400
InvalidAccessKeyId	No such AccessKeyId exists.	403
InvalidBucketName	The bucket name is invalid.	400
InvalidDigest	The digest is invalid.	400
InvalidObjectName	The object name is invalid.	400
InvalidPart	The part is invalid.	400
InvalidPartOrder	The part order is invalid.	400

Error Code	Description	HTTP status code
InvalidTargetBucketForLogging	The buckets that store log files are invalid.	400
InternalError	Internal OSS error	500
MalformedXML	The XML format is invalid.	400
MethodNotAllowed	The method is not allowed.	405
MissingArgument	Parameters are not configured.	411
MissingContentLength	The content length is not configured.	411
NoSuchBucket	No such bucket exists.	404
NoSuchKey	No such object exists.	404
NoSuchUpload	No such uploadId exists.	404
NotImplemented	The methods are not implemented.	501
PreconditionFailed	Precondition error	412
RequestTimeTooSkewed	The local time set for OSSClient is deviated from the time set for the OSS server by over 15 minutes.	403
RequestTimeout	Request timeout	400
SignatureDoesNotMatch	Signature mismatch	403
InvalidEncryptionAlgorithmError	The specified encryption algorithm (the entropy encoding type) is invalid.	400

2 Python

2.1 Quick start

This topic describes how to use OSS Python SDK to perform routine operations such as bucket creation, object uploads, and object downloads.

Create a bucket

Run the following code to create a bucket:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
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)
```

Upload objects

Run the following code to upload a file to OSS:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# <yourLocalFile> consists of a local file path and a file name with
extension, for example: /users/local/myfile.txt
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

Download objects

Run the following code to download a specified object to a local file:

```
# -*- coding: utf-8 -*-
```

```
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# <yourLocalFile> consists of a local file path and a file name with
extension, for example: /users/local/myfile.txt
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

List objects

Run the following code to list 10 objects in a specified bucket:

```
# -*- coding: utf-8 -*-
import oss2
from itertools import islice

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We strongly recommend that you create a RAM account and use it for
API access and daily O&M. Log on to https://ram.console.aliyun.com to
create a RAM account.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
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

Run the following code to delete a specified object:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')
```

```
bucket.delete_object('<yourObjectName>')
```

2.2 Initialization

This topic describes how to initialize OSS Python SDK.

Most operations in OSS Python SDK are performed through oss2.Service and oss2.Bucket.

- The oss2.Service class is used to list buckets.
- The oss2.Bucket class is used to upload, download, and delete objects, and configure buckets.

To initialize the two classes, you must specify an endpoint. The oss2.Service class does not support access with the custom domain (CNAME). For more information about endpoints, see *Regions and endpoints* and *Bind a custom domain name*.

Initialize the oss2.Service class

For more information, see the bucket listing part in *Manage buckets*.

Initialize the oss2.Bucket class.

- Use a OSS domain name to initialize the class

  Run the following code to initialize the oss2.Bucket class with a OSS domain name.

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This case uses the Hangzhou endpoint as an example. Fill in the
region endpoint name according to the actual circumstances.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'

bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>')
```

- Use a custom domain name to initialize the class

  Run the following code to initialize the oss2.Bucket class with a custom domain name

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
```

```
# Use a custom domain name: my-domain.com as an example. is_cname
=True indicates that the CNAME is enabled. Cname means binding a
custom domain name to the storage space. CNAME indicates a custom
domain bound to a bucket.
cname = 'http://my-domain.com'
bucket = oss2.Bucket(auth, cname, '<yourBucketName>', is_cname=True)
```

- Set connection timeout

  Run the following code to set connection timeout:

```
# -*- coding: utf-8 -*-
import oss2

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'

# Set the connection timeout to 30 seconds.
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', connect_ti
meout=30)
```

- Disable CRC verification

  CRC verification is enabled by default during uploads and downloads to ensure data integrity during uploads and downloads. Run the following code to disable CRC verification:

  Warning:

  We recommended that you do not disable CRC verification. If you disable CRC verification, the data integrity during uploads and downloads is not guaranteed.

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
endpoint = 'http://oss-cn-hangzhou.aliyuncs.com'
```

```
bucket = oss2.Bucket(auth, endpoint, '<yourBucketName>', enable_crc=
False)
```

2.3 Bucket

A bucket serves as a container that stores objects. Objects belong to a bucket.

Create a bucket

Run the following code to create a bucket:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# You can create a new bucket in a specified region by specifying the
 endpoint and bucket name. This example uses endpoint China (Hangzhou
). Specify the actual endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# The default storage class and ACL of the created bucket are set to
standard and private read respectively.
bucket.create_bucket()
```

For more information about bucket naming rules, see *naming conventions* in *Basic concepts*.

You can specify the ACL and the storage class when you create a bucket, as shown in *Bucket-level permissions* and *Storage class*. The sample code is as follows:

```
# Set the bucket ACL to public read and set the storage class to
Infrequent Access.
bucket.create_bucket(oss2. BUCKET_ACL_PUBLIC_READ, oss2.models.
BucketCreateConfig(oss2. BUCKET_STORAGE_CLASS_IA))
```

List buckets

Run the following code to list buckets:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
service = oss2. Service(auth, 'http://oss-cn-hangzhou.aliyuncs.com')
```

```
print([b.name for b in oss2. BucketIterator(service)])
```

Configure an ACL for a bucket

The ACL of a bucket includes the following permissions.

Permission	Description	Value
Private	The bucket owner and the authorized users can read and write objects in the bucket. Other users cannot perform any operation on the objects.	oss2. BUCKET_ACL _PRIVATE
Public read	The bucket owner and the authorized users can read and write objects in the bucket . Other users can only read the objects in the bucket. Authorize this permission with caution.	oss2. BUCKET_ACL _PUBLIC_READ
Public read-write	All users can read and write objects in the bucket. Authorize this permission with caution.	oss2. BUCKET_ACL _PUBLIC_READ_WRITE

Run the following code to configure an ACL for a bucket:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Set the ACL for the bucket to private.
bucket.put_bucket_acl(oss2. BUCKET_ACL_PRIVATE)
```

Obtain the ACL for a bucket

Run the following code to obtain the ACL for a bucket:

```
# -*- coding: utf-8 -*-
import oss2
```

```
# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

print(bucket.get_bucket_acl().acl)
```

Obtain bucket information

Run the following code to obtain bucket information:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2. Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Obtain bucket information
bucket_info = bucket.get_bucket_info()
print('name: ' + bucket_info.name)
print('storage class: ' + bucket_info.storage_class)
print('creation date: ' + bucket_info.creation_date)
print('intranet_endpoint: ' + bucket_info.intranet_endpoint)
print('extranet_endpoint ' + bucket_info.extranet_endpoint)
print('owner: ' + bucket_info.owner.id)
print('grant: ' + bucket_info.acl.grant)
```

Delete a bucket

Before a bucket is deleted, ensure that all objects in the bucket and fragments that are generated from multipart upload are deleted.

Note:

To delete the fragments that are generated from multipart upload, use *Bucket.ListMultipartUploads* to list all fragments, and then use *Bucket.AbortMultipartUpload* to delete the fragments.

Run the following code to delete a bucket:

```
# -*- coding: utf-8 -*-
```

```
import oss2
# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2. Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
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')
```

For buckets which are not empty, you can list and delete objects stored in these buckets first (stop uploading for objects uploaded by multipart upload), and then delete the buckets.

2.4 Upload objects

2.4.1 Simple upload

This topic describes how to use simple upload.

You can use the bucket.put_object method to upload a file. This method supports multiple types of inputs, which are listed in the following table.

Type	Upload method
String	Direct upload
Bytes	Direct upload
Unicode	Upload after converting to bytes encoded in UTF-8.
Local files	Upload as File object, which must be opened in the binary mode.
Network stream	Upload as Iterable object in the Chunked Encoding method.

- Upload a string

  You can run the following code to upload a string:

  ```
  # -*- coding: utf-8 -*-
  import oss2
  ```

```
# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Return value.
result = bucket.put_object('<yourObjectName>', 'content of object')
# HTTP return code.
print('http status: {0}'.format(result.status))
# Request ID. The request ID is the unique identification of a
request. We strongly recommend you to add this parameter in the
program log.
print('request_id: {0}'.format(result.request_id))
# Etag is a unique attribute of the value returned by put_object.
print('ETag: {0}'.format(result.etag))
# HTTP response header.
print('date: {0}'.format(result.headers['date']))
```

- Upload bytes

  You can run the following code to upload bytes:

```
# -*- coding: utf-8 -*-

import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

bucket.put_object('<yourObjectName>', b'content of object')
```

- Upload Unicode

  You can run the following code to upload Unicode:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
```

```
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

bucket.put_object('<yourObjectName>', u'content of object')
```

- Upload a local file

  You can run the following code to upload a local file:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Files must be opened in the binary mode because the number of
bytes included in the file is required for upload.
with open('<yourLocalFile>', 'rb') as fileobj:
    # The seek method specifies that the read or write operations
start from the 1000th byte. The upload starts from the 1000th byte
until the end of the file.
    fileobj.seek(1000, os.SEEK_SET)
    # The tell method is used to return to the current location.
    current = fileobj.tell()
    bucket.put_object('<yourObjectName>', b'content of object')
```

  OSS Python SDK provides a more convenient method to upload local files:

```
bucket.put_object_from_file('<yourObjectName>', '<yourLocalFile>')
```

- Upload a network stream

  You can run the following code to upload a network stream:

```
# -*- coding: utf-8 -*-
import oss2
import requests

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# The requests.get method returns an Iterable object, and the Python
 SDK uploads the network stream in the Chunked Encoding method.
input = requests.get('http://www.aliyun.com')
```

```
bucket.put_object('<yourObjectName>', input)
```

2.4.2 Append upload

This topic describes how to use append upload.

Run the following code for append upload:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Set the position (the Position parameter) where the object is
appended to 0.
result = bucket.append_object('<yourObjectName>', 0, 'content of first
 append')
# If the object is not uploaded for the first time, you can use the
bucket.head_object method or the next_position attribute returned in
the last append upload to get the append position.
bucket.append_object('<yourObjectName>', result.next_position, '
content of second append')
```

If the object already exists, exceptions occur in the following two sceanrios:

- The file cannot be uploaded with append upload. In this case, an ObjectNotAppendable exception occurs.
- The file can be uploaded with append upload, but the append position does not match the current file length. In this case, a PositionNotEqualToLength exception occurs.

2.4.3 Multipart upload

This topic describes how to use multipart upload.

To enable multipart upload, perform the following steps:

1. Initialization (using bucket.init_multipart_upload): Obtain the Upload ID.
2. Upload parts (using bucket.upload_part): Upload data parts. You can upload multiple parts concurrently.
3. Completing multipart upload (using bucket.complete_multipart_upload): After you have uploaded all parts, combine these parts into a complete object.

Run the following code for multipart upload:

```
# -*- coding: utf-8 -*-
import os
from oss2 import SizedFileAdapter, determine_part_size
from oss2.models import PartInfo
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

key = '<yourObjectName>'
filename = '<yourLocalFile>'

total_size = os.path.getsize(filename)
# Use determine_part_size to determine the part size.
part_size = determine_part_size(total_size, preferred_size=100 * 1024)

# Initialize a multipart upload event.
upload_id = bucket.init_multipart_upload(key).upload_id
parts = []

# Upload 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 re-calculates the initial append location.
        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 multipart upload.
bucket.complete_multipart_upload(key, upload_id, parts)

# Verify the multipart upload.
with open(filename, 'rb') as fileobj:
    assert bucket.get_object(key).read() == fileobj.read()
```

2.4.4 Upload progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of progress bars, see *GitHub*.

The following code is used as an example to describe how to view progress information with bucket.put_object:

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2
# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')
# If the length of the data to be uploaded 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 implement
progress bars.
bucket.put_object('<yourObjectName>', 'a'*1024*1024, progress_callback
=percentage)
```

2.4.5 Upload callback

This topic describes how to use upload callback.

For the complete code of upload callback, see *GitHub*.

Run the following code for upload callback:

```
# -*- coding: utf-8 -*-
import json
import base64
import os
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Prepare callback parameters.
callback_dict = {}
# Configure the IP address of the server you want to send the callback
 request to, for example, http://oss-demo.aliyuncs.com:23450 or http
://127.0.0.1:9090.
```

```
callback_dict['callbackUrl'] = 'http://oss-demo.aliyuncs.com:23450'
# Configure the value of the Host field carried in the callback
request header, such as oss-cn-hangzhou.aliyuncs.com.
callback_dict['callbackHost'] = 'oss-cn-hangzhou.aliyuncs.com'
# Configure the value of the body field carried 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 Base64-encoded.
callback_param = json.dumps(callback_dict).strip()
base64_callback_body = base64.b64encode(callback_param)
# Encoded callback parameters are carried in the request header and
are sent to OSS.
headers = {'x-oss-callback': base64_callback_body}

# Upload and callback.
result = bucket.put_object('<yourObjectName>', 'a'*1024*1024, headers)
```

The put_object, put_object_from_file, and complete_multipart_upload methods provide upload callback features. For more information about upload callback, see *Upload callback* in the Development Guide.

2.5 Download objects

2.5.1 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 each time until you have downloaded the entire object.

Run the following code for streaming download:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
// This example uses the endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# The return value of bucket.get_object is a file-like object, which
is also an iterable object.
object_stream = bucket.get_object('<yourObjectName>')
print(object_stream.read())
```

```
# The get_object interface returns a stream, which must be read by
read() before being used to calculate the CRC checksum of the returned
 object data. Therefore, you must perform CRC verification after
calling the interface.
if object_stream.client_crc ! = object_stream.server_crc:
    print "The CRC checksum between client and server is inconsistent
!"
```

Run the following code to download data to a local file:

```
import shutil

# object_stream is a file-like object. Therefore, you can use the
shutil.copyfileobj method to download data to a local file.
object_stream = bucket.get_object('<yourObjectName>')
with open('<yourLocalFile>', 'wb') as local_fileobj:
    shutil.copyfileobj(object_stream, local_fileobj)
```

Run the following code to copy an object to another object using streaming copy.

```

# object_stream is an iterable object, which can be copied to another
object using streaming copy.
object_stream = bucket.get_object('<yourObjectName>')
bucket.put_object('<yourBackupObjectName>', object_stream)
```

2.5.2 Download to your local file

This topic describes how to download an object from OSS to a local file.

You can run the following code to download the specified OSS object to a local file:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
#  This example uses the endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Download an object to your local file. If the name of the object is
the same as that of your local file, the object will replace the local
 file. If the name of the object is different from that of your local
 file, the object will be downloaded and a new file will be added to
your local device.
```

```
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>')
```

2.5.3 Range download

If you only need part of the data in an object, you can use range downloads to download specified content.

Run the following code for range download:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses the endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Obtain the data ranging from byte 0 to the 99th byte of the object
 (including byte 0 and the 99th byte , a total of 100 bytes). If the
 specified value range is invalid, for example, the specified range
includes a negative number, or the specified value is larger than the
object size, the entire object is downloaded.
object_stream = bucket.get_object('<yourObjectName>', byte_range=(0,
99))
```

2.5.4 Download progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of progress bars, see *GitHub*.

The following code is used as an example to describe how to view progress information with bucket.get_object_to_file:

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import os, sys
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')
```

```
# If the HTTP response header does not carry Content-Length, 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
progress bar.
bucket.get_object_to_file('<yourObjectName>', '<yourLocalFile>',
progress_callback=percentage)
```

2.6 Manage objects

2.6.1 Determine whether a specified object exists

This topic describes how to determine whether a specified object exists.

Run the following code to determine whether a specified object exists:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

exist = bucket.object_exists('<yourObjectName>')
# If the returned value is true, it indicates that the specified
object exists. If the returned value is false, it indicates that the
specified object does not exist.
if exist:
 print('object exist')
else:
 print('object not eixst')
```

2.6.2 List objects

Objects are listed alphabetically. You can use OSS Python SDK to list objects in different methods.

Simple list

Run the following code to list 10 objects in a specified bucket.

```
# -*- coding: utf-8 -*-
import oss2
```

```
from itertools import islice

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

for b in islice(oss2. ObjectIterator(bucket), 10):
    print(b.key)
```

List objects by a specified prefix

Run the following code to list objects with a specified prefix:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# List objects by a specified prefix 100 objects are listed by default
.
for obj in oss2. ObjectIterator(bucket, prefix = 'img-'):
    print(obj.key)
```

List all objects in a bucket

Run the following code to list all objects in a bucket:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Set the delimiter parameter to a slash (/).
for obj in oss2. ObjectIterator(bucket, delimiter = '/'):
```

```
 # Determine whether the obj is a directory using the is_prefix method
.
    if obj.is_prefix():  # directory
        print('directory: ' + obj.key)
    else:                # object
        Print ('file: '+ obj. Key)
```

2.6.3 Copy objects

You can copy an object from a bucket (source bucket) to another bucket (target bucket) in the same region.

You must note the following limits before copy an object:

- You must have the read and write permission on the source object.
- You cannot copy an object from a region to another region. For example, you cannot copy an object from a bucket in Hangzhou region to a bucket in Qingdao region.

Simple copy

You can use simple copy for objects smaller than 1 GB. Run the following code for simple copy:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourDestinationBucketName>')

bucket.copy_object('<yourSourceBucketName>', '<yourSourceObjectName>',
 '<yourDestinationObjectName>')
```

Copy a large-sized object

To copy an object larger than 1 GB, you need to use UploadPartCopy. To enable UploadPartCopy , perform the following steps:

1. Use bucket.init_multipart_upload to initiate an UploadPartCopy task.
2. Start UploadPartCopy with bucket.upload_part_copy. Aside from the last part, all parts must be larger than 100 KB.
3. Submit the UploadPartCopy task with bucket.complete_multipart_copy.

Run the following code for UploadPartCopy task

```
# -*- coding: utf-8 -*-
import oss2
from oss2.models import PartInfo
from oss2 import determine_part_size

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
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 an UploadPartCopy task
upload_id = bucket.init_multipart_upload(dst_object).upload_id
parts = []

# Copy parts one by one.
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 UploadPartCopy task.
bucket.complete_multipart_upload(dst_object, upload_id, parts)
```

For more information about UploadPartCopy, see *UploadPartCopy*

2.6.4 Restore an archive object

This topic describes how to restore an archive object.

You must restore an archive object before you read it. Do not call restoreObject for non-archive objects.

The state conversion process of an archive object is as follows:

1. An archive object is in the frozen state.

2. After you submit it for restoration, the server restores the object. The object is in the restoring state.
3. You can read the object after it is restored. The restored state of the object lasts one day by default. You can prolong this period to a maximum of seven days. Once this period expires, the object returns to the frozen state.

Run the following code to restore an object:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')
objectName = '<yourObjectName>'

bucket.restore_object(objectName)
```

For more information about the archive storage classes, see *Introduction to storage classes*.

2.6.5 Manage a symbolic link

This topic describes how to manage a symbolic link.

Create a symbolic link

A symbolic link is a special object that maps to an object similar to a shortcut used in Windows.

Run the following code to create a symbolic link:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

objectName = '<yourObjectName>'
symlink = "<yourSymlink>";
```

```
bucket.put_symlink(objectName, symlink)
```

For more information about symbolic links, see *PutSymlink*.

Obtain object content for a symbolic link

To obtain a symbolic link, you must have the read permission on it. Run the following code to obtain the object content for a symbolic link:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

symlink = "<yourSymlink>";

bucket.get_symlink(symlink)
```

For more information about symbolic links, see *GetSymlink*.

2.7 Authorized access

This topic describes how to authorize access to users.

Sign a URL to authorize temporary access

You can provide a signed URL to a visitor for temporary access. When you sign a URL, you can specify the expiration time for a URL to restrict the period of access from visitors.

Run the following code to download objects with a signed URL:

```
# -*- coding: utf-8 -*-
import oss2

# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
auth = oss2.Auth('<yourAccessKeyId>', '<yourAccessKeySecret>')
# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
bucket = oss2.Bucket(auth, 'http://oss-cn-hangzhou.aliyuncs.com', '<
yourBucketName>')

# Set the expiration time of the  URL to one hour.
```

```
print(bucket.sign_url('GET', '<yourObjectName>', 60))
```

Use STS for temporary access authorization

For the complete code of using STS, see *GitHub*.

You can use Security Token Service (STS) for temporary access authorization. For more information about STS, see *Introduction* in the access control API (STS) reference. For more information about accounts and authorization, see *STS temprory access authorization* in best practices.

You must install the official Python STS client first:

```
pip install aliyun-python-sdk-sts
```

Please use version 2.0.6 and later. Run the following code to upload files with STS temporary authorization:

```
# -*- coding: utf-8 -*-

from aliyunsdkcore import client
from aliyunsdksts.request.v20150401 import AssumeRoleRequest
import json
import oss2

# This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
endpoint = 'oss-cn-hangzhou.aliyuncs.com'
# It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
access_key_id = '<yourAccessKeyId>'
access_key_secret = '<yourAccessKeySecret>'
bucket_name = '<yourBucketName>'
# role_arn is the resource name of the role.
role_arn = '<yourRoleArn>'

clt = client.AcsClient(access_key_id, access_key_secret, 'cn-hangzhou
')
req = AssumeRoleRequest.AssumeRoleRequest()

# Set the format of returned values to JSON.
req.set_accept_format('json')
req.set_RoleArn(role_arn)
req.set_RoleSessionName('session-name')
body = clt.do_action(req)

# Use the AccessKeyId and AccessKeySecret to apply for a temporary
token from STS.
token = json.loads(body)

# Use the authentication information in the temporary token to
initialize the StsAuth instance.
auth = oss2.StsAuth(token['Credentials']['AccessKeyId'],
```

```
                    token['Credentials']['AccessKeySecret'],
                    token['Credentials']['SecurityToken'])

# Use the StsAuth instance to initialize the bucket.
bucket = oss2.Bucket(auth, endpoint, bucket_name)

# Upload a string.
bucket.put_object('object-name.txt', b'hello world')
```

3 PHP

3.1 Manage objects

3.1.1 Determine whether a specified object exists

Use the following code to determine whether a specified 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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);
```

3.1.2 Manage ACL for an object

The following table describes the permissions included in the Access Control List (ACL) for an object.

Permission	Description	Value
default	The ACL of an object is the same with that of its bucket.	default
Private	Only the object owner and authorized users can read and write the object.	private
Public read	Only the object owner and authorized users can read and write the object. Other users can only read the object . Authorize this permission with caution.	public-read
Public read-write	All users can read and write the object. Authorize this permission with caution.	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.

Configure an ACL for an object

You can run the following code to configure an ACL for a specified 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
// Set the ACL for the object to public read, which is inherited from
the bucket by default.
$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");
```

Obtain the ACL for an object

You can run the following code to obtain the ACL for a specified 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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);
```

3.1.3 Manage Object Meta

For more information about Object Meta, see *Object Meta* in OSS Developer Guide.

Configure Object Meta

Run the following code to configure Object Meta:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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 Meta

Run the following code to modify Object Meta:

```
<? 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;
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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");
```

Obtain Object Meta

Run the following code to obtain Object Meta:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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");
}
```

3.1.4 Delete objects

Delete an object

Run the following code 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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

Run the following code to delete multiple objects simultaneously:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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");
```

3.1.5 Copy objects

Simple copy

Run the following code to copy objects smaller than 1 GB:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$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-sized object

To copy an object larger than 1 GB, you need to use multipart copy. To use multipart copy, perform the following steps:

1. Use $ossClient->initiateMultipartUpload to initiate an UploadPartCopy task.
2. Use $ossClient->uploadPartCopy to start a multipart copy. Aside from the last part, all parts must be larger than 100 KB.
3. Submit the multipart copy task with $ossClient->completeMultipartUpload.

Run the following code for multipart copy:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
```

```
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

$src_bucket = "<yourSourceBucketName>";
$src_object = "<yourSourceObjectName>";
$dst_bucket = "<yourDestinationBucketName>";
$dst_object = "<yourDestinationObjectName>";

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    // Initialize a multipart copy.
    $upload_id = $ossClient->initiateMultipartUpload($dst_bucket, $
dst_object);

    $copyId = 1;
    // Copy the parts one-by-one.
    $eTag = $ossClient->uploadPartCopy( $src_bucket, $src_object, $
dst_bucket, $dst_object,$copyId, $upload_id);
    $upload_parts[] = array(
        'PartNumber' => $copyId,
        'ETag' => $eTag,
    );

    // Complete the multipart copy.
    $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");
```

3.1.6 Restore an archive object

You must restore an archive object before you read it. Do not call restoreObject for non-archive objects.

The state conversion process of an archive object is as follows:

1. An archive object is in the frozen state.
2. After you submit it for restoration, the server restores the object. The object is in the restoring state.
3. You can read the object after it is restored. The restored state of the object lasts one day by default. You can prolong this period to a maximum of seven days. Once this period expires, the object returns to the frozen state.

Run the following code to restore 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    $ossClient->restoreObject($bucket, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

For more information about the archive storage classes, see *Introduction to storage classes*.

3.1.7 Manage a symbolic link

Create a symbolic link

A symbolic link is a special object that maps to an object similar to a shortcut used in Windows.

You can configure user-defined Object Meta for symbolic links.

Run the following code to create a symbolic link:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$symlink = "<yourSymlink>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    $ossClient->putSymlink($bucket, $symlink, $object);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Obtain object content for a symbolic link

To obtain a symbolic link, you must have the read permission on it. Run the following code to obtain the object content for a symbolic link:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$symlink = "<yourSymlink>";
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    $Symlinks = $ossClient->getSymlink($bucket, $symlink);
```

```
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
var_dump($Symlinks);
```

3.1.8 Enable MD5 verification

MD5 verification is performed to ensure data integrity during transmission. MD5 verification brings performance loss. MD5 verification is disabled by default during file uploads.

Run the following code to enable MD5 verification during file uploads:

```
<? php
if (is_file(__DIR__ . '/../Autopoload. 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

$options = array(OssClient::OSS_CHECK_MD5 => true);
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    $ossClient->uploadFile($bucket, $object, __FILE__, $options);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

The putObject, uploadFile, appendObject, appendFile and multiuploadFile methods support MD5 verification.

3.2 Authorized access

Sign a URL to authorize temporary access

You can provide a signed URL to a visitor for temporary access. When you sign a URL, you can specify the expiration time for a URL to restrict the period of access from visitors. For the complete code of authorizing access, see *GitHub*.

- Sign a URL to download an object

  Run the following code to sign a 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on
to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";
$securityToken = "<yourSecurityToken>";

// Set the expiration time of the URL to 3600 seconds.
$timeout = 3600;
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint, false, $securityToken);

    // Generates a signed URL for access with 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 access a signed URL by running code or input it in a
browser.
$request = new RequestCore($signedUrl);
// The signed URL is accessed with GET by default.
$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");
};
```

- Sign a URL to upload a file

  Run the following code to sign a URL to upload a 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;
use OSS\Core\OssException;
use OSS\Http\RequestCore;
use OSS\Http\ResponseCore;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on
to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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 a URL for access with 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);
// The signed URL is accessed with PUT.
$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");
};
```

Use STS for temporary access authorization

OSS supports Alibaba Cloud Security Token Service (STS) for temporary access authorization. For more information about STS, see *Introduction* of the access control API (STS) reference. For more information on accounts and authorization, see *STS temporary access authorization*.

Run the following code to use STS for temporary file upload access authorization:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
 account because the account has permissions on all APIs in OSS. We
 recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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();
```

```
}
```

3.3 Static website hosting

You can set your bucket configuration to the static website hosting mode. After the configuration takes effect, you can access this static website with the bucket domain and be redirected to a specified index page or error page.

For more information about static website hosting, see *Static website hosting*.

For the complete code of the following scenarios, see *GitHub*.

Configure static website hosting

Run the following code 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

// Configure static website hosting: The index paga is index.html and
the error page is 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");
```

View static website hosting configurations

Run the following code to view 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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

Run the following code 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
```

```
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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");
```

3.4 Manage lifecycle rules

You can use OSS to configure lifecycle rules to reduce storage costs. The lifecycle management rules can be used for automatic deletion of expired objects and fragments, or storage class conversion to Infrequent Access for objects that will expire soon. Each rule includes:

- Rule ID: It identifies a rule. Ensure that each rule ID in a bucket is unique.
- Policy: You can configure a policy by using the following configuration methods. Only one method can be configured for a bucket.
  - Configure by Prefix: You can create multiple rules with this method. Ensure that each prefix is unique.
  - Configure for Entire Bucket: You can configure only one rule with this method.
- Expired: You can configure the following configuration methods:
  - Set by Number of Days: It specifies the number of days from the last modification time of objects.
  - Set by Date: It specifies the date prior to which objects are created. If objects are created prior to the date, these objects are deleted.
- Status: This lifecyle rule takes effect or not.

For more information about lifecycle management, see *Manage object lifecycle*. For the complete code of lifecycle management, see *GitHub*.

Configure lifecycle rules

Run the following code to configure lifecycle rules:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

// Configure the rule ID and object name prefix.
$ruleId0 = "rule0";
$matchPrefix0 = "A0/";
$ruleId1 = "rule1";
$matchPrefix1 = "A1/";

$lifecycleConfig = new LifecycleConfig();
$actions = array();
// Set the expiration duration to 3 days after last modified date.
$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();
// Configure expiration date to delete objects that are created before
 the date.
$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");
```

View lifecycle rules

Run the following code to view lifecycle rules:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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");
```

Clear lifecycle rules

Run the following code to clear lifecycle rules:

```
<? 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;
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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");
```

3.5 Set logging

You can enable access logs to record bucket access to log files, which are stored in a specified bucket.

The log file format is as follows:

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

For more information about access log files, see *Set access logging*. For the complete code of access logging, see *GitHub*.

Enable access logging

Run the following code to enable bucket access logging:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
```

```
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

$option = array();
// Configure the bucket that stores log files.
$targetBucket = "<yourBucketName>";
$targetPrefix = "access.log";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    // Enable access logging.
    $ossClient->putBucketLogging($bucket, $targetBucket, $targetPrefix
, $option);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

View access logging configurations

Run the following code to view the access logging configurations 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$loggingConfig = null;
$options = array();
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    // View access log 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 access logging

Run the following code to disable access 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    // Disable access logging.
    $ossClient->deleteBucketLogging($bucket);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

3.6 CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to another region. OSS provides CORS APIs for convenient cross-origin access control.

For more information, see *Cross-origin resource sharing* in OSS Developer Guide. For the complete code of CORS, see *GitHub*.

Configure CORS rules

Run the following code to configure CORS rules for the 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;
use OSS\Model\CorsConfig;
use OSS\Model\CorsRule;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$corsConfig = new CorsConfig();
$rule = new CorsRule();
// AllowedHeaders and ExposeHeaders do not allow wildcards.
$rule->addAllowedHeader("x-oss-header");
// AllowedOlowedMethods allows only one wildcard asterisk (*).
Wildcard asterisks (*) indicate that all sources of the cross-origin
requests and operations are allowed.
$rule->addAllowedOrigin("http://www.b.com");
$rule->addAllowedMethod("POST");
$rule->setMaxAgeSeconds(10);
// A maximum of 10 rules are allowed for each bucket.
$corsConfig->addRule($rule);

try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

    // The existing rules will be replaced.
    $ossClient->putBucketCors($bucket, $corsConfig);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

Obtain CORS rules

Run the following code to obtain CORS rules:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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

Run the following code to delete all CORS rules for 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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");
```

3.7 Anti-leech

To prevent your data on OSS from being leeched, OSS supports anti-leeching through the referer field settings in the HTTP header, including the following parameters:

- Referer whitelist: Used to allow access only for specified domains to OSS data.
- Empty referer: Determines whether the referer can be empty. If it is not allowed, only requests with the referer filed in their HTTP or HTTPS headers can access OSS data.

For more information about anti-leaching, see *Anti-leeching settings*. For the complete code of anti-leech, see *GitHub*.

Configure the referer whitelist

Run the following code to configure the referer whitelist:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$refererConfig = new RefererConfig();
// Allow empty referers.
```

```
$refererConfig->setAllowEmptyReferer(true);
// Add the referer field. The referer field allows question marks (?)
and asterisks (*) for wildcard use.
$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");
```

Obtain a referer whiltelist

Run the following code to obtain a referer whiltelist:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$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 a referer whitelist

Run the following code to clear a referer whitelist:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China (Hangzhou). Specify the actual
endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

$refererConfig = new RefererConfig();
try{
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

 // You cannot clear a referer whitelist directly. To clear a referer
 whitelist, you need to create the rule that allows an empty referer
field and replace the original rule with the new rule.
    $ossClient->putBucketReferer($bucket, $refererConfig);
} catch(OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

3.8 Bind a custom domain name

You can bind a custom domain name to your OSS bucket and by adding a CNAME record in the Domain Name System (DNS). After you bind a custom domain name to a bucket, you can access the bucket with your own domain name. For example, if your domain name is "my-domain.com" and all your images were previously located at `http://img.my-domain.com/x.jpg`, after you migrate the image data to OSS, you can still access the images with the original IP address by binding a custom domain name to the bucket.

For more information, see *Bind a custom domain name* in the OSS Developer Guide.

Add a CNAME record

Run the following code to add a CNAME record for a bucket:

```
<? php
if (is_file(__DIR__ . '/../autoload.php')) {
    Quire_once _ DIR __. '/../autoload.php';
}
if (is_file(__DIR__ . '/../vendor/autoload.php')) {
    require_once __DIR__ . '/../vendor/autoload.php';
}

use OSS\OssClient;
use OSS\Core\OssException;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

// Configure the custom domain name.
$myDomain = '<yourDomainName>';
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

 // Add a CNAME record.
    $ossClient->addBucketCname($bucket, $myDomain);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

View CNAME records

Run the following code to obtain CNAME records added to the 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;
```

```
// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

 // View CNAME records.
    $cnameConfig = $ossClient->getBucketCname($bucket);
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
var_dump($cnameConfig);
print(__FUNCTION__ . ": OK" . "\n");
```

Delete a CNAME record

Run the following code to delete a CNAME record added to 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
routine operations and maintenance. To create a RAM account, log on to
 https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";

// Configure the custom domain name.
$myDomain = '<yourDomainName>';
try {
    $ossClient = new OssClient($accessKeyId, $accessKeySecret, $
endpoint);

 // Delete a CNAME record.
    $ossClient->deleteBucketCname($bucket, $myDomain);
```

```
} catch (OssException $e) {
    printf(__FUNCTION__ . ": FAILED\n");
    printf($e->getMessage() . "\n");
    return;
}
print(__FUNCTION__ . ": OK" . "\n");
```

3.9 Image processing

Image Processing (IMG) is a massive, secure, cost-effective and highly reliable image processing service. After source images are uploaded to OSS, you can process images on any Internet device at any anytime, from anywhere through simple RESTful APIs.

For more information about IMG, see *Image Processing*. For the complete code of image processing, see *GitHub*.

Basic features

OSS offers the following IMG features:

- *Retrieve dominant image tones*
- *Format conversion*
- *Resize images*
- *Incircle*
- *Adaptive orientation*
- *Brightness*
- *Add watermarks*: allows you to add images, texts, or image-text watermarks to another image.
- *Image processing*
- *Image processing access rules*: calls multiple IMG functions.

Usage

IMG uses standard HTTP GET. You can configure IMG parameters in QueryString of a URL.

If the ACL of an image object is private read and write, only authorized users are allowed for access.

- Anonymous access

You can use the following format in level-3 domains to access a processed image:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-
process=image/<yourAction>,<yourParamValue>
```

Parameter	Description
bucket	Specifies the name of a bucket.
endpoint	Specifies endpoint used to access a region.
object	Specifies the name of an image object.
image	Specifies the reserved identifier for IMG.
action	Specifies the operations on an image, such as scaling, cropping, and rotating.
param	Specifies the parameter that indicates the operation on an image.

- Basic operations

  For example, scale an image to a width of 100 PX. Adjust the height based on the ratio.

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
  process=image/resize,w_100
  ```

- Customized image styles

  You can use the following format in level-3 domains to access a processed image:

  ```
  http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-
  process=style/<yourStyleName>
  ```

  - style: specifies a reserved identifier of a customized image style.
  - yourStyleName: specifies the name of a custom image style. It is the name specified in the rule that is created on the OSS console.

  Example:

  ```
  http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
  process=style/oss-pic-style-w-100
  ```

- Cascade operations

Cascade operations allow you to perform multiple operations on an image sequentially.

```
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
```

- HTTPS access

  IMG supports access through HTTPS. Example:

  ```
  https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
  process=image/resize,w_100
  ```

- Authorized access

  Authorized access allows you to customize an image style, HTTPS access, and cascade operations.

  Run the following code to generate a signed URL for 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;

// It is highly risky to log on with AccessKey of an Alibaba Cloud
account because the account has permissions on all the APIs in OSS.
We recommend that you log on as a RAM user to access APIs or perform
 routine operations and maintenance. To create a RAM account, log on
 to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
actual endpoint based on your requirements.
$endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
$bucket= "<yourBucketName>";
$object = "<yourObjectName>";

$ossClient = new OssClient($accessKeyId, $accessKeySecret, $endpoint
);

// Generate a signed URL that is valid for 3600 seconds and can be
directly accessed by browsers.
$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 with SDK

  You can use SDK to access and process an image object.

  SDK allows you to customize image styles, HTTPS access, and cascade operations.

  ▬ Basic operations

  Run the following code to perform basic operations on an image:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba
Cloud account because the account has permissions on all APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
$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");

// Scale 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.
$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 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);
```

- Customize an image style

  Run the following code to customize an image style:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba
Cloud account because the account has permissions on all APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
$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 an 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 operations

  Use the following code to perform cascade operations on an image:

```
<? 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;

// It is highly risky to log on with AccessKey of an Alibaba
Cloud account because the account has permissions on all APIs in
 OSS. We recommend that you log on as a RAM user to access APIs
 or perform routine operations and maintenance. To create a RAM
account, log on to https://ram.console.aliyun.com.
$accessKeyId = "<yourAccessKeyId>";
$accessKeySecret = "<yourAccessKeySecret>";
// This example uses endpoint China East 1 (Hangzhou). Specify the
 actual endpoint based on your requirements.
$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 operations.
$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);
```

IMG tools

You can use the IMG viewer *ImageStyleViever* to directly view the IMG result.

3.10 Exception handling

OSS PHP SDK exceptions (OssException) include various errors, such as invalid parameters and file missing. You can use the getMessage method to obtain error messages.

For more information about OssException, see *GitHub*.

Example for handling exceptions

The following code shows how to handle exceptions when creating a existing bucket and prints error messages.

```
    try {
        $ossClient->createBucket($bucket);
    } catch (OssException $e) {
  print("Exception:" . $e->getMessage() . "\n");
    }
```

You can also obtain the following information:

Parameter	Description
HTTPStatus	HTTP status code. You can get the status code using the getHTTPStatus method.
ErrorCode	Specifies the error code returned by OSS. You can get the error code using the getErrorCode method.
ErrorMessage	Specifies the error messages returned by OSS . You can get the error messages using the getErrorMessage method.
RequestId	Specifies the UUID. It is unique and used to identify a request. If a problem persists, send the RequestId to OSS developers for help. You can get the RequestId using the getRequestId method.
Details	Specifies detailed error messages returned by OSS. You can get the detailed error messages using the getDetails method.

Common OSS error codes

Error code	Description	HTTP status code
AccessDenied	Access denied	403
BucketAlreadyExists	The bucket already exists.	409
BucketNotEmpty	The bucket is not empty.	409
EntityTooLarge	The entity size exceeds the maximum limit.	400
EntityTooSmall	The entity size is below the minimum limit.	400
FileGroupTooLarge	The total file group size exceeds the maximum limit.	400
FilePartNotExist	No such part exists.	400
FilePartStale	The part has expired.	400
InvalidArgument	The parameter format is invalid .	400
InvalidAccessKeyId	No such AccessKeyId exists.	403
InvalidBucketName	The bucket name is invalid.	400
InvalidDigest	The digest is invalid.	400
InvalidObjectName	The object name is invalid.	400
InvalidPart	The part is invalid.	400
InvalidPartOrder	The part order is invalid.	400
InvalidTargetBucketF orLogging	The buckets that store log files are invalid.	400
InternalError	Internal OSS error	500
MalformedXML	The XML format is invalid.	400
MethodNotAllowed	The method is not allowed.	405
MissingArgument	Parameters are not configured.	411
MissingContentLength	The content length is not configured.	411
NoSuchBucket	No such bucket exists.	404
NoSuchKey	No such object exists.	404
NoSuchUpload	No such uploadId exists.	404

Error code	Description	HTTP status code
NotImplemented	The methods are not implemented.	501
PreconditionFailed	Precondition error	412
RequestTimeTooSkewed	The local time set for OSSClient is deviated from the time set for the OSS server by over 15 minutes.	403
RequestTimeout	Request timeout	400
SignatureDoesNotMatch	Signature mismatch	403
InvalidEncryptionAlgorithmError	The specified encryption algorithm (the entropy encoding type) is invalid.	400

4 C

4.1 Preface

This document is written based on OSS C SDK 3.5.0.

Compatibility

- For SDK series of 3.*.*: compatible
- For SDK series of 2.*.*:
  - Windows: compatible
  - Linux API: compatible except for the linked list (aos_list_t) traversal interface.
    - os_list_for_each_entry
    - aos_list_for_each_entry_reverse
    - aos_list_for_each_entry_safe
    - aos_list_for_each_entry_safe_reverse
- For SDK series of 1.0.0: incompatible for only the following APIs
  - 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 SDK series of 0.0.*: incompatible

SDK source code

For the Java SDK source code, see *GitHub*.

Code samples

OSS SDK for C provides a variety of code samples for your reference or use. The following table describes the content of code samples:

Sample file	Content
oss_put_object_sample	*Upload objects*

Sample file	Content
oss_get_object_sample.c	*Download objects*
oss_append_object_sample.c	*Append upload*
oss_multipart_upload_sample.c	*Multipart upload*
oss_resumable_sample.c	*Resumable upload*, *Resumable download*
oss_head_object_sample.c	*Manage the metadata of an object*
oss_list_object_sample.c	*List objects*
oss_delete_object_sample.c	*Delete an object*
oss_callback_sample.c	*Upload callback*
oss_progress_sample.c	*Progress bar*, *Progress bar*
oss_crc_sample.c	CRC check during upload or download
oss_image_sample.c	*Image processing*

4.2 Installation

Install OSS C SDK in Linux

- Install CMake and third-party libraries

  When installing OSS C SDK in Linux, you must install CMake and the following third-party libraries: curl, apr, apr-util and minixml.

  The following parameters are required to install the environment:

Parameter	Description	Version requirement
CMake	Build and installation tool	Version 2.6.0 and later
curl	Focuses on network problems .	Version 7.32.0 and later
apr-util	Focuses on memory management and cross-platform operations.	Version 1.5.2 and later
minixml	Parses XML returned by a request.	We recommend you use version 2.9.

  Select the operation system that you need to install CMake and third-party libraries.

  — Ubuntu/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
```

If the mxml installation package is not included in the yum source, you can install the libraries using RPM. Download the rpm installation package based on your operation system:

- 64-bit: *mxml-2.9-1.x86_64.rpm*
- 32-bit: *mxml-2.9-1.i386.rpm*

Run the following code to install mxml with the rpm installation package:

```
rpm -ivh mxml-2.9-1.x86_64.rpm
```

— RedHat/Aliyun/CentOS

- Install CMake

```
sudo yum install cmake
```

- Install third-party libraries

```
sudo yum install curl-devel apr-devel apr-util-devel  mxml mxml
-devel
```

— SuSE

- Install CMake

```
zypper install cmake
```

- Install third-party libraries

```
zypper install libcurl-devel libapr1-devel libapr-util1-devel
mxml-devel
```

— Other Linux

- Install CMake: *Download link*

The common installation method is as follows:

```
./configure
make
```

```
make install
```

Note:

When running ./configure, the default installation path is /usr/local/. If you want to specify another path, use the ./configure --prefix option.

- Install libcurl: *Download link*

  The common installation method is as follows:

```
./configure
make
make install
```

- Install apr: *Download link*

  The common installation method is as follows:

```
./configure
make
make install
```

- Install apr-util: *Download link*

  The common installation method is as follows:

```
// You must specify the --with-apr option in the installation.
./configure --with-apr=/your/apr/install/path
make
make install
```

- Install minixml: *Download link*

  The common installation method is as follows:

```
./configure
make
sudo make install
```

- Download SDK

  — *Download SDK directly*

  — *Download SDK from GitHub*

  — *Download historical SDK*

- Install SDK

  Use **cmake**. If third-party libraries, such as url, apr, apr-util and mxml are not installed in the default method during the compiling process, you must specify the path of their header files and library files.

Install the SDK as follows:

```
cmake .
make
make install
```

An installation example in CentOS is as follows:

```
cmake -f CMakeLists.txt
// The compilation type is Release. Common compilation types include
: Debug, Release, RelWithDebInfo, and MinSizeRel, in which Debug is
the default type.
-DCMAKE_BUILD_TYPE=Release
// Customize the installation directory.
-DCMAKE_INSTALL_PREFIX=/usr/local/
// Specify the path where the header files and library files of
third party libaries, such as curl, apr, apr-util, and xml, are
located,
-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 occurs
during the compiling process, it indicates that the apr-1-config
file cannot be found in the default path. Add this option.
-DAPR_CONFIG_BIN=/path/to/bin/apr-1-config
// If the "Could not find apu-config/apu-1-config" error occurs, add
 this option.
-DAPU_CONFIG_BIN=/path/to/bin/apu-1-config
```

Install OSS C SDK in Windows

- Download SDK
  - *Download SDK directly*
  - *Download SDK from GitHub*
  - *Download historical SDK*
- Install SDK

  For detailed steps and common problems when using Visual Studio to compile OSS C SDK, see *Compile and Use Alibaba Cloud OSS C SDK in Windows*.

  Note:
  If you use Visual Studio 2012 and later versions, it prompts you to upgrade the project to the latest compiler and libraries. We recommended you to determine whether to upgrade based on your project. You can upgrade if your project uses the latest compiler and libraries.

4.3 Quick start

This topic describes how to use OSS C SDK to complete routine operations such as bucket creation, objects uploads, and object downloads.

Demo

- Linux demo

  Linux-based demo: *aliyun-oss-c-sdk-demo.tar.gz*.

  1. Download and extract the demo project.
  2. You must specify the installation directory for oss-c-sdk-demo-specified-installation: `/home/your/oss/csdk`. You do not need to specify installation directories for other demo projects because they are automatically installed based on the OSS C SDK and other dependent third-party libraries.
  3. Replace the OSS_ENDPOINT, ACCESS_KEY_ID, ACCESS_KEY_SECRET, and BUCKET_NAME in the demo with valid values. Use `LD_LIBRARY_PATH` to specify the directory of the OSS C SDK and the dynamic libraries of the dependent libraries if they are not in the system directory.
  4. Enter the directory of the project (oss-c-sdk-demo-xxx), run the `make` command to compile the demo project. Then, run the `./main` command to run the executable program. If the project needs to be re-compiled, run the `make clean`

  command first before compiling.*https://yq.aliyun.com/articles/60924*

- Windows demo

  Windows-based demo: *aliyun-oss-c-sdk-sample.zip*. You can download more demo projects from *GitHub*.

  1. Download and extract the demo project.
  2. Use Visual Studio to open the demo project, and then replace the following parameters in the oss_config.c with 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 demo project.

     Note:
     For detailed steps in running demo pojects with Visual Studio, see related documents. *https://yq.aliyun.com/articles/57947*

Create a bucket

A bucket is a global namespace on OSS used to store objects. It is equivalent to a data container.

The following code creates 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
);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 the network and memory. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize 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 a char* to the bucket. */
    aos_str_set(&bucket, bucket_name);
    resp_status = oss_create_bucket(oss_client_options, &bucket,
oss_acl, &resp_headers);
    /* Identify whether the request succeeds */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
```

```
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Upload an object

You can run the following code to upload an object 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>";
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
);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) != AOSE_OK) { = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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 the char* type data to the aos_list_t type data. */
    aos_list_init(&buffer);
    content = aos_buf_pack(oss_client_options->pool, data, strlen(data
));
    aos_list_add_tail(&content->node, &buffer);
    /* Upload an object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &
bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the upload request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("put file succeeded\n");
    } else {
        printf("put file failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Download an object

You can run the following code to download a specified 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 *download_filename = "<yourDownloadedFilename>";
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
);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
```

```
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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 an object. */
    resp_status = oss_get_object_to_file(oss_client_options, &bucket,
 &object, headers, params, &file, &resp_headers);
    /* Determine whether the download request is successful. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List objects in a bucket

You can run the following code to list objects stored 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 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
);
```

```
    /* Determine whether the CNAME is used. 0 indicates the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool = NULL;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes
 global configuration information, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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 list request is successful. */
    if (! aos_status_is_ok(resp_status))
    {
        printf("list object failed\n");
    }
    else{
        /* Print 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
```

```
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete an object.

You can run the following code 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 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
);
    /* Determine whether the CNAME is used. 0 indicates the CNAME is
not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool = NULL;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes
 global configuration information, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options = NULL;
    /* Allocate memory in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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 an object. */
    resp_status = oss_delete_object(oss_client_options, &bucket, &
object, &resp_headers);
```

```
    /* Determine whether the delete request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeeded\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.4 Initialization

To use OSS C SDK, you must initialize request options (oss_request_options_t) and specify an endpoint. For more information about endpoint, see *Regions and endpoints* and *Bind a custom domain name*.

Use the OSS domain name to initialize request options.

You can run the following code to initialize request options with the OSS domain name:

```
#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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 first 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);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
```

```
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

For the configuration of request options, see related documents. *http://bbs.aliyun.com/read/261421.html?spm=5176.bbsr248504.0.0.NoFhRX*

Use a custom domain name to initialize request options.

You can run the following code to initialize request options with a domain name:

```
#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 the aos_string_t 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
);
    /* 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);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Use STS to initialize request options

You can run the following code to initialize request options with 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 = "<yourSecurityToken>";
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 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
);
    /* Configure STS */
    aos_str_set(&options->config->sts_token, sts_token);
    /* Determine whether the CNAME is used. 0 indicates that the 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 first 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);
    /* Logical code, which is not listed here. */
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Set the timeout value.

You can run the following code to set the timeout value:

```
#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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
```

```
    /* Used to configure network parameters, such as timeout */
    options->ctl = aos_http_controller_create(options->pool, 0);
    /* Set the connection timeout value, which is 10 seconds by
default. */
    options->ctl->options->connect_timeout = 10;
    /* Set the DNS timeout value, which is 60 seconds by default. */
    options->ctl->options->dns_cache_timeout = 60;
    /* Set the request timeout value:
    Set the value of speed_limit to specify the minimum speed that can
 be accepted, which is 1,024 Bytes/s by default.
    Set the value of speed_time to specify the maximum time period
that can be accepted, which is 15 seconds by default.
    The following code indicates that if the trasmission speed is
continuously lower than 1,024 Bytes/s for 15 seconds, the request is
time-out. */
    options->ctl->options->speed_limit = 1024;
    options->ctl->options->speed_time = 15;
}
```

4.5 Bucket

A bucket serves as a container that stores objects. Objects belong to a bucket.

Create a bucket

Run the following code 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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 a char* to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Create a bucket. */
    resp_status = oss_create_bucket(oss_client_options, &bucket,
oss_acl, &resp_headers);
    /* Determine whether the request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

For more information about bucket naming rules, see naming conventions in *Basic concepts*. You can specify the ACL and the storage class when you create a bucket, as shown in *Bucket-level permissions* and *Storage class*.

You can run the following code to create a bucket with a specified ACL:

```
    /* Create a bucket and set the bucket ACL to public read (which is
 private by default). */
    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 ");
    }
```

You can run the following code to creates a bucket of the archive class:

```
    resp_status = oss_create_bucket_with_storage_class(oss_client
_options, &bucket, OSS_ACL_PRIVATE, OSS_STORAGE_CLASS_ARCHIVE);
    /* Determine whether the request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
```

```
    }
```

You can run the followign code to create a bucket of the Infrequent Access class:

```
    resp_status = oss_create_bucket_with_storage_class(oss_client
_options, &bucket, OSS_ACL_PRIVATE, OSS_STORAGE_CLASS_IA);
    /* Determine whether the request is successful. */
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
```

List buckets

Run the following code to list all buckets:

```
#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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
```

```
    init_options(oss_client_options);
    /* Initialization 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 all 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");
    }
    /* Print bucket information. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Configure an ACL for a bucket

The ACL of a bucket includes the following permissions.

Permission	Description	Value
Private	The bucket owner and the authorized users can read and write objects in the bucket. Other users cannot perform any operation on the objects.	OSS_ACL_PRIVATE
Public read	The bucket owner and the authorized users can read and write objects in the bucket . Other users can only read the objects in the bucket. Authorize this permission with caution.	OSS_ACL_PUBLIC_READ
Public read-write	All users can read and write objects in the bucket. Authorize this permission with caution.	OSS_ACL_PUBLIC_READ_ WRITE

For more information about the ACL, see *Access control* in the developer guide.

You can run the following code to configure an ACL 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign a char* to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Set the bucket ACL to 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Obtain an ACL for a bucket

Run the following code to obtain the ACL 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
```

```
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t oss_acl;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign a char* to the aos_string_t bucket */ */
    aos_str_set(&bucket, bucket_name);
    /* Obtain the ACL for a 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Obtain the region of a bucket

Run the following code to obtain the region (or location) 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
```

```
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t oss_location;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign a char* to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Obtain the region of the bucket. */
    resp_status = oss_get_bucket_location(oss_client_options, &bucket,
 &oss_location, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket location succeeded : %s \n", oss_location.
data);
    } else {
        printf("get bucket location failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Obtain bucket information

Run the following code to obtain bucket 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>";
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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
```

```
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    oss_bucket_info_t bucket_info;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign a char* to the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    /* Obtain the bucket inforamtion. */
    resp_status = oss_get_bucket_info(oss_client_options, &bucket, &
bucket_info, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get bucket info succeeded\n");
        printf("location: %s\n", bucket_info.location.data);
        printf("owner_id: %s\n", bucket_info.owner_id.data);
        printf("owner_name: %s\n", bucket_info.owner_name.data);
        printf("created_date: %s\n", bucket_info.created_date.data);
        printf("location: %s\n", bucket_info.location.data);
    } else {
        printf("get bucket info failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

Delete a bucket

Before a bucket is deleted, ensure that all objects in the bucket and fragments that are generated from multipart upload are deleted.

> **Note:**
> To delete the fragments that are generated from multipart upload, use oss_list_upload_part to list all fragments, and then use oss_abort_multipart_upload to delete the fragments. For more information, see *Multipart upload*.

Run the following code 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    /* Assign a char* 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.6 Upload objects

4.6.1 Overview

In OSS, an object is the basic unit for data operations. OSS C SDK provides the following file upload methods:

- *Simple upload*: supports the upload of a file up to 5 GB in size. It includes streaming upload and file upload.
- *Append upload*: supports the upload of a file up to 5 GB in size.
- *Resumable upload*: supports the upload of a file up to 48.8 TB in size. It applies to the upload of large files. It also supports concurrent upload and you can define the size of each part.
- *Multipart upload*: supports the upload of a file up to 48.8 TB in size. It applies to the upload of large files.

Note:

For more information about the usage scenarios of each upload method, see “Upload files” in OSS Developer Guide.

During the file upload, you can set *Object Meta*, and check the upload progress in *upload progress bar*. After you complete the file upload, you can perform *upload callback*.

4.6.2 Simple upload

For the complete code of simple upload, see *GitHub*

Upload data from memory to OSS

You can run the following code to upload data from memory 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 *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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 a file. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &
bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the upload is successful. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Upload a local file to OSS

You can run the following code 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 a file. */
    resp_status = oss_put_object_from_file(oss_client_options, &bucket
, &object, &file, headers, &resp_headers);
    /* Determine whether the upload is successful. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.6.3 Append upload

You cannot perform copyObject for appended objects.

Append data from memory

Run the following code to append data from memories

```
#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
);
```

```
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new pool does not inherit any other memory pool.
 */
    aos_pool_create(&p, 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 file. */
    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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Append data from a local file

Run the following code 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new pool does not inherit any other memory pool.
 */
    aos_pool_create(&p, 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 file. */
    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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

If the file already exists, the following may occur:

- If the file cannot be uploaded with append upload, an ObjectNotAppendable exception occurs.
- If the file can be uploaded with append upload, the file is upload by the Put Object method and overwrites the existing object. The object type is changed to Normal Object. After the Head Object operation is performed, the system returns x-oss-object-type, which indicates the type of the object.

4.6.4 Resumable upload

In resumable upload, the object is divided into multiple parts, which are then uploaded separately. After all the parts are uploaded, they are merged into a complete object to finish the upload.

Current upload progress is recorded into a checkpoint file during the upload. If a part fails to be uploaded during the process, the object is uploaded from the part recorded in the checkpoint file when the upload restarts, that is, the resumable upload. After the upload is complete, the checkpoint is deleted.

You can use the `oss_resumable_clt_params_t` method to achieve resumable upload.

Common parameters used in this method are listed as follows:

Parameter	Description
thread_num	The number of concurrent threads, which is 1 by default.
enable_checkpoint	Whether to enable resumable upload, which is disabled by default.
partSize	The part size, ranging from 100 KB to 5 GB.
checkpoint_path	The path that the checkpoint file is stored, which is {upload_file_path}.cp by default.

You can run the following code 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 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
);
    /* Deternube whether the CNAME is used. 0 indicates that it is not
 used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
```

```
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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);
    /* 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.6.5 Multipart upload

To enable multipart upload, perform the following steps:

1. Initiate a multipart upload event.

   You can call ossClient.initiateMultipartUpload to return the globally unique uploadId created in OSS.

2. Upload parts.

   You can call ossClient.uploadPart to upload part data.

   Note:

   - For parts with a same uploadId, parts are sequenced by their part numbers. If you have uploaded a part and use the same part number to upload another part, the later part will replace the former part.
   - OSS places the MD5 value of part data in ETag and returns the MD5 value to the user.

- SDK automatically configures Content-MD5. OSS calculates the MD5 value of uploaded data and compares it with the MDS value calculated by SDK. If the two values vary, the error code of InvalidDigest is returned.

3. Complete multipart upload.

   After you have uploaded all parts, call partossClient.completeMultipartUpload to combine these parts into a complete object.

The following code is used as a complete example that describes the process of 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)
{
    ptions->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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter mainly includes
 global configuration nformation, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Allocate memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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;
    /* Initialize multipart upload and obtain an upload ID (upload_id
). */
    resp_status = oss_init_multipart_upload(oss_client_options, &
bucket, &object, &upload_id, headers, &resp_headers);
    /* Determines whether the slice upload Initialization is
successful. */
    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)) {
            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;
    /* Obtain 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 the part list is successfully obtained. */
    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_p
art_list);
    }
    /* Complete multipart upload. */
    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 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

You must provide all valid PartETags when you call oss_complete_multipart_upload to obtain uploaded parts. The ETags can be obtained in two methods: 1. The ETag of a part is included in the returned result when the part is uploaded. 2. You can call oss_list_upload_part to obtain the ETag of an uploaded part. In the example, the second method is used.

Cancel a multipart upload event

Run the following code to cancel a multipart upload event:

```
#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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not

inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Initialize multipart upload and obtain an upload ID (upload_id
). */
    resp_status = oss_init_multipart_upload(oss_client_options, &
bucket, &object, &upload_id, headers, &resp_headers);
    /* Determine whether the multipart upload is initialized
successfully. */
    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. */
    resp_status = oss_abort_multipart_upload(oss_client_options, &
bucket, &object, &upload_id, &resp_headers);
    /* Determine whether the multipart upload is canceled successfully
. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

If you cancel a multipart upload event, you are not allowed to perform any other operations with this upload_id anymore. The uploaded parts will be deleted.

4.6.6 Upload progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of upload progress bar, see *GitHub*.

The following code is used as an example to describe how to view progress information with PutObject:

```
#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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
```

```
    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 the aos_http_io_initialize method in main() to initialize
global resources, such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Upload with a progress bar displayed. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

oss_put_object_from_file and oss_put_object_from_buffer do not support progress bars.

4.6.7 Upload callback

For the complete code of upload callback, see *GitHub*.

Run the following code for 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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. */
    char *callback =  "{"
        "\"callbackUrl\":\"http://oss-demo.aliyuncs.com:23450\","
        "\"callbackHost\":\"oss-cn-hangzhou.aliyuncs.com\","
        "\"callbackBody\":\"bucket=${bucket}&object=${object}&size=${
size}&mimeType=${mimeType}\","
        "\"callbackBodyType\":\"application/x-www-form-urlencoded\""
        "}";
    /* Call the aos_http_io_initialize method in main() to initialize
global resources, such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Initialize 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 into 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);
    /* Start 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 in 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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(p);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

For more information about upload callback, see *Upload callback*. For detailed debugging methods and troubleshooting information, see *Upload callback error and exclusion*.

4.7 Download objects

4.7.1 Overview

For the complete code of object download, see *GitHub*.

OSS C SDK provides the following download methods:

- *Download to your local file*
- *Download to local memory*

- *Range download*
- *Resumable download*

You can view the download progress shown in the *download progress bar*.

4.7.2 Download an object to a local file

For the complete code of downloading OSS object, see *GitHub*.

You can run the following code to download the specified OSS 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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization 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. A new local file is created if the
specified file does not exist. If the specified local file exists, it
is overwritten. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

Compared to the 1.0.0 version, the params parameter is added to the oss_get_object_to_file interface, and the value of the headers and params parameters can be NULL.

4.7.3 Download to your local memory

For the complete code of object download, see *GitHub*.

Run the following code 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 an object to local memory. */
    resp_status = oss_get_object_to_buffer(oss_client_options, &bucket
, &object,
                                    headers, params, &buffer, &resp_heade
rs);
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

4.7.4 Range download

If you only need part of the data in an object, you can use range downloads to download specified content.

Run the following code for 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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 download range, which ranges from the 20th byte to the
100th byte. */
    apr_table_set(headers, "Range", "bytes=20-100");
    /* Download the object to 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 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.7.5 Resumable download

Large-sized objects may fail to be downloaded because of unstable network or other exceptions, and the entire object needs to be downloaded again. However, the object may still fail to be downloaded after multiple attempts. Therefore, OSS provides resumable download.

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.

You can use `oss_resumable_clt_params_t` to enable resumable download. `oss_resumable_clt_params_t` contains the following parameters:

Parameter	Description
thread_num	The number of concurrent threads, which is 1 by default.
enable_checkpoint	Determines whether to enable resumable download, which is disabled by default.
partSize	Specifies the size of a part. The value can be between 1 byte to 5 GB and the unit is byte.
checkpoint_path	The path where the checkpoint file is located, which is under {upload_file_path}.cp by default.

This `checkpoint` file stores information about the download progress.If you fail to download a part, the download will continue based on the progress recorded in the `checkpoint` file. After the object is downloaded, the `checkpoint` file will be deleted.

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 the aos_string_t 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
```

```
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Download the object using resumable download. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.7.6 Download progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of download progress bar, see *GitHub*.

The following code is used as an example to describe how to use the download progress bar.

```
#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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 the aos_http_io_initialize method in main() to initialize
global resources, such as networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Upload with a progress bar displayed. */
    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");
    }
    /* Download with a progress bar displayed. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

oss_get_object_from_file and oss_get_object_from_buffer do not support progress bars.

4.8 Manage objects

4.8.1 Overview

You can perform the following operations on objects in a bucket with APIs:

- *Manage ACL for an object*
- *Manage Object Meta*
- *List objects*
- *Copy objects*
- *Delete objects*
- *Restore an archive object*
- *Manage a symbolic link*

4.8.2 Manage ACL for an object

The following table describes the permissions included in the Access Control List (ACL) for an object.

Permission	Description	Value
default	The ACL of an object is the same with that of its bucket.	OSS_ACL_DEFAULT
Private	Only the object owner and authorized users can read and write the object.	OSS_ACL_PRIVATE
Public read	Only the object owner and authorized users can read and write the object. Other users can only read the object	OSS_ACL_PUBLIC_READ

Permission	Description	Value
	. Authorize this permission with caution.	
Public read-write	All users can read and write the object. Authorize this permission with caution.	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.

Configure an ACL for an object

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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
```

```
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 an ACL for an object. */
    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");
    }
    /* Obtain the ACL for an 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.8.3 Manage Object Meta

Object Meta includes HTTP headers and user-defined Object Meta. For more information, see *Object Meta* in OSS Developer Guide.

Obtain Object Meta

After uploading files to OSS or before reading objects from OSS, you can use the `oss_get_object_meta` interface to obtain some of meta information about the objects, such as the length and type. Run the following code to configure and obtain Object Meta:

```
#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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 custom Object Meta. */
    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);
    /* Set object permissions and upload the object from the cache. */
    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");
    }
    /* Obtain the object permission. */
    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_AUTHOR
IZATION);
        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_le
ngth);
    } else {
        printf("req:%s, get object meta failed\n", resp_status->req_id
);
    }
    /* Release the memory pool, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

For detailed information about HTTP header, see *RFC2616*.

4.8.4 List objects

For the complete code for listing objects, see *GitHub*.

OSS objects are listed alphabetically. You can use `oss_list_object` to list all objects in a bucket. The following table describes common parameters used to list objects.

Parameter	Description
delimiter	Specifies a delimiter used to group objects. All those objects whose names contain the specified prefix and after which the delimiter occurs for the first time, act as a group of elements, that is, commonPrefixes.
prefix	Specifies the prefix that the listed objects must contain.
max_ret	Specifies the maximum number of objects that can be listed. The default value is 1000.

Parameter	Description
marker	Specifies the initial object that is listed.

List all objects

You can run the following code to list all objects in the current 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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Note:

By default, if a bucket contains more than 1,000 objects, only the first 1,000 objects are returned and the truncated parameter in the returned results is true. Additonally, the next_marker is returned to serve as the start point of next read. To adjust the number of returned objects, you can modify the max_ret parameter or use the marker parameter for multiple reads.

List specified number of objects

You can run the following code to list 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);
    /* aos_str_set uses a char* string to initialize the aos_string_t
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
);
    /* Determine whether the CNAME is used.*/
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.*/
    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 memories.*/
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pool. */ */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes
 global configuration information, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Memories for options are allocated by the memory pool and are
released after the memory pool is released. Therefore, you do not need
 to release the memory for options individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 listed 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 a request is executed, which
means that memories allocated for all resources during the request are
 released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize
method to release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

List objects with specified prefix

You can run the following code to list objects with 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);
    /* aos_str_set uses 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
);
    /* Determine whether the CNAME is used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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, including networks and memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
 */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes
 global configuration information, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
```

```
    /* Memories for options are allocated by the memory pool and are
released after the memory pool is released. Therefore, you do not need
 to release the memory for options individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 objects with 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 a request is executed, which
means that memories allocated for all resources during the request are
 released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize
method to release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

Specifies the initial object that is listed.

The value of marker is the name of the initial object. You can run the following code to list objects after marker in alphabetic order.

```
#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);
    /* aos_str_set uses a char* string to initialize the aos_string_t
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
);
    /* Determine whether the CNAME is used.*/
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout.*/
    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 network and memories.*/
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly includes
 global configuration information, such as endpoint, access_key_id,
acces_key_secret, is_cname, and curl. */
    oss_request_options_t *oss_client_options;
    /* Memories for options are allocated by the memory pool and are
released after the memory pool is released. Therefore, you do not need
 to release the memory for options individually. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 = "";
    Specifies the initial object that is listed. */
    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 a request is executed, which
means that memories allocated for all resources during the request are
 released. */
    aos_pool_destroy(pool);
    /* Before the project ends, call the aos_http_io_deinitialize
method to release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.8.5 Delete objects

Warning:

Delete objects with caution because deleted objects cannot be recovered.

For the complete code of deleting objects, see *GitHub*.

Delete an object

Run the following code 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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 the aos_string_t bucket. */
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&object, object_name);
    /* Delete an object. */
    resp_status = oss_delete_object(oss_client_options, &bucket, &
object, &resp_headers);
    /* Determine whether the object is deleted successfully. */
    if (aos_status_is_ok(resp_status)) {
        printf("delete object succeed\n");
    } else {
        printf("delete object failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

Delete multiple objects

Run the following code to delete multiple objects simultaneously:

```
#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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Construct the list of objects to be deleted. */
    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 objects in the list. `is_quiet` is used to determine
whether the result of deletion is returned. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Delete multiple objects with a specified prefix

You can delete a directory by deleting multiple objects with a specified prefix. For example, you can delete the dir directory by deleting objects with the prefix **dir/**. Run the following code to delete multiple objects with a specified prefix simultaneously:

```
#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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
```

```
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 objects with a 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Warning:

If the prefix is an empty string or null, all objects in the bucket will be deleted. Perform this operation with caution.

4.8.6 Copy objects

Copy a small-sized object

You must note the following limits before copy a small-sized object:

- The user must have the read and write permission on the source object.
- You cannot copy an object from a region to another region. For example, you cannot copy an object from a bucket in Hangzhou region to a bucket in Qingdao.
- The object size must be smaller than 1 GB.

Run the following code to copy a small-sized 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 *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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 an 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Copy a large-sized object

For objects larger than 1 GB, we recommend that you should use **oss_upload_part_copy** to perform multipart copy (UploadPartCopy). Run the following code for 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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
```

```
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 of the object. */
    upload_part_copy_params1 = oss_create_upload_part_copy_params(pool
);
    aos_str_set(&upload_part_copy_params1->source_bucket, source_buc
ket_name);
    aos_str_set(&upload_part_copy_params1->source_object, source_obj
ect_name);
    aos_str_set(&upload_part_copy_params1->dest_bucket, dest_bucke
t_name);
    aos_str_set(&upload_part_copy_params1->dest_object, dest_objec
t_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_par
t_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 of the object. */
    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_buc
ket_name);
    aos_str_set(&upload_part_copy_params2->source_object, source_obj
ect_name);
    aos_str_set(&upload_part_copy_params2->dest_bucket, dest_bucke
t_name);
    aos_str_set(&upload_part_copy_params2->dest_object, dest_objec
t_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_par
t_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");
    }
    /* Lists the 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_bucke
t, &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. */
    resp_status = oss_complete_multipart_upload(oss_client_options, &
dest_bucket, &dest_object, &upload_id, &complete_part_list, headers, &
complete_resp_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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.8.7 Restore an archive object

You must restore an archive object before you read it. Do not call restoreObject for non-archive objects.

The state conversion process of an archive object is as follows:

- An archive object is in the frozen state.
- After you submit it for restoration, the server restores the object. The object is in the restoring state.
- You can read the object after it is restored. The restored state of the object lasts one day by default. You can prolong this period to a maximum of seven days. Once this period expires, the object returns to the frozen state.

Run the following code to restore 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 *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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
```

```
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_list_t buffer;
    oss_acl_e oss_acl = OSS_ACL_PRIVATE;
    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);
    headers = aos_table_make(pool, 0);
    /* Create a bucket of the archive storage class. */
    resp_status = oss_create_bucket_with_storage_class(oss_client
_options, &bucket, oss_acl, OSS_STORAGE_CLASS_ARCHIVE, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("create bucket succeeded\n");
    } else {
        printf("create bucket failed\n");
    }
    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 a file. */
    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 succeeded\n");
    } else {
        printf("put object from buffer failed\n");
    }
    /* Restore the object. */
    do {
        headers = aos_table_make(pool, 0);
        resp_status = oss_restore_object(oss_client_options, &bucket,
 &object, headers, &resp_headers);
        printf("restore object resp_status->code: %d \n", resp_status-
>code);
        if (resp_status->code ! = 409) {
            break;
        } else {
            printf("restore object is already in progress, resp_status
->code: %d \n", resp_status->code);
            apr_sleep(5000);
        }
    } while (1);
```

```
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

For more information about the archive storage classes, see *Introduction to storage classes*. For more information about related status code, see *RestoreObject*.

4.8.8 Manage a symbolic link

Create a symbolic link

A symbolic link is a special object that maps to an object similar to a shortcut used in Windows.

You can configure user-defined object meta for symbolic links.

Run the following code to create a symbolic link:

```
#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 *link_object_name = "<yourLinkObjectName>";
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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t object;
    aos_string_t sym_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);
    aos_str_set(&sym_object, link_object_name);
    resp_status = oss_put_symlink(oss_client_options, &bucket, &
sym_object, &object, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("put symlink succeeded\n");
    } else {
        printf("put symlink failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Obtain object content for a symbolic link

To obtain a symbolic link, you must have the read permission on it. Run the following code to obtain the object content for a symbolic link:

```
#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 *link_object_name = "<yourLinkObjectName>";
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 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
```

```
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_string_t link_object;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    char *target_object_name = NULL;
    aos_str_set(&bucket, bucket_name);
    aos_str_set(&link_object, link_object_name);
    resp_status = oss_get_symlink(oss_client_options, &bucket, &
link_object, &resp_headers);
    if (aos_status_is_ok(resp_status)) {
        printf("get symlink succeeded\n");
        target_object_name = (char*)(apr_table_get(resp_headers,
OSS_CANNONICALIZED_HEADER_SYMLINK));
        printf("target_object_name: %s \n", target_object_name);
    } else {
        printf("get symlink failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.9 Authorized access

Use STS for temporary access authorization

OSS supports Alibaba Cloud Security Token Service (STS) for temporary access authorization. STS is a web service that provides a temporary access token to a cloud computing user. Through the STS, you can assign a third-party application or a RAM user (you can manage the user ID) an access credential with a custom validity period and permissions. For more information about STS, see *STS introduction*.

STS advantages:

- Your long-term key (AccessKey) is not exposed to a third-party application. You only need to generate an access token and send the access token to the third-party application. You can customize access permissions and the validity of this token.
- You do not need to keep track of permission revocation issues. The access token automatically becomes invalid when it expires.

For more information about the process of access to OSS with STS, see *RAM and STS scenario practices* in OSS Developer Guide.

Create a signature request with STS

Run the following code to create a signature request with 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 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);
 /* Determine whether the CNAME is used. 0 indicates that the CNAME is
 not used. */
 options->config->is_cname = 0;
 /* Used to configure network parameters, such as timeout. */
 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 memories. */
 if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
  exit(1);
 }

 /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
 aos_pool_t *pool;
 /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
 aos_pool_create(&pool, NULL);
 /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
 oss_client_options = oss_request_options_create(pool);
 /* Use oss_client_options to initialize client options. */
 init_options(oss_client_options);

    /*  Initialization 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 an object. */
    resp_status = oss_put_object_from_buffer(oss_client_options, &
bucket, &object, &buffer, headers, &resp_headers);
    /* Determine whether the upload is successful. */
    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, that is, memories allocated to resources
during the request. */
 aos_pool_destroy(pool);

 /* Release allocated global resources. */
 aos_http_io_deinitialize();

 return 0;
}
```

Sign a URL to authorize temporary access

You can provide a signed URL to a visitor for temporary access. When you sign a URL, you can specify the expiration time for a URL to restrict the period of access from visitors.

- Use a signed URL to upload an object

  Run the following code to upload an object with a signed URL:

```
#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 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);
    /* Determine whether the CNAME is used. 0 indicates that the
CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library.
 */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 the signed URL. */
    url_str = oss_gen_signed_url(oss_client_options, &bucket, &
object, expire_time, req);
    aos_str_set(&url, url_str);
```

```
    printf("Temporary upload URL: %s\n", url_str);
    /* Use the signed URL to upload an 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Use a signed URL to download an object

  Run the following code to upload an object with a signed URL:

```
#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 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);
    /* Determine whether the CNAME is used. 0 indicates that the
CNAME is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library.
 */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&p, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /*  Initialization 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 the 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 an 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.10 Manage lifecycle rules

You can use OSS to configure lifecycle rules to reduce storage costs. The lifecycle management rules can be used for automatic deletion of expired objects and fragments, or storage class conversion to Infrequent Access for objects that will expire soon. Each rule includes:

- Rule ID: It identifies a rule. Ensure that each rule ID in a bucket is unique.
- Policy: You can configure a policy by using the following configuration methods. Only one method can be configured for a bucket.

  - Configure by Prefix: You can create multiple rules with this method. Ensure that each prefix is unique.
  - Configure for Entire Bucket: You can configure only one rule with this method.
- Expired: You can configure the following configuration methods:
  - Set by Number of Days: It specifies the number of days from the last modification time of objects.
  - Set by Date: It specifies the date prior to which objects are created. If objects are created prior to the date, these objects are deleted.
- Status: This lifecyle rule takes effect or not.

The parts uploaded by uploadPart method also support lifecycle rules. The last modification time of an object is the time the multipart upload event is initiated.

For more information about lifecycle management, see *Manage object lifecycle*.

Configure lifecycle rules

Run the following code to configure lifecycle 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
```

```
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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 a lifecycle rule for the bucket. */
    aos_str_set(&bucket, bucket_name);
    aos_list_init(&lifecycle_rule_list);
    /* Specify the expiration duration. */
    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 a lifecycle rule. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

View lifecycle rules

Run the following code to view lifecycle 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Obtain and print the lifecycle of 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Clear lifecycle rules

Run the following code to clear lifecycle 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Used to configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
```

```
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Use oss_client_options to initialize client options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete bucket lifecycle rules. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.11 Image processing

Image Processing (IMG) is a massive, secure, cost-effective and highly reliable image processing service. After source images are uploaded to OSS, you can process images on any Internet device at any anytime, from anywhere through simple RESTful APIs.

For more information about IMG, see *Image Processing*.

For the complete code of IMG, see *GitHub*.

Basic features

OSS offers the following IMG features:

- *Retrieve dominant image tones*
- *Format conversion*
- *Resize images*
- *Incircle*
- *Adaptive orientation*

- *Brightness*
- *Add watermarks*: allows you to add images, texts, or image-text watermarks to another image.
- *Image processing*
- *Image processing access rules*: calls multiple IMG functions.

Usage

IMG uses standard HTTP GET. You can configure IMG parameters in QueryString of a URL.

If the ACL of an image object is private read and write, only authorized users are allowed for access.

- Anonymous access

  You can use the following format in level-3 domains to access a processed image:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=image/<yourAction>,<yourParamValue>
```

Parameter	Description
bucket	Specifies the name of a bucket.
endpoint	Specifies endpoint used to access a region.
object	Specifies the name of an image object.
image	Specifies the reserved identifier for IMG.
action	Specifies the operations on an image, such as scaling, cropping, and rotating.
param	Specifies the parameter that indicates the operation on an image.

  — Basic operations

  For example, scale an image to a width of 100 px. Adjust the height based on the ratio.

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

  — Customized image styles

  You can use the following format in level-3 domains to access a processed image:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
```

  ■ style: specifies a reserved identifier of a customized image style.

- yourStyleName: specifies the name of a custom image style. It is the name specified in the rule that is created on the OSS console.

Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
process=style/oss-pic-style-w-100
```

— Cascade operations

Cascade operations allow you to perform multiple operations on an image sequentially.

```
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
```

— HTTPS access

IMG supports access through HTTPS. Example:

```
https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
process=image/resize,w_100
```

- Authorized access

Authorized access allows you to customize an image style, HTTPS access, and cascade operations.

Use the following code to generate a signed URL for 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);
    /* Determine whether the CNAME is used. The value 0 indicates
that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t
. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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);
    /* Start image processing. */
    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 expiration time (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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Access with SDK

  You can use SDK to access and process an image object.

  SDK allows you to customize image styles, HTTPS access, and cascade operations.

- Basic operations

Run the following code to perform basic operations on an image:

```
#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);
    /* Determine whether the CNAME is used. The value 0 indicates
that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to
apr_pool_t. The implementation code is included in the apr library
. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 processed images 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 processed images 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 processed images 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 processed images 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.
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/watermark,text_SGVsb
G8g5Zu-54mH5pyN5YqhIQ");
    /* Download processed images 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 processed images 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 processed images 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, that is, the memories allocated to
 resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Customize an image style

  Run the following code to customize an image style:

```
#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);
    /* Determine whether the CNAME is used. The value 0 indicates
that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
```

```
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to
apr_pool_t. The implementation code is included in the apr library
. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 processed images 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, that is, the memories allocated to
 resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

- Cascade operations

  Run the following code to perform cascade operations on an image:

```
#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);
    /* Determine whether the CNAME is used. The value 0 indicates
that the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to
apr_pool_t. The implementation code is included in the apr library
. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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 operations. */
    params = aos_table_make(pool, 1);
    apr_table_set(params, OSS_PROCESS, "image/resize,m_fixed,w_100
,h_100");
    /* Download processed images 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, that is, the memories allocated to
 resources during the request. */
    aos_pool_destroy(pool);
```

```
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

IMG tools

You can use the IMG viewer *ImageStyleViever* to directly view the IMG result.

4.12 CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to another region. OSS provides CORS APIs for convenient cross-origin access control.

For more information, see *Cross-origin resource sharing* and *PutBucketcors* in OSS Developer Guide.

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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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 rule. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

Obtain 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Obtain 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, that is, memories allocated to
resources during 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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete all 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.13 Set logging

You can enable access logs to record bucket access to log files, which are stored in a specified bucket.

The log file format is as follows:

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

For more information about access log files, see *Set access logging*.

Enable access 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    options->ctl = aos_http_controller_create(options->pool, 0);
}
int main(int argc, char *argv[])
{
    /* Call the maid method at the program portal to initialize global
 resources such as network, memory, and so on. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_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 bucket access logging. */
    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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

View access 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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 bucket access logging configurations. */
    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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Disable access 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Disable access 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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.14 Anti-leech

This topic describes how to use anti-leech.

To prevent your data on OSS from being leeched, OSS supports anti-leeching through the referer field settings in the HTTP header, including the following parameters:

- Referer whitelist: Used to allow access only for specified domains to OSS data.
- Empty referer: Determines whether the referer can be empty. If it is not allowed, only requests with the referer filed in their HTTP or HTTPS headers can access OSS data.

For more information about anti-leaching, see *Anti-leeching settings*.

Configure the referer whitelist

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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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 the referer field. The referer field allows question marks
 (?) and asterisks (*) for wildcard use. */
    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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

Obtain a referer whiltelist

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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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);
    /* Obtain the referer list for a 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_co
nfig.referer_list, node) {
            printf("get referer %s\n", referer->referer.data);
        }
    } else {
        printf("get bucket referer failed\n");
    }
    /* Release the memory pool, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

Clear a referer whitelist

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
);
    /* Determine whether the CNAME is used. 0 indicates that the CNAME
 is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, which is equivalent to
apr_pool_t. The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a new memory pool. The second parameter is NULL,
indicating that it does not inherit from any other memory pools. */
    aos_pool_create(&pool, NULL);
    /* Create and initialize options. This parameter mainly 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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 a referer whitelist directly. To clear a
referer whitelist, you need to create the rule that allows an empty
referer field and replace the original rule with the new 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, that is, memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
}
```

4.15 Static website hosting

You can set your bucket to static website hosting mode. After the configuration takes effect, you can access this static website with the bucket domain and be redirected to a specified index page or error page.

For more information about static website hosting, see *Configure static website hosting*.

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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new pool does not inherit any other memory pool.
 */
    aos_pool_create(&p, 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize 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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialization 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 static web 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, that is, the memories allocated to
resources during 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
);
    /* Determine whether the CNAME is used. The value 0 indicates that
 the CNAME is not used. */
    options->config->is_cname = 0;
    /* Configure network parameters, such as timeout. */
    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 memories. */
    if (aos_http_io_initialize(NULL, 0) ! = AOSE_OK) {
        exit(1);
    }
    /* Memory pool used to manage memories, equivalent to apr_pool_t.
The implementation code is included in the apr library. */
    aos_pool_t *pool;
    /* Re-create a memory pool. The second parameter is NULL,
indicating that the new 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 the memories in the memory pool to options. */
    oss_client_options = oss_request_options_create(pool);
    /* Initialize oss_client_options. */
    init_options(oss_client_options);
    /* Initialize parameters. */
    aos_string_t bucket;
    aos_table_t *resp_headers = NULL;
    aos_status_t *resp_status = NULL;
    aos_str_set(&bucket, bucket_name);
    /* Delete 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, that is, the memories allocated to
resources during the request. */
    aos_pool_destroy(pool);
    /* Release the allocated global resources. */
    aos_http_io_deinitialize();
    return 0;
```

```
}
```

4.16 Error handling

Exception handling

If a request error occurs in the use of OSS C SDK, the corresponding error message is output in aos_status_s. The attributes of aos_status_s include:

Error code	Description	Type
code	The HTTP status code of the error request	Integer
error_code	Error code returned by OSS	Character string
error_msg	Error message returned by OSS	Character string
req_id	The UUID that identifies the request. You can provide the req_id to OSS development engineers for further help.	Character string

Timeout handling

If the code in the returned aos_status_t is not “2XX”, and the error_code is -992 or -995, it indicates a connection timeout error or a request timeout error occurs. You can try again.

You can use the aos_should_retry(aos_status_t *) in the aos_status.h to determine whether the returned error code requires a retry. If the returned value is 1, you must try again.

Common error codes

Error code	Description	HTTP status code
AccessDenied	Access denied	403
BucketAlreadyExists	The bucket already exists.	409
BucketNotEmpty	The bucket is not empty.	409
EntityTooLarge	The entity size exceeds the maximum limit.	400
EntityTooSmall	The entity size is below the minimum limit.	400
FileGroupTooLarge	The total file group size exceeds the maximum limit.	400

Error code	Description	HTTP status code
FilePartNotExist	The file part does not exist.	400
FilePartStale	The file part has expired.	400
InvalidArgument	The parameter format is invalid .	400
InvalidAccessKeyId	The AccessKeyId does not exist.	403
InvalidBucketName	The bucket name is invalid.	400
InvalidDigest	The digest is invalid.	400
InvalidObjectName	The object name is invalid.	400
InvalidPart	The part is invalid.	400
InvalidPartOrder	The part order is invalid.	400
InvalidTargetBucketForLogging	The buckets that store log files are invalid.	400
InternalError	Internal OSS error	500
MalformedXML	The XML format is invalid.	400
MethodNotAllowed	The method is not supported.	405
MissingArgument	Parameters are not configured.	411
MissingContentLength	The content length is not configured.	411
NoSuchBucket	The bucket does not exist.	404
NoSuchKey	The object does not exist	404
NoSuchUpload	The multipart upload ID does not exist.	404
NotImplemented	The method cannot be processed.	501
PreconditionFailed	Pre-processing error.	412
RequestTimeTooSkewed	The local time set for OSSClient is deviated from the time set for the OSS server by over 15 minutes.	403
RequestTimeout	Request timeout	400
SignatureDoesNotMatch	Signature mismatch	403

Error code	Description	HTTP status code
TooManyBuckets	The number of buckets exceed the limits,.	400

4.17 FAQ

This topic lists the FAQs about OSS C SDK.

- *How to configure the output of application logs in OSS C SDK.*
- *How to configure parameters for cURL-based communication in OSS C SDK.*
- *How to upload data using callback of cURL in OSS C SDK.*
- *How to upload data using callback of cURL in OSS C SDK.*
- *How to download data using callback of cURL in OSS C SDK.*
- *How to upload and download objects with names containing Chinese characters in OSS C SDK (for Windows).*

5 . NET

5.1 Preface

The OSS C# SDK supports .NET Framework 2.0 and later. This document is written based on OSS C# SDK 2.8.0.

Compatibility

- For SDK series of 2.x.x
  - API: compatible
  - Namespace: compatible
- For SDK series of 1.0.x
  - API: compatible
  - Namespace: incompatible. Aliyun.OpenServices.OpenStorageService is changed to Aliyun.OSS

SDK source code and API documents

For the SDK source code, see *GitHub*. For more information, see *API documents*.

Code samples

OSS SDK for C# provides a variety of code samples for your reference or use. You can obtain the code samples from *GitHub*. The following table describes the content of code samples.

Sample file	Content
PutObjectSample.cs	*Upload objects*
AppendObjectSample.cs	*Append upload*
DoesObjectExistSample.cs	*Determine whether an object exists*
DeleteObjectsSample.cs	*Delete objects*
CopyObjectSample.cs	*Copy objects*
ModifyObjectMetaSample.cs	*Manage Object Meta*
MultipartUploadSample.cs	*Multipart upload*
ResumableSample.cs	*Resumable upload*
GetObjectSample.cs	*Download objects*
GetObjectByRangeSample.cs	*Range download*

Sample file	Content
GetObjectAclSample.cs	*Manage ACL for an object*
SetObjectAclSample.cs	*Manage ACL for an object*
ListObjectsSample.cs	*List objects*
UrlSignatureSample.cs	*Authorized access*
UploadCallbackSample.cs	*Upload callback*
ProgressSample.cs	*Upload progress bars* and *Download progress bars*
CNameSample.cs	Use custome domain name to access OSS (CNAME)
PostPolicySample.cs	*Form upload*
CreateBucketSample.cs	*Create a bucket*
DeleteBucketSample.cs	*Delete a bucket*
Doesbucketexistsample. CS	*Determine whether a bucket exists*
ListBucketsSample.cs	*List buckets*
SetBucketAclSample.cs	*Configure an ACL for a bucket*
SetBucketLifecycleSample.cs	*Lifecycle management*
SetBucketLoggingSample.cs	*Access log files*
SetBucketRefererSample.cs	*Anti-leech*
SetBucketWetbsiteSample.cs	*Static website hosting*
SetBucketCorsSample.cs	*CORS*
ImageProcessSample.cs	*Image processing*

5.2 Installation

This topic describes how to install OSS C# SDK.

Preparation

- Windows
  - Applicable to `.NET 2.0` and later.
  - Applicable to `Visual Studio 2010` and later.
- Linux Mac
  - Applicable to `Mono 3.12` and later.

Download SDK

- *Direct download*
- *Download from GitHub*
- *Download previous versions*

Install SDK

- Install OSS C# SDK in Windows
  - Install SDK through NuGet
    1. If your Visual Studio does not have NuGet installed, install *NuGet* first.
    2. Create a project or open an existing project in Visual Studio, select**Tools** > **NuGet Package Manager** > **Management Solution NuGet Package**.
    3. Search for aliyun.oss.sdk and find Aliyun.OSS.SDK in the search results. Select the latest version, and click **Install**.
  - Install SDK through DLL reference
    1. Download and unzip the C# SDK package.
    2. In the Visual Studio, access **Solution Resource Manager**, select your project, right click Project Name. Select **Reference** > **Add Reference**, and then select **Browse** in the prompt dialog box.
    3. Find thedirectory that the SDK package is unzipped to, select `Aliyun.OSS.dll` in the bin directory, and click **OK**.
  - Install SDK through project introduction

    If you download SDK package or the source code from GitHub and you want to install SDK package using the source code, you can
    1. right click **Solution** in Visual Studio and select **Add** in the dialog box menu to add an existing project.
    2. In the prompt dialog box, select `aliyun-oss-sdk.csproj`, and click **Open**.
    3. Right click Your Projects and select **Reference** > **Add Reference**. In the prompt dialog box, click the **Project** tab, select the aliyun-oss-sdk project, and then click **OK**.
- Install OSS C# SDK in Windows
  - Install SDK through NuGet
    1. In Xamarin, create a project or open an existing project, and select**Add NuGet Packages** from Tool.

2. Search and find aliyun.oss.sdk, select the latest version, and click **Add Package** to add the package to project application.

- Install SDK through DLL reference

1. Download and unzip the C# SDK package.
2. In Xamarin, select your project in **Solution**, right click **Reference**, and then select **Edit References** in the prompt menu.
3. In the Edit References dialog box, select *.Net Assembly*, click**Browse**. Find the directory that SDK is unzipped to, select *Aliyun.OSS.dll* in the bin directory, and then click **Open**.

5.3 Quick start

This topic describes how to use OSS .NET SDK to perform routine operations such as bucket creation, object uploads, and object downloads.

Create a bucket

A bucket is a global namespace in OSS. It is similar to a data container that stores files.

Run the following code to create a bucket:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Create a bucket.
    var bucket = client.CreateBucket(bucketName);
    Console.WriteLine("Create bucket succeeded, {0} ", bucket.Name);
}
catch (Exception ex)
{
    Console.WriteLine("Create bucket failed, {0}", ex.Message);
}
```

For more information about bucket naming rules, see naming conventions in *Basic concepts*.

Upload objects

Run the following code to upload a file to OSS:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
```

```
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Upload a file
    var result = client.PutObject(bucketName, objectName, localFilen
ame);
    Console.WriteLine("Put object succeeded, ETag: {0} ", result.ETag
);
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

For more information, see *Upload an object*.

Download objects

Run the following code to download specified object to a local file:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    client.PutObject(bucketName, objectName, localFilename);
    // Download the object.
    var result = client.GetObject(bucketName, objectName);
    using (var requestStream = result.Content)
    {
        using (var fs = File.Open(downloadFilename, FileMode.
OpenOrCreate))
        {
            int length = 4 * 1024;
            var buf = new byte[length];
            do
            {
                length = requestStream.Read(buf, 0, length);
                fs.Write(buf, 0, length);
            } while (length ! = 0);
        }
    }
    Console.WriteLine("Get object succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
```

```
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

List objects

Run the following code to list objects in a specified bucket:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var objects = new List<string>();
    ObjectListing result = null;
    String nextmarker = string. empty;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        {
            Marker = nextMarker,
        };
        // List objects.
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            objects.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ",
bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Delete objects

Run the following code to delete a specified object:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
```

```
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Delete an object.
    client.DeleteObject(bucketName, objectName);
    Console.WriteLine("Delete object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Delete object failed, {0}", ex.Message);
}
```

5.4 Initialization

OSSClient serves as the OSS C# client to manage OSS resources such as buckets and objects.

Create an OSSClient instance

To create an OSSClient instance, you need to specify an endpoint. For more information about endpoints, see *Regions and endpoints* and *Bind a custom domain name*.

- Use an OSS domain to create an OSSClient instance

  Run the following code to create an OSSClient instance with a domain assigned by OSS:

```
using Aliyun.OSS;

const string accessKeyId = "<yourAccessKeyId>";
const string accessKeySecret = "<yourAccessKeySecret>";
const string endpoint = "http://oss-cn-hangzhou.aliyuncs.com";

// Create a new OSSClient instance with the OSS access address
specified by the user and the AccessKeyId/AccessKeySecret granted by
 Alibaba Cloud.
var ossClient = new OssClient(endpoint, accessKeyId, accessKeySecret
);
```

- Use a custom domain (CNAME) to create an OSSClient instance

  Run the following code to create an OSSClient instance with CNAME:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

const string accessKeyId = "<yourAccessKeyId>";
const string accessKeySecret = "<yourAccessKeySecret>";
const string endpoint = "<yourDomain>";

// Create a ClientConfiguration instance. Modify parameters as
required.
var conf = new ClientConfiguration();

// Enable CNAME. CNAME indicates a custom domain bound to a bucket.
conf.IsCname = true;
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret,
conf);
```

Note:

The ossClient.listBuckets method cannot be used when a CNAME is used.

Configure an OSSClient instance

ClientConfiguration is a configuration class of OSSClient. ClientConfiguration is used to configure parameters such as user agents, host proxies, connection timeout, and the maximum number of connections. You can configure the following parameters.

Parameter	Description	Configuration method
ConnectionLimit	Specifies the maximum number of HTTP connections that can be enabled.	512
MaxErrorRetry	Specifies the maximum number of retry attempts in the case of a request error.	3
ConnectionTimeout	Specifies the timeout time in milliseconds when connections are established. The default value is -1, indicating that the connections are not time-out.	-1
IsCname	Specifies whether CNAME can be used as an endpoint.	false
ProgressUpdateInterval	Specifies the update interval of the progress bar, which is calculated by bytes.	8096

A code example is given as follows:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var conf = new ClientConfiguration();
conf.ConnectionLimit = 512;
conf.MaxErrorRetry = 3;
conf.ConnectionTimeout = 300;
```

```
var client = new OssClient(endpoint, accessKeyId, accessKeySecret,
conf);
```

- Data verification

  Run the following code for MD5 data verification:

  ```
  using Aliyun.OSS;
  using Aliyun.OSS.Common;

  var conf = new ClientConfiguration();
  // Enable MD5 verification. Set EnableMD5Check to perform MD5
  verification on the uploaded or downloaded data. MD5 verification is
   disabled by default.
  conf.EnalbeMD5Check = true;

  var client = new OssClient(endpoint, accessKeyId, accessKeySecret,
  conf);
  ```

  Note:

  System performance reduces when MD5 verification is disabled.

- Proxy network

  If you use proxy networks, you can configure the following parameters to access OSS:

Parameter	Description	Default value
ProxyHost	A proxy server, such as `8.8.8.8` or `abc.def.com`.	Null
ProxyPort	A proxy port, such as `3128` or `8080`.	Null
ProxyUserName	The proxy service account, which is optional	Null
ProxyPassword	The proxy service password, which is optional	Null

  An example of proxy network access without using the account and password is as follows:

  ```
  using Aliyun.OSS;
  using Aliyun.OSS.Common;

  var conf = new ClientConfiguration();
  conf.ProxyHost = "8.8.8.8";
  conf.ProxyPort = 3128;
  ```

```
var client = new OssClient(endpoint, accessKeyId, accessKeySecret,
conf);
```

An example of proxy network access using the account and password is as follows:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var conf = new ClientConfiguration();
conf.ProxyHost = "8.8.8.8";
conf.ProxyPort = 3128;
conf.ProxyUserName = "user";
conf.ProxyPassword = "6666";

var client = new OssClient(endpoint, accessKeyId, accessKeySecret,
conf);
```

5.5 Manage a bucket

A bucket serves as a container that stores objects. Objects belong to a bucket.

Create a bucket

For the complete code of creating a bucket, see *GitHub*.

You can run the following code to create a bucket:

```
using Aliyun.OSS;

// Initialize an OSSClient.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);

// Create a new bucket.
public void CreateBucket(string bucketName)
{
    try
    {
        // Create a bucket. The bucketName is globally unique.
        client.CreateBucket(bucketName);
        Console.WriteLine("Create bucket succeeded");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Create bucket failed. {0}", ex.Message);
    }
}
```

List buckets

For the complete code of listing buckets, see *GitHub*.

You can run the following code to list all buckets under an account:

```
using Aliyun.OSS;

// Initialize an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
```

```
// List the information about all buckets under an account.
public void ListBuckets()
{
    try
    {
        var buckets = client.ListBuckets();

        Console.WriteLine("List bucket succeeded");
        foreach (var bucket in buckets)
        {
         Console.WriteLine("Bucket name : {0} , Location : {1} , Owner : {2
}", bucket.Name, bucket.Location, bucket.Owner);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("List bucket failed. {0}", ex.Message);
    }
}
```

Determine whether a bucket exists

For the complete code of determining whether a bucket exists, see *GitHub*.

You can run the following code to determine whether a bucket exists:

```
using Aliyun.OSS;

//Initialize an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);

// Determine whether a bucket exists.
public void DoesBucketExist(string bucketName)
{
    try
    {
        var exist = client.DoesBucketExist(bucketName);

        Console.WriteLine("Check object Exist succeeded");
        Console.WriteLine("exist ? {0}", exist);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Check object Exist failed. {0}", ex.Message
);
    }
}
```

Configure an ACL for a bucket

For the complete code of configuring ACL, see *GitHub*.

The ACL of a bucket includes the following permissions.

Permission	Description	Value
Private	The bucket owner and the authorized users can read and write objects in the bucket. Other users cannot perform any operation on the objects.	oss.ACLPrivate
Public read	The bucket owner and the authorized users can read and write objects in the bucket . Other users can only read the objects in the bucket. Authorize this permission with caution.	oss.ACLPublicRead
Public read-write	All users can read and write objects in the bucket. Authorize this permission with caution.	oss.ACLPublicReadWrite

You can run the following code to configure an ACL for a bucket:

```
using Aliyun.OSS;

// Initialize an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);

// Configure an access ACL for the bucket
public void SetBucketAcl(string buckteName)
{
    try
    {
        // Set the ACL of the bucket to public read.
        client.SetBucketAcl(bucketName, CannedAccessControlList.
PublicRead);
        Console.WriteLine("Set bucket ACL succeeded");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Set bucket ACL failed. {0}", ex.Message);
    }
}
```

Obtain the ACL for a bucket

For the complete code of obtaining the ACL for a bucket, see *GitHub*.

You can run the following code to obtain the ACL for a bucket:

```
using Aliyun.OSS;

// Initialize an OSSClient instance.
```

```
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);

// Obtain the ACL for the bucket.
public void GetBucketAcl(string bucketName)
{
    try
    {
        string bucketName = "your-bucket";
        var acl = client.GetBucketAcl(bucketName);

        Console.WriteLine("Get bucket ACL success");
        foreach (var grant in acl.Grants)
        {
             Console.WriteLine ("The bucket ACL has been obtained
successfully. Current ACL: {0}", grant.Permission.ToString());
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Get bucket ACL failed. {0}", ex.Message);
    }
}
```

Delete a bucket

For the complete code of deleting a bucket, see *GitHub*.

Before a bucket is deleted, ensure that all objects in the bucket and fragments that are generated from multipart upload are deleted.

Note:
To delete the fragments that are generated from multipart upload, use *Bucket.ListMultipartUploads* to list all fragments, and then use *Bucket.AbortMultipartUpload* to delete the fragments.

You can run the following code to delete a bucket:

```
using Aliyun.OSS;

// Initialize an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);

// Delete the bucket.
public void DeleteBucket(string bucketName)
{
    try
    {
        client.DeleteBucket(bucketName);

        Console.WriteLine("Delete bucket succeeded");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Delete bucket failed. {0}", ex.Message);
    }
```

```
}
```

5.6 Upload objects

5.6.1 Overview

In OSS, an object is the basic unit for data operations. OSS .NET SDK provides the following file upload methods:

- *Simple upload*: supports the upload of a file up to 5 GB in size.
- *Append upload*: supports the upload of a file up to 5 GB in size.
- *Resumable upload*: supports concurrent upload and you can define the size of each part. It applies to the upload of large files. Resumable upload supports the upload of a file up to 48.8 TB in size.
- *Multipart upload*: supports the upload of a file up to 48.8 TB in size. It applies to the upload of large files.

Note:

For more information about the usage scenarios of each upload method, see “Upload files” in OSS Developer Guide.

During the file upload, you can set *Object Meta*, and view the *upload progress*. After you complete the file upload, you can perform *upload callback*.

5.6.2 Simple upload

Upload a string

For the complete code of uploading a string, see *GitHub*.

You can run the following code to upload a string:

```
using System.Text;
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var objectContent = "More than just cloud." ;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    // Upload a file.
```

```
    client.PutObject(bucketName, objectName, requestContent);
    Console.WriteLine("Put object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

Upload a specified local file

For the complete code of uploading a specified local file, see *Git Hub*.

You can run the following code to upload a specified local file:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Upload a local file.
    client.PutObject(bucketName, objectName, localFilename);
    Console.WriteLine("Put object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

Upload a file with MD5 verification

To ensure the consistency between the data sent by the SDK and the data received by the OSS client, you can add a Content-MD5 value in ObjectMeta. Then OSS checks the received data against the MD5 value.

For the complete code of uploading a file with MD5 verification, see *GitHub*.

```
using Aliyun.OSS;
using Aliyun.OSS.Util;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Calculate the MD5 value.
    string md5;
    using (var fs = File.Open(localFilename, FileMode.Open))
```

```
    {
        md5 = OssUtils.ComputeContentMd5(fs, fs.Length);
    }
    var objectMeta = new ObjectMetadata
    {
        ContentMd5 = md5
    };
    // Upload a file.
    client.PutObject(bucketName, objectName, localFilename, objectMeta
);
    Console.WriteLine("Put object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

Note:

MD5 verification may cause a reduction in performance.

Asynchronous upload

For the complete code of asynchronous upload, see *GitHub*.

You can run the following code to perform asynchronous upload:

```
using System;
using System.IO;
using System.Threading;

using Aliyun.OSS;
using Aliyun.OSS.Common;

// Upload a file asynchronously.
namespace AsyncPutObject
{
    class Program
    {
        static string endpoint = "<yourEndpoint>";
        static string accessKeyId = "<yourAccessKeyId>";
        static string accessKeySecret = "<yourAccessKeySecret>";
        static string bucketName = "<yourBucketName>";
        static string objectName = "<yourObjectName>";
        static string localFilename = "<yourLocalFilename>";

        static AutoResetEvent _event = new AutoResetEvent(false);

        // Create an OSSClient instance.
        static OssClient client = new OssClient(endpoint, accessKeyId
, accessKeySecret);

        private static void PutObjectCallback(IAsyncResult ar)
        {
            try
            {
                client.EndPutObject(ar);
                Console.WriteLine(ar.AsyncState as string);
                Console.WriteLine("Put object succeeded");
```

```
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                _event.Set();
            }
        }

        public static void AsyncPutObject()
        {
            try
            {
                using (var fs = File.Open(localFilename, FileMode.Open
))
                {
                    var metadata = new ObjectMetadata();
     // Add custom meta information.
                    metadata.UserMetadata.Add("mykey1", "myval1");
                    metadata.UserMetadata.Add("mykey2", "myval2");
                    metadata.CacheControl = "No-Cache";
                    metadata.ContentType = "text/html";
                    string result = "Notice user: put object finish";

                    // Upload a file asynchronously.
                    client.BeginPutObject(bucketName, objectName, fs,
metadata, PutObjectCallback, result.ToCharArray());

                    _event.WaitOne();
                }
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error
info: {1}. \nRequestID:{2}\tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId
);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}", ex.
Message);
            }
        }

        static void Main(string[] args)
        {
            Program.AsyncPutObject();
        }
    }
}
```

Note:

When using asynchronous upload, you must implement your own callback handler.

5.6.3 Append upload

For the complete code of append upload, see *GitHub*.

You cannot perform copyObject for appended objects. When you call AppendObject to upload data, if the target object does not exist, an appendable object is created. If the object already exists, data is appened to the object.

Run the following code for append upload:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// If the object is appended for the first time, the append position
is 0, and the returned value is the position for the next append. The
position for the next append is the length of object before the append
.
long position = 0;
try
{
    var metadata = client.GetObjectMetadata(bucketName, objectName);
    position = metadata.ContentLength;
}
catch (Exception) { }
try
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        var request = new AppendObjectRequest(bucketName, objectName)
        {
            ObjectMetadata = new ObjectMetadata(),
            Content = fs,
            Position = position
        };
        // Append the object.
        var result = client.AppendObject(request);
        // Configure the position where the object is appended
        position = result.NextAppendPosition;
        Console.WriteLine("Append object succeeded, next append
position:{0}", position);
    }
    // Obtain the start position for the next append.
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        var request = new AppendObjectRequest(bucketName, objectName)
        {
            ObjectMetadata = new ObjectMetadata(),
            Content = fs,
            Position = position
        };
        var result = client.AppendObject(request);
        position = result.NextAppendPosition;
```

```
        Console.WriteLine("Append object succeeded, next append
position:{0}", position);
    }
}
catch (Exception ex)
{
    Console.WriteLine("Append object failed, {0}", ex.Message);
}
```

5.6.4 Resumable upload

You can use resumable upload to split a file you want to upload into several parts and upload them simultaneously. After you have uploaded all parts, you can combine these parts into a complete object. Thus, you can complete uploading the entire object.

Current upload progress is recorded into a checkpoint file during the upload. If a part fails to be uploaded during the process, the object is uploaded from the part recorded in the checkpoint file when the upload restarts, that is, the resumable upload. After the upload is complete, the checkpoint is deleted.

For the complete code of resumable upload, see *GitHub*.

Run the following code for resumable upload:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
string checkpointDir = "<yourCheckpointDir>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    //  Configure parameters with UploadFileRequest.
    UploadObjectRequest request = new UploadObjectRequest(bucketName,
objectName, localFilename)
    {
        // Specify the size of a part you want to upload.
        PartSize = 8 * 1024 * 1024,
        // Specify the number of threads for simultaneous upload.
        ParallelThreadCount = 3,
        // The progress information of a resumable upload is stored
in the checkpointDir directory.  If a part fails to be uploaded, the
progress information is used to continue the upload. If the checkpoint
Dir directory is null, the resumable upload does not take effect, and
the file that failed to be uploaded is uploaded all over again.
        CheckpointDir = checkpointDir,
    };
    // Start resumable upload.
    client.ResumableUploadObject(request);
    Console.WriteLine("Resumable upload object:{0} succeeded",
objectName);
```

```
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.6.5 Multipart upload

For the complete code of multipart upload, see *GitHub*.

To enable multipart upload, perform the following steps:

1. Initiate a multipart upload event.

   You can call OssClient.initiateMultipartUpload to return the globally unique uploadId created in OSS.

2. Upload parts.

   You can call OssClient.uploadPart to upload part data.

   Note:

   - For parts with a same uploadId, parts are sequenced by their part numbers. If you have uploaded a part and use the same part number to upload another part, the later part will replace the former part.
   - OSS places the MD5 value of part data in ETag and returns the MD5 value to the user.
   - SDK automatically configures Content-MD5. OSS calculates the MD5 value of uploaded data and compares it with the MD5 value calculated by SDK. If the two values vary, the error code of InvalidDigest is returned.

3. Complete multipart upload.

   After you have uploaded all parts, call partossClient.completeMultipartUpload to combine these parts into a complete object.

The following code is used as a complete example that describes the process of multipart upload:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
```

```
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Initiate the multipart upload.
var uploadId = "";
try
{
    // Specifiy the name and the bucket of the uploaded object. You
can configure ObjectMeta when using InitiateMultipartUploadRequest.
However, you do not need to specify the ContentLength.
    var request = new InitiateMultipartUploadRequest(bucketName,
objectName);
    var result = client.InitiateMultipartUpload(request);
    uploadId = result.UploadId;
    // Print the UploadId
    Console.WriteLine("Init multi part upload succeeded");
    Console.WriteLine("Upload Id:{0}", result.UploadId);
}
catch (Exception ex)
{
    Console.WriteLine("Init multi part upload failed, {0}", ex.Message
);
}
// Calculate the total number of parts.
var partSize = 100 * 1024;
var fi = new FileInfo(localFilename);
var fileSize = fi.Length;
var partCount = fileSize / partSize;
if (fileSize % partSize ! = 0)
{
    partCount++;
}
// Start the multipart upload. partETags is a list of partETags. OSS
 verifies the validity of all parts one by one after it receives the
partETags. After part verification is successful, OSS combines these
parts into a complete object.
var partETags = new List<PartETag>();
try
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        for (var i = 0; i < partCount; i++)
        {
            var skipBytes = (long)partSize * i;
            // Locate to the start position of the upload.
            fs.Seek(skipBytes, 0);
            // Calculate the size of the uploaded part. The size of
the last part is the size of the data remained after being splited by
the part size.
            var size = (partSize < fileSize - skipBytes) ? partSize :
 (fileSize - skipBytes);
            var request = new UploadPartRequest(bucketName, objectName
, uploadId)
            {
                InputStream = fs,
                PartSize = size,
                PartNumber = i + 1
            };
            // Call the UploadPart interface to upload the object. The
 returned results contain the ETag value of the uploaded part.
            var result = client.UploadPart(request);
            partETags.Add(result.PartETag);
```

```
            Console.WriteLine("finish {0}/{1}", partETags.Count,
partCount);
        }
        Console.WriteLine("Put multi part upload succeeded");
    }
}
catch (Exception ex)
{
    Console.WriteLine("Put multi part upload failed, {0}", ex.Message
);
}
// List uploaded parts.
try
{
    var listPartsRequest = new ListPartsRequest(bucketName, objectName
, uploadId);
    var listPartsResult = client.ListParts(listPartsRequest);
    Console.WriteLine("List parts succeeded");
    // Upload each part simultaneously until all parts are uploaded.
    var parts = listPartsResult.Parts;
    foreach (var part in parts)
    {
        Console.WriteLine("partNumber: {0}, ETag: {1}, Size: {2}",
part.PartNumber, part.ETag, part.Size);
    }
}
catch (Exception ex)
{
    Console.WriteLine("List parts failed, {0}", ex.Message);
}
// Complete the multipart upload.
try
{
    var completeMultipartUploadRequest = new CompleteMultipartUpl
oadRequest(bucketName, objectName, uploadId);
    foreach (var partETag in partETags)
    {
        completeMultipartUploadRequest.PartETags.Add(partETag);
    }
    var result = client.CompleteMultipartUpload(completeMultipartUpl
oadRequest);
    Console.WriteLine("complete multi part succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("complete multi part failed, {0}", ex.Message);
}
```

Cancel a multipart upload event

For the complete code of canceling a multiplart upload event, see *GitHub*.

Run the following code to cancel a multipart upload event:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var uploadId = "";
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Initiate the multipart upload.
try
{
    var request = new InitiateMultipartUploadRequest(bucketName,
objectName);
    var result = client.InitiateMultipartUpload(request);
    uploadId = result.UploadId;
    // Print the UploadId.
    Console.WriteLine("Init multi part upload succeeded");
    Console.WriteLine("Upload Id:{0}", result.UploadId);
}
catch (Exception ex)
{
    Console.WriteLine("Init multi part upload failed, {0}", ex.Message
);
}
// Cancel the multipart upload.
try
{
    var request = new AbortMultipartUploadRequest(bucketName,
objectName, uploadId);
    client.AbortMultipartUpload(request);
    Console.WriteLine("Abort multi part succeeded， {0}", uploadId);
}
catch (Exception ex)
{
    Console.WriteLine("Abort multi part failed, {0}", ex.Message);
}
```

List uploaded parts

For the complete code of listing uploaded parts, see *GitHub*.

Run the following code to list uploaded parts:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var uploadId = "<yourUploadId>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    PartListing listPartsResult = null;
    var nextMarker = 0;
    do
    {
        var listPartsRequest = new ListPartsRequest(bucketName,
objectName, uploadId)
        {
            PartNumberMarker = nextMarker,
        };
        // List uploaded parts.
        listPartsResult = client.ListParts(listPartsRequest);
        Console.WriteLine("List parts succeeded");
```

```
        // Upload each part simultaneously until all parts are
uploaded.
        var parts = listPartsResult.Parts;
        foreach (var part in parts)
        {
            Console.WriteLine("partNumber: {0}, ETag: {1}, Size: {2
}", part.PartNumber, part.ETag, part.Size);
        }
        nextMarker = listPartsResult.NextPartNumberMarker;
    } while (listPartsResult.IsTruncated);
}
catch (Exception ex)
{
    Console.WriteLine("List parts failed, {0}", ex.Message);
}
```

List multipart upload events

Call ossClient.listMultipartUploads to list all ongoing part upload events (events that have been initiated but not completed or have been canceled). You can configure the following parameters:

Parameter	Description	Configuration method
prefix	Specifies the prefix that must be included in the returned object name. Note that if you use a prefix for query, the returned object name will contain the prefix.	ListMultipartUploadsRequest. setPrefix(String prefix)
delimiter	Specifies a delimiter of a forward slash (/) used to group object names. The object between the specified prefix and the first occurrence of a delimiter of a forward slash (/) is commonPrefixes.	ListMultipartUploadsRequest. setDelimiter(String delimiter)
maxUploads	Specifies the maximum number of part upload events . The maximum value (also default value) you can set is 1, 000.	ListMultipartUploadsRequest .setMaxUploads(Integer maxUploads)
keyMarker	Lists all part upload events with the object whose names start with a letter that comes after the keyMarker value in the alphabetical order. You can use this parameter with the uploadIdMarker parameter to	ListMultipartUploadsRequest .setKeyMarker(String keyMarker)

Parameter	Description	Configuration method
	specify the initial position for the specified returned result.	
uploadIdMarker	You can use this parameter with the keyMarker parameter to specify the initial position for the specified returned result. If you do not configure keyMarker, the uploadIdMarker parameter is invalid. If you configure keyMarker, the query result contains: • All objects whose names start with a letter that comes after the keyMarker value. • All objects whose names start with a letter that is the same as the keyMarker value in the alphabetical order and the value of uploadId greater than that of uploadIdMarker.	ListMultipartUploadsRequest .setUploadIdMarker(String uploadIdMarker)

For the complete code of listing multiplart upload events, see *GitHub*.

Run the following code to list all multipart upload events:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    MultipartUploadListing multipartUploadListing = null;
    var nextMarker = string.Empty;
    do
    {
        // List multipart upload events.
        var request = new ListMultipartUploadsRequest(bucketName)
        {
            KeyMarker = nextMarker,
        };
        multipartUploadListing = client.ListMultipartUploads(request);
        Console.WriteLine("List multi part succeeded");
        // List the information about multipart upload events.
```

```
        foreach (var mu in multipartUploadListing.MultipartUploads)
        {
            Console.WriteLine("Key: {0},  UploadId: {1}", mu.Key, mu.
UploadId);
        }
        // If the returned result shows that the value of isTruncate
d is false, the values of nextKeyMarker and nextUploadIdMarker are
returned and used as the initial position for the next object reading.
        nextMarker = multipartUploadListing.NextKeyMarker;
    } while (multipartUploadListing.IsTruncated);
}
catch (Exception ex)
{
    Console.WriteLine("List multi part uploads failed, {0}", ex.
Message);
}
```

Specify the prefix and maximum number of returned results

Run the following code to perform multipart upload with a specified prefix and maximum number of returned results:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Define prefix.
var prefix = "<yourObjectPrefix>";
// Set the maximum number of returned result to 100.
var maxUploads = 100;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    MultipartUploadListing multipartUploadListing = null;
    var nextMarker = string.Empty;
    do
    {
        // List multipart upload events. By default, 1,000 parts are
listed.
        var request = new ListMultipartUploadsRequest(bucketName)
        {
            KeyMarker = nextMarker,
            // Specify the prefix.
            Prefix = prefix,
            // Specify the maximum number of returned result.
            MaxUploads = maxUploads,
        };
        multipartUploadListing = client.ListMultipartUploads(request);
        Console.WriteLine("List multi part succeeded");
        // List the information about each multipart upload event.
        foreach (var mu in multipartUploadListing.MultipartUploads)
        {
            Console.WriteLine("Key: {0},  UploadId: {1}", mu.Key, mu.
UploadId);
        }
        nextMarker = multipartUploadListing.NextKeyMarker;
    } while (multipartUploadListing.IsTruncated);
}
```

```
catch (Exception ex)
{
    Console.WriteLine("List multi part uploads failed, {0}", ex.
Message);
}
```

5.6.6 Upload progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of progress bars, see *GitHub*.

The following code is used as an example to describe how to view progress information with PutObject:

```
using System;
using System.IO;
using System.Text;
using Aliyun.OSS;
using Aliyun.OSS.Common;
namespace PutObjectProgress
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.PutObjectProgress();
            Console.ReadKey();
        }
        public static void PutObjectProgress()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localFilename = "<yourLocalFilename>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            // Upload with a progress bar displayed.
            try
            {
                using (var fs = File.Open(localFilename, FileMode.Open
))
                {
                    var putObjectRequest = new PutObjectRequest(
bucketName, objectName, fs);
                    putObjectRequest.StreamTransferProgress +=
streamProgressCallback;
                    client.PutObject(putObjectRequest);
                }
                Console.WriteLine("Put object:{0} succeeded",
objectName);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error
info: {1}. \nRequestID: {2}\tHostID: {3}",
```

```
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId
);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}", ex.
Message);
            }
        }
        // Obtain the upload progress.
        private static void streamProgressCallback(object sender,
StreamTransferProgressArgs args)
        {
            System.Console.WriteLine("ProgressCallback - Progress: {0
}%, TotalBytes:{1}, TransferredBytes:{2} ",
                args.TransferredBytes * 100 / args.TotalBytes, args.
TotalBytes, args.TransferredBytes);
        }
    }
}
```

5.6.7 Upload callback

For the complete code of upload callback, see *GitHub*.

Run the following code for upload callback:

```
using System;
using System.IO;
using System.Text;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace Callback
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.PutObjectCallback();
            Console.ReadKey();
        }
        public static void PutObjectCallback()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localFilename = "<yourLocalFilename>";
            // Specify the IP address of the server you want to send
the callback request to.
            const string callbackUrl = "http://oss-demo.aliyuncs.com:
23450";
            # Configure the value of the body field carried in the
callback request.
            const string callbackBody = "bucket=${bucket}&object=${
object}&etag=${etag}&size=${size}&mimeType=${mimeType}&" +
                                        "my_var1=${x:var1}&my_var2=${x
:var2}";
```

```
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            try
            {
                string responseContent = "";
                var metadata = BuildCallbackMetadata(callbackUrl,
callbackBody);
                using (var fs = File.Open(localFilename, FileMode.Open
))
                {
                    var putObjectRequest = new PutObjectRequest(
bucketName, objectName, fs, metadata);
                    var result = client.PutObject(putObjectRequest);
                    responseContent = GetCallbackResponse(result);
                }
                Console.WriteLine("Put object:{0} succeeded, callback
response content:{1}", objectName, responseContent);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error
info: {1}. \nRequestID: {2}\tHostID: {3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId
);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}", ex.
Message);
            }
        }
        // Configure upload callback.
        private static ObjectMetadata BuildCallbackMetadata(string
callbackUrl, string callbackBody)
        {
            string callbackHeaderBuilder = new CallbackHeaderBuilder(
callbackUrl, callbackBody). Build();
            string CallbackVariableHeaderBuilder = new CallbackVa
riableHeaderBuilder().
                AddCallbackVariable("x:var1", "x:value1"). AddCallbac
kVariable("x:var2", "x:value2"). Build();
            var metadata = new ObjectMetadata();
            metadata.AddHeader(HttpHeaders.Callback, callbackHe
aderBuilder);
            metadata.AddHeader(HttpHeaders.CallbackVar, CallbackVa
riableHeaderBuilder);
            return metadata;
        }
        // Read the message returned from upload callback.
        private static string GetCallbackResponse(PutObjectResult
putObjectResult)
        {
            string callbackResponse = null;
            using (var stream = putObjectResult.ResponseStream)
            {
                var buffer = new byte[4 * 1024];
                var bytesRead = stream.Read(buffer, 0, buffer.Length);
                callbackResponse = Encoding.Default.GetString(buffer,
0, bytesRead);
            }
            return callbackResponse;
```

```
        }
    }
}
```

Note:

For more information about upload callback, see *Upload callback* in the Developer Guide.

5.7 Download objects

5.7.1 Overview

OSS .NET SDK provides the following download methods:

- *Streaming download*
- *Range download*
- *Resumable download*

You can view the download progress shown in *download progress bars*.

5.7.2 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 each time until you have downloaded the entire object.

For the complete code of streaming download, see *GitHub*.

Run the following code to download a specified OSS object to a stream:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Download an object to a stream. OssObject includes object
information, such as the bucket that contains the object, object name
, metadata, and an input stream.
    var obj = client.GetObject(bucketName, objectName);
    using (var requestStream = obj.Content)
    {
        byte[] buf = new byte[1024];
        var fs = File.Open(downloadFilename, FileMode.OpenOrCreate);
        var len = 0;
        // Use the input stream to read the object content into a file
 or the memory.
```

```
        while ((len = requestStream.Read(buf, 0, 1024)) ! = 0)
        {
            fs.Write(buf, 0, len);
        }
        fs.Close();
    }
    Console.WriteLine("Get object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Get object failed. {0}", ex.Message);
}
```

5.7.3 Range download

If you only need part of the data in an object, you can use range downloads to download specified content.

For the complete code of range download, see *GitHub*.

Run the following code for range download:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var getObjectRequest = new GetObjectRequest(bucketName, objectName
);
    // Set the range, which is from the 20th byte to the 100th byte.
    getObjectRequest.SetRange(20, 100);
    // Start range download. The setRange of getObjectRequest can be
used to implement multipart download and resumable download.
    var obj = client.GetObject(getObjectRequest);
    // Download the data and write it into a file.
    using (var requestStream = obj.Content)
    {
        byte[] buf = new byte[1024];
        var fs = File.Open(downloadFilename, FileMode.OpenOrCreate);
        var len = 0;
        while ((len = requestStream.Read(buf, 0, 1024)) ! = 0)
        {
            fs.Write(buf, 0, len);
        }
        fs.Close();
    }
    Console.WriteLine("Get object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Get object failed. {0}", ex.Message);
```

```
}
```

You can configure the following parameters of GetObjectRequest.

Parameter	Description
Range	Specifies the transmission range of an object.
ModifiedSinceConstraint	If the specified time is earlier than the actual modification time of the file, the file is transmitted normally. Otherwise, a 304 Not Modified exception is thrown.
UnmodifiedSinceConstraint	If the time passed to the parameter is equal to or later than the actual modification time of the file, the file is transmitted normally Otherwise, a 412 precondition failed exception is thrown.
MatchingETagConstraints	Passes in a group of ETags. If the ETags match the ETag of the file, the file is transmitted normally. Otherwise, a 412 precondition failed exception is thrown.
NonmatchingEtagConstraints	Passes in a group of ETags. If the ETags do not match the ETag of the file, the file is transmitted normally. Otherwise, a 304 Not Modified exception is thrown.
ResponseHeaderOverrides	Customizes some headers in requests returned by OSS.

5.7.4 Resumable download

Large-sized objects may fail to be downloaded because the network is unstable or the Java program exits abnormally and the entire object needs to be downloaded again. However, the object may still fail to be downloaded after multiple attempts. Therefore, OSS provides resumable download.

Run the following code for resumable download:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
var downloadFilename = "<yourDownloadFilename>";
var checkpointDir = "<yourCheckpointDir>";
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Configure parameters with DownloadObjectRequest.
    DownloadObjectRequest request = new DownloadObjectRequest(
bucketName, objectName, downloadFilename)
    {
        // Specify the size of a part you want to download.
        PartSize = 8 * 1024 * 1024,
        // Specifies the number of concurrent download threads.
        ParallelThreadCount = 3,
        // Specify the checkpointDir to store the download progress
information. If the download fails, it can be continued based on the
 progress information. If the checkpointDir directory is null, the
resumable upload is not enabled, which means that objects that fails
to be downloaded are downloaded all over again.
        CheckpointDir = checkpointDir,
    };
    // Start resumable download.
    client.ResumableDownloadObject(request);
    Console.WriteLine("Resumable download object:{0} succeeded",
objectName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

For more information about resumable download, see *Resumable download* in OSS Developer Guide.

5.7.5 Download progress bars

You can use progress bars to indicate the upload or download progress.

For the complete code of download progress bar, see *GitHub*.

The following code is used as an example to describe how to view progress information with GetObject:

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
namespace GetObjectProgress
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.GetObjectProgress();
```

```
            Console.ReadKey();
        }
        public static void GetObjectProgress()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            try
            {
                var getObjectRequest = new GetObjectRequest(bucketName
, objectName);
                getObjectRequest.StreamTransferProgress += streamProg
ressCallback;
                // Download an object.
                var ossObject = client.GetObject(getObjectRequest);
                using (var stream = ossObject.Content)
                {
                    var buffer = new byte[1024 * 1024];
                    var bytesRead = 0;
                    while ((bytesRead = stream.Read(buffer, 0, buffer.
Length)) > 0)
                    {
                        // Process the read data (detailed code is not
 listed here).
                    }
                }
                Console.WriteLine("Get object:{0} succeeded",
objectName);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0}; Error
info: {1}. \nRequestID:{2}\tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId
);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}", ex.
Message);
            }
        }
        private static void streamProgressCallback(object sender,
StreamTransferProgressArgs args)
        {
            System.Console.WriteLine("ProgressCallback - Progress: {0
}%, TotalBytes:{1}, TransferredBytes:{2} ",
                args.TransferredBytes * 100 / args.TotalBytes, args.
TotalBytes, args.TransferredBytes);
        }
    }
}
```

5.8 Manage objects

5.8.1 Overview

This topic is the overview of object management in OSS .NET SDK.

You can perform the following operations on objects in a bucket with APIs:

- *Determine whether an object exists*
- *Manage ACL for an object*
- *Manage Object Meta*
- *List objects*
- *Delete an object*
- *Copy an object*
- *Restore an archive object*
- *Manage a symbolic link*

5.8.2 Determine whether a specified object exists

Run the following code to determine whether a specified object exists:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";

// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    //  Determine whether a specified object exists.
    var exist = client.DoesObjectExist(bucketName, objectName);
    Console.WriteLine("Object exist ? " + exist);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.8.3 Manage ACL for an object

The following table describes the permissions included in the Access Control List (ACL) for an object.

Permission	Description	Value
default	The ACL of an object is the same with that of its bucket.	CannedAccessControlList.Default
Private	Only the object owner and authorized users can read and write the object.	CannedAccessControlList.Private
Public read	Only the object owner and authorized users can read and write the object. Other users can only read the object. Authorize this permission with caution.	CannedAccessControlList.PublicRead
Public read-write	All users can read and write the object. Authorize this permission with caution.	CannedAccessControlList.PublicReadWrite

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.

For the complete code of configuring an ACL for an object, see *GitHub*. For the complete code of obtaining the ACL for an object, see *GitHub*.

You can run the following code to configure an ACL for an object and obtain the ACL for an object:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;

var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";

// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// Configure an ACL for an object.
try
{
    // Use SetObjectAcl to configure an ACL for an object.
    client.SetObjectAcl(bucketName, objectName, CannedAccessControlL
ist.PublicRead);
    Console.WriteLine("Set Object:{0} ACL succeeded ", objectName);
}
catch (Exception ex)
{
    Console.WriteLine("Set Object ACL failed with error info: {0}", ex
.Message);
}
```

```
// Obtain the ACL for an object.
try
{
    // Use GetObjectAcl to obtain the ACL for an object.
    var result = client.GetObjectAcl(bucketName, objectName);
    Console.WriteLine("Get Object ACL succeeded, Id: {0}  ACL: {1}",
        result.Owner.Id, result.ACL.ToString());
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID: {2}\tHostID: {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

For details about object ACL, see *Access control*.

5.8.4 Manage Object Meta

Object Meta includes HTTP headers and user-defined Object Meta. For more information, see *Object Meta* in OSS Developer Guide.

For the complete code of configuring Object Meta, see *GitHub*. For the complete code of modifying Object Meta, see *GitHub*.

Run the following code to configure, modify, and obtain Object Meta:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var localFilename = "<yourLocalFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    using (var fs = File.Open(localFilename, FileMode.Open))
    {
        // Create Object Meta. You can configure HTTP headers with
Object Meta.
        var metadata = new ObjectMetadata()
        {
            // Specify the object type.
            ContentType = "text/html",
            // Configure the expiration time (GMT is used).
            ExpirationTime = DateTime.Parse("2025-10-12T00:00:00.000Z
"),
        };
        // Configure the length of the object you want to upload.
If the object length is greater than the configured length, only
the configured length of the object is uploaded. If the configured
```

```
length is greater than the actual object length, the entire object is
uploaded.
        metadata.ContentLength = fs.Length;
        // Configure the cache action of a webpage when an object is
downloaded.
        metadata.CacheControl = "No-Cache";
        // Set the value of metadata mykey1 to myval1.
        metadata.UserMetadata.Add("mykey1", "myval1");
        // Set the value of metadata mykey2 to myval2.
        metadata.UserMetadata.Add("mykey2", "myval2");
        var saveAsFilename = "Filetest123.txt";
        var contentDisposition = string.Format("attachment;filename*=
utf-8''{0}", HttpUtils.EncodeUri(saveAsFilename, "utf-8"));
        // Provide a default file name when saving the requested
content as a file.
        metadata.ContentDisposition = contentDisposition;
        // Upload a file and configure the Object Meta.
        client.PutObject(bucketName, objectName, fs, metadata);
        Console.WriteLine("Put object succeeded");
        //Obtain Object Meta.
        var oldMeta = client.GetObjectMetadata(bucketName, objectName
);
        // Configure new Object Meta.
        var newMeta = new ObjectMetadata()
        {
            ContentType = "application/octet-stream",
            ExpirationTime = DateTime.Parse("2035-11-11T00:00:00.000Z
"),
            // Specify the encoding format of a download object.
            ContentEncoding = null,
            CacheControl = ""
        };
        // Add custom Object Meta.
        newMeta.UserMetadata.Add("author", "oss");
        newMeta.UserMetadata.Add("flag", "my-flag");
        newMeta.UserMetadata.Add("mykey2", "myval2-modified-value");
        // Use the ModifyObjectMeta method to modify Object Meta.
        client.ModifyObjectMeta(bucketName, objectName, newMeta);
    }
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
```

Note:

- For detailed information about HTTP header, see *RFC2616*.
- The size of Object Meta with custom HTTP headers is no larger than 8 KB.

5.8.5 List objects

For the complete code of listing objects, see *GitHub*.

Objects are listed alphabetically. You can use ListObjects to list objects in a bucket. The following table describes the parameters of ListObjects

Parameter	Description
Delimiter	Specifies a delimiter of a forward slash (/) used to group object names. CommonPrefixes is a set of objects which are ended with the delimiter and have the same prefix.
Marker	Specifies the initial object in the list.
MaxKeys	Specifies the maximum number of objects that can be listed. The default value of MaxKeys is 100. The maximum value of MaxKeys is 1,000.
Prefix	Specifies the prefix you configure to list required objects.

Simple list

For the complete code of simple list, see *GitHub*.

Run the following code to list objects in a specified bucket.

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var listObjectsRequest = new ListObjectsRequest(bucketName);
    // Simply list the objects in a specified bucket. 100 records are
returned by default.
    var result = client.ListObjects(listObjectsRequest);
    Console.WriteLine("List objects succeeded");
    foreach (var summary in result.ObjectSummaries)
    {
        Console.WriteLine("File name:{0}", summary.Key);
    }
}
catch (Exception ex)
{
    Console.WriteLine("List objects failed. {0}", ex.Message);
}
```

List a specified number of objects

Use the following code to list a specified number of objects:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var listObjectsRequest = new ListObjectsRequest(bucketName)
    {
        // A maximum of 200 records can be returned.
        MaxKeys = 200,
    };
    var result = client.ListObjects(listObjectsRequest);
    Console.WriteLine("List objects succeeded");
    foreach (var summary in result.ObjectSummaries)
    {
        Console.WriteLine(summary.Key);
    }
}
catch (Exception ex)
{
    Console.WriteLine("List objects failed, {0}", ex.Message);
}
```

List objects by a specified prefix

Use the following code to list objects with a specified prefix:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var prefix = "<yourObjectPrefix>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    ObjectListing result = null;
    string nextMarker = string.Empty;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        {
            Marker = nextMarker,
            MaxKeys = 100,
            Prefix = prefix,
        };
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            keys.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ",
bucketName);
}
catch (OssException ex)
{
```

```
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

List objects by marker

The marker parameter indicates the name of an object from which the listing begins. Run the following code to specify an object (specified with marker) after which the listing begins:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var marker = "<yourObjectMarker>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    ObjectListing result = null;
    string nextMarker = marker;
    do
    {
        var listObjectsRequest = new ListObjectsRequest(bucketName)
        // To increase the number of returned objects, you can modify
the MaxKeys parameter or use the Marker parameter for multiple reads.
        {
            Marker = nextMarker,
            MaxKeys = 100,
        };
        result = client.ListObjects(listObjectsRequest);
        foreach (var summary in result.ObjectSummaries)
        {
            Console.WriteLine(summary.Key);
            keys.Add(summary.Key);
        }
        nextMarker = result.NextMarker;
    // If the value of IsTruncated is true, the next read starts from
NextMarker.
    } while (result.IsTruncated);
    Console.WriteLine("List objects of bucket:{0} succeeded ",
bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

List objects in an asynchronous mode

For the complete code of listing objects in an asynchronous mode, see *GitHub*.

Run the following code to list objects in an asynchronous mode:

```
using System;
using System.IO;
using System.Threading;
using Aliyun.OSS;
namespace AsyncListObjects
{
    class Program
    {
        static string endpoint = "<yourEndpoint>";
        static string accessKeyId = "<yourAccessKeyId>";
        static string accessKeySecret = "<yourAccessKeySecret>";
        static string bucketName = "<yourBucketName>";
        static AutoResetEvent _event = new AutoResetEvent(false);
        // Create an OSSClient instance.
        static OssClient client = new OssClient(endpoint, accessKeyId
, accessKeySecret);
        static void Main(string[] args)
        {
            Program.AsyncListObjects();
            Console.ReadKey();
        }
        public static void AsyncListObjects()
        {
            try
            {
                var listObjectsRequest = new ListObjectsRequest(
bucketName);
                client.BeginListObjects(listObjectsRequest, ListObject
Callback, null);
                _event.WaitOne();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Async list objects failed. {0}", ex
.Message);
            }
        }
        // The ListObjectCallback method is implemented after an
asynchronous call. Similar interfaces must be used when an asynchrono
us interface is called to list objects.
        private static void ListObjectCallback(IAsyncResult ar)
        {
            try
            {
                var result = client.EndListObjects(ar);
                foreach (var summary in result.ObjectSummaries)
                {
                    Console.WriteLine ("Object name: 0", summary.Key);
                }
                _event.Set();
                Console.WriteLine("Async list objects succeeded");
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine("Async list objects failed. {0}", ex
.Message);
            }
        }
    }
}
```

List all objects in a bucket

Run the following code to list all objects in a bucket:

```
using Aliyun.OSS;
// Initialize OssClient.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
// List all objects in the bucket.
public void ListObject(string bucketName)
{
    try
    {
        ObjectListing result = null;
        string nextMarker = string.Empty;
        do
        {
            // The number of objects listed in each page is specified
 by the maxKeys parameter. If the object number is larger than the
specified value, other files are listed in another page.
            var listObjectsRequest = new ListObjectsRequest(bucketName
)
            {
                Marker = nextMarker,
                MaxKeys = 100
            };
            result = client.ListObjects(listObjectsRequest);
            Console.WriteLine("File:");
            foreach (var summary in result.ObjectSummaries)
            {
                Console.WriteLine("Name:{0}", summary.Key);
            }
            nextMarker = result.NextMarker;
        } while (result.IsTruncated);
    }
    catch (Exception ex)
    {
        Console.WriteLine("List object failed. {0}", ex.Message);
    }
}
```

5.8.6 Delete objects

This topic describes how to delete objects.

Warning:

Delete objects with caution because deleted objects cannot be recovered.

For the complete code of deleting objects, see *GitHub*.

Delete an object

Run the following code to delete a single object:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    /// Delete an object.
    client.DeleteObject(bucketName, objectName);
    Console.WriteLine("Delete object succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Delete object failed. {0}", ex.Message);
}
```

Delete multiple objects

You can delete a maximum of 1,000 objects simultaneously. Objects can be deleted in two modes : detail (verbose) and simple (quiet) modes.

- verbose: returns the list of objects that you have deleted successfully. The default mode is verbose.
- quiet: returns the list of objects that you failed to delete.

Run the following code to delete multiple objects simultaneously:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var keys = new List<string>();
    var listResult = client.ListObjects(bucketName);
    foreach (var summary in listResult.ObjectSummaries)
    {
        keys.Add(summary.Key);
    }
    // If the quietMode is true, the quiet mode is used. If the
quietMode is false, the verbose mode is used. The default mode is
verbose.
    var quietMode = false;
    // The third parameter of the DeleteObjectsRequest specifies the
return mode.
    var request = new DeleteObjectsRequest(bucketName, keys, quietMode
);
```

```
    // Delete multiple objects.
    var result = client.DeleteObjects(request);
    if ((! quietMode) && (result.Keys ! = null))
    {
        foreach (var obj in result.Keys)
        {
            Console.WriteLine("Delete successfully : {0} ", obj.Key);
        }
    }
    Console.WriteLine("Delete objects succeeded");
}
catch (Exception ex)
{
    Console.WriteLine("Delete objects failed. {0}", ex.Message);
}
```

5.8.7 Copy objects

Copy a small-sized object

For the complete code of copying a small-sized object, see *GitHub*.

Note the following limits before copying a small-sized object:

- You must have the read and write permission on the source object.
- You cannot copy an object from a region to another region. For example, you cannot copy an object from a bucket in Hangzhou region to a bucket in Qingdao region.
- The object size is limited to a maximum of 1 GB.

Run the following code to copy a small-sized object:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var metadata = new ObjectMetadata();
    metadata.AddHeader("mk1", "mv1");
    metadata.AddHeader("mk2", "mv2");
    var req = new CopyObjectRequest(sourceBucket, sourceObject,
targetBucket, targetObject)
    {
        // If the value of NewObjectMetadata is null, the COPY mode is
 used (the metadata of the source object is copied). If the value of
NewObjectMetadata is not null, the REPLACE mode is used (the metadata
of the source object is replaced).
        NewObjectMetadata = metadata
    };
    // Copy an object.
```

```
    client.CopyObject(req);
    Console.WriteLine("Copy object succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID: {2} \tHostID: {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Copy a large-sized object

For the complete code of copying large-sized objects, see *GitHub*.

- UploadPartCopy

  To copy an object larger than 1 GB, you need to use UploadPartCopy.

  1. Use InitiateMultipartUploadRequest to initiate an UploadPartCopy event.
  2. Use the UploadPartCopy method to perfome a multipart copy.
  3. Use the CompleteMultipartUpload method to complete the object copy.

  Run the following code for UploadPartCopy task:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
var uploadId = "";
var partSize = 50 * 1024 * 1024;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Initiate a copy task. You can use the InitiateMultipartUpl
oadRequest method to specify the metadata information about the
target object.
    var request = new InitiateMultipartUploadRequest(targetBucket,
targetObject);
    var result = client.InitiateMultipartUpload(request);
    // Print the uploadId.
    uploadId = result.UploadId;
    Console.WriteLine("Init multipart upload succeeded， Upload Id:
 {0}", result.UploadId);
    // Calculate the number of parts.
    var metadata = client.GetObjectMetadata(sourceBucket, sourceObje
ct);
    var fileSize = metadata.ContentLength;
    var partCount = (int)fileSize / partSize;
    if (fileSize % partSize ! = 0)
```

```
    {
        partCount++;
    }
    // Start the UploadPartCopy task.
    var partETags = new List<PartETag>();
    for (var i = 0; i < partCount; i++)
    {
        var skipBytes = (long)partSize * i;
        var size = (partSize < fileSize - skipBytes) ? partSize : (
fileSize - skipBytes);
        // Create UploadPartCopyRequest. You can specify conditions
with UploadPartCopyRequest.
        var uploadPartCopyRequest = new UploadPartCopyRequest(
targetBucket, targetObject, sourceBucket, sourceObject, uploadId)
            {
                PartSize = size,
                PartNumber = i + 1,
                // Beginindex is used to locate the location that
the last UploadPartCopy starts.
                BeginIndex = skipBytes
            };
        // Call the uploadPartCopy method to copy each part.
        var uploadPartCopyResult = client.UploadPartCopy(uploadPart
CopyRequest);
        Console.WriteLine("UploadPartCopy : {0}", i);
        partETags.Add(uploadPartCopyResult.PartETag);
    }
    // Complete the UploadPartCopy task.
    var completeMultipartUploadRequest =
    new CompleteMultipartUploadRequest(targetBucket, targetObject,
uploadId);
    // The partETags refers to the list of partETags saved during
 the multipart upload. After OSS receives the part list submitted
by the user, it verifies the validity of each data part one by one
. After the verification is passed, OSS conbine these parts into a
complete object.
    foreach (var partETag in partETags)
    {
        completeMultipartUploadRequest.PartETags.Add(partETag);
    }
    var completeMultipartUploadResult = client.CompleteMultipartUpl
oad(completeMultipartUploadRequest);
    Console.WriteLine("CompleteMultipartUpload succeeded");
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}.
 \nRequestID: {2} \tHostID: {3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

- Resumable copy

  If the copy is interrupted, you can use resumable copy to continue the copy. For the complete code of resumable copy, see *GitHub*.

Run the following code for resumable copy:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var sourceBucket = "<yourSourceBucketName>";
var sourceObject = "<yourSourceObjectName>";
var targetBucket = "<yourDestBucketName>";
var targetObject = "<yourDestObjectName>";
var checkpointDir = @"<yourCheckpointDir>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var request = new CopyObjectRequest(sourceBucket, sourceObject,
targetBucket, targetObject);
    // The checkpointDir directory stores the intermediate state
information of a resumable upload. The information will be used when
 a failed upload task is resumed. // If the checkpointDir directory
 is null, resumable copy will not take effect, and the object that
previously failed to be copied will be copied all over again.
    client.ResumableCopyObject(request, checkpointDir);
    Console.WriteLine("Resumable copy new object:{0} succeeded",
request.DestinationKey);
}
catch (Exception ex)
{
    Console.WriteLine("Resumable copy new object failed, {0}", ex.
Message);
}
```

5.8.8 Restore an archive object

For the complete code of restoring an archive object, see *GitHub*.

You must restore an archive object before you read it. Do not call restoreObject for non-archive objects.

The state conversion process of an archive object is as follows:

- An archive object is in the frozen state.
- After you submit it for restoration, the server restores the object. The object is in the restoring state.
- You can read the object after it is restored. The restored state of the object lasts one day by default. You can prolong this period to a maximum of seven days. Once this period expires, the object returns to the frozen state.

Run the following code to restore an object:

```
using Aliyun.OSS;
using Aliyun.OSS.Model;
var endpoint = "<yourEndpoint>";
```

```
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var objectContent = "More than just cloud.";
int maxWaitTimeInSeconds = 600;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Create a bucket of the archive storage class..
    var bucket = client.CreateBucket(bucketName, StorageClass.Archive
);
    Console.WriteLine("Create Archive bucket succeeded, {0} ", bucket.
Name);
}
catch (Exception ex)
{
    Console.WriteLine("Create Archive bucket failed, {0}", ex.Message
);
}
// Upload a file.
try
{
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    client.PutObject(bucketName, objectName, requestContent);
    Console.WriteLine("Put object succeeded, {0}", objectName);
}
catch (Exception ex)
{
    Console.WriteLine("Put object failed, {0}", ex.Message);
}
var metadata = client.GetObjectMetadata(bucketName, objectName);
string storageClass = metadata.HttpMetadata["x-oss-storage-class"] as
string;
if (storageClass ! = "Archive")
{
    Console.WriteLine("StorageClass is {0}", storageClass);
    return;
}
// Restore the object.
RestoreObjectResult result = client.RestoreObject(bucketName,
objectName);
Console.WriteLine("RestoreObject result HttpStatusCode : {0}", result.
HttpStatusCode);
if (result.HttpStatusCode ! = HttpStatusCode.Accepted)
{
    throw new OssException(result.RequestId + ", " + result.HttpStatus
Code + " ,");
}
while (maxWaitTimeInSeconds > 0)
{
    var meta = client.GetObjectMetadata(bucketName, objectName);
    string restoreStatus = meta.HttpMetadata["x-oss-restore"] as
string;
    if (restoreStatus ! = null && restoreStatus.StartsWith("ongoing-
request=\"false\"", StringComparison.InvariantCultureIgnoreCase))
    {
        break;
    }
    Thread.Sleep(1000);
```

```
    maxWaitTimeInSeconds--;
}
if (maxWaitTimeInSeconds == 0)
{
    Console.WriteLine("RestoreObject is timeout. ");
    throw new TimeoutException();
}
else
{
    Console.WriteLine("RestoreObject is successful. ");
}
```

For more information about the archive storage classes, see *Introduction to storage classes*. For detailed information about related status code, see *RestoreObject*.

5.8.9 Manage a symbolic link

A symbolic link is a special object that maps to an object similar to a shortcut used in Windows . You can configure user-defined Object Meta for symbolic links. To obtain a symbolic link, you must have the read permission on it.

Run the following code to create and obtain a symbolic link:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var targetObjectName = "<yourTargetObjectName>";
var symlinkObjectName = "<yourSymlinkObjectName>";
var objectContent = "More than just cloud." ;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Upload the target file.
    byte[] binaryData = Encoding.ASCII.GetBytes(objectContent);
    MemoryStream requestContent = new MemoryStream(binaryData);
    client.PutObject(bucketName, targetObjectName, requestContent);
    // Create a symbolic link.
    client.CreateSymlink(bucketName, symlinkObjectName, targetObje
ctName);
    // Obtain a symbolic link.
    var ossSymlink = client.GetSymlink(bucketName, symlinkObjectName);
    Console.WriteLine("Target object is {0}", ossSymlink.Target);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

For more information about creating symbolic links, see *PutSymlink*.

For more information about obtaining symbolic links, see *GetSymlink*.

5.9 Authorized access

Use STS for temporary access authorization

OSS supports Alibaba Cloud Security Token Service (STS) for temporary access authorization. STS is a web service that provides a temporary access token to a cloud computing user. Through the STS, you can assign a third-party application or a RAM user (you can manage the user ID) an access credential with a custom validity period and permissions. For more information about STS, see *STS introduction*.

STS advantages:

- Your long-term key (AccessKey) is not exposed to a third-party application. You only need to generate an access token and send the access token to the third-party application. You can customize access permissions and the validity of this token.
- You do not need to keep track of permission revocation issues. The access token automatically becomes invalid when it expires.

For more information about the process of access to OSS with STS, see *RAM and STS scenario practices* in OSS Developer Guide.

Run the following code to create a signature request with STS:

```
using Aliyun.OSS;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var securityToken = "<yourSecurityToken>";
// After a user obtains a temporary STS credential, the OSSClient is
generated with the security token and temporary access key (AccessKeyI
D and AccessKeySecret).
// Create an OSSClient instance.
var ossStsClient = new OssClient(endpoint, accessKeyId, accessKeyS
ecret, securityToken);
// Perform operations on OSS.
```

Sign a URL to authorize temporary access

You can provide a signed URL to a visitor for temporary access. When you sign a URL, you can specify the expiration time for a URL to restrict the period of access from visitors.

For the complete code of signing a URL to authorize temporary access, see *GitHub*.

- Use a signed URL to upload a file

  Run the following code to upload a file with a signed URL:

  ```
  using Aliyun.OSS;
  using Aliyun.OSS.Common;
  ```

```
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var objectContent = "More than just cloud." ;
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Generate a signed URL for upload.
    var generatePresignedUriRequest = new GeneratePresignedUri
Request(bucketName, objectName, SignHttpMethod.Put)
    {
        Expiration = DateTime.Now.AddHours(1),
    };
    var signedUrl = client.GeneratePresignedUri(generatePresignedUri
Request);
    // Upload a file with the signed URL.
    var buffer = Encoding.UTF8. GetBytes(objectContent);
    using (var ms = new MemoryStream(buffer))
    {
        client.PutObject(signedUrl, ms);
    }
    Console.WriteLine("Put object by signatrue succeeded. {0} ",
signedUrl.ToString());
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}.
 \nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

- Use a signed URL to download an object

  Run the following code to download an object with a signed URL:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
var downloadFilename = "<yourDownloadFilename>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var metadata = client.GetObjectMetadata(bucketName, objectName);
    var etag = metadata.ETag;
    // Generate a signed URL for download.
    var req = new GeneratePresignedUriRequest(bucketName, objectName
, SignHttpMethod.Get);
    var uri = client.GeneratePresignedUri(req);
    // Download an object with the signed URL.
```

```
    OssObject ossObject = client.GetObject(uri);
    using (var file = File.Open(downloadFilename, FileMode.
OpenOrCreate))
    {
        using (Stream stream = ossObject.Content)
        {
            int length = 4 * 1024;
            var buf = new byte[length];
            do
            {
                length = stream.Read(buf, 0, length);
                file.Write(buf, 0, length);
            } while (length ! = 0);
        }
    }
    Console.WriteLine("Get object by signatrue succeeded. {0} ", uri
.ToString());
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}.
 \nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.10 Manage lifecycle rules

You can use OSS to configure lifecycle rules to reduce storage costs. The lifecycle management rules can be used for automatic deletion of expired objects and fragments, or storage class conversion to Infrequent Access for objects that will expire soon. Each rule includes:

- Rule ID: It identifies a rule. Ensure that each rule ID in a bucket is unique.
- Policy: You can configure a policy by using the following configuration methods. Only one method can be configured for a bucket.
  - Configure by Prefix: You can create multiple rules with this method. Ensure that each prefix is unique.
  - Configure for Entire Bucket: You can configure only one rule with this method.
- Expired: You can configure the following configuration methods:
  - Set by Number of Days: It specifies the number of days from the last modification time of objects.
  - Set by Date: It specifies the date prior to which objects are created. If objects are created prior to the date, these objects are deleted.
- Status: This lifecyle rule takes effect or not.

The parts uploaded by uploadPart method also support lifecycle rules. The last modification time of an object is the time the multipart upload event is initiated.

For more information about lifecycle management, see *Manage object lifecycle*.

For the complete code of lifecycle management, see *GitHub*.

Configure lifecycle rules

Run the following code to configure lifecycle rules:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var setBucketLifecycleRequest = new SetBucketLifecycleRequest(
bucketName);
    // Create the first lifecycle rule that is valid within 3 days.
    LifecycleRule lcr1 = new LifecycleRule()
    {
        ID = "delete obsoleted files",
        Prefix = "obsoleted/",
        Status = RuleStatus.Enabled,
        ExpriationDays = 3
    };
    // Create the second lifecycle rule.
    LifecycleRule lcr2 = new LifecycleRule()
    {
        ID = "delete temporary files",
        Prefix = "temporary/",
        Status = RuleStatus.Enabled,
        ExpriationDays = 20
    };
    setBucketLifecycleRequest.AddLifecycleRule(lcr1);
    setBucketLifecycleRequest.AddLifecycleRule(lcr2);
    // Configure the lifecycle.
    client.SetBucketLifecycle(setBucketLifecycleRequest);
    Console.WriteLine("Set bucket:{0} Lifecycle succeeded ",
bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

View lifecycle rules

For the complete code of viewing lifecycle rules, see *GitHub*.

Run the following code to view lifecycle rules:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // View the lifecycle rules.
    var rules = client.GetBucketLifecycle(bucketName);
    Console.WriteLine("Get bucket:{0} Lifecycle succeeded ",
bucketName);
    foreach (var rule in rules)
    {
        Console.WriteLine("ID: {0}", rule.ID);
        Console.WriteLine("Prefix: {0}", rule.Prefix);
        Console.WriteLine("Status: {0}", rule.Status);
        if (rule.ExpriationDays.HasValue)
            Console.WriteLine("ExpirationDays: {0}", rule.Expriation
Days);
    }
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Clear lifecycle rules

Run the following code to clear lifecycle rules:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Clear the lifecycle rules
    client.DeleteBucketLifecycle(bucketName);
```

```
    Console.WriteLine("Delete bucket:{0} Lifecycle succeeded ",
bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.11 Set logging

You can enable access logs to record bucket access to log files, which are stored in a specified bucket.

The log file format is as follows:

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

For more information about access log files, see *Set access logging*. For the complete code of access logging, see *GitHub*.

Enable access logging

Run the following code to enable bucket access logging:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var targetBucketName = "<yourTargetBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // The targetBucketName parameter is the name of the bucket that
logs are stored. The logging- option indicates the prefix of log files
. The bucketName and targetBucketName can be the same bucket.
    var request = new SetBucketLoggingRequest(bucketName, targetBuck
etName, "logging-");
    // Enable access logging.
    client.SetBucketLogging(request);
    Console.WriteLine("Set bucket:{0} Logging succeeded ", bucketName
);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
```

```
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

View access logging configurations

For the complete code of viewing access logging configurations, see *GitHub*.

Run the following code to view the access logging configurations for a bucket:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // View bucket access logging configurations.
    var result = client.GetBucketLogging(bucketName);
    Console.WriteLine("Get bucket:{0} Logging succeeded, prefix:{1}",
bucketName, result.TargetPrefix);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Disable access logging

For the complete code of disabling access logging, see *GitHub*.

Run the following code to disable access logging for a bucket:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Disable access logging. Existing log files are not affected.
    client.DeleteBucketLogging(bucketName);
    Console.WriteLine("Delete bucket:{0} Logging succeeded ",
bucketName);
}
catch (OssException ex)
```

```
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.12 CORS

Cross-origin resource sharing (CORS) allows web applications to access resources that belong to another region. OSS provides CORS APIs for convenient cross-origin access control.

For more information, see *Cross-origin resource sharing* and *PutBucketcors* in OSS Developer Guide.

For the complete code of CORS, see *GitHub*.

Configure CORS rules

For complete code of configuring CORS rules, see *GitHub*.

Run the following code to configure CORS rules for a specified bucket:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Creates an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var request = new SetBucketCorsRequest(bucketName);
    var rule1 = new CORSRule();
    // Specify the allowed source of the cross-origin access request.
    rule1. AddAllowedOrigin("http://www.a.com");
    // Specify the cross-region request methods (GET, PUT, DELETE,
POST, and HEAD) that are allowed.
    rule1. AddAllowedMethod("POST");
    // AllowedHeaders and ExposeHeaders do not support wildcards.
    rule1. AddAllowedHeader("*");
    // Specify the response header that allows user access from
applications.
    rule1. AddExposeHeader("x-oss-test");
    // A maximum of 10 rules is allowed.
    request.AddCORSRule(rule1);
    var rule2 = new CORSRule();
    // AllowedOrigins and AllowedMethods allow only one wildcard
asterisk (*). Wildcard asterisks (*) indicate that all sources of the
cross-origin requests and operations are allowed.
    rule2. AddAllowedOrigin("http://www.b.com");
    rule2. AddAllowedMethod("GET");
```

```
    // Determine whether the header specified in the Access-Control-
Headers in the pre-flight (OPTIONS) requests is allowed.
    rule2. AddExposeHeader("x-oss-test2");
    // Specify the cache time (seconds) for the response of browser
pre-flight (OPTIONS) requests to a specific resource.
    rule2. MaxAgeSeconds = 100;
    request.AddCORSRule(rule2);
    // Configure CORS rules.
    client.SetBucketCors(request);
    Console.WriteLine("Set bucket:{0} Cors succeeded ", bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Obtain CORS rules

For the complete code of obtaining CORS rules, see *GitHub*.

Run the following code to obtain CORS rules:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Obtain CORS rules.
    var result = client.GetBucketCors(bucketName);
    Console.WriteLine("Get bucket:{0} Cors succeeded ", bucketName);
    foreach (var rule in result)
    {
        foreach (var origin in rule.AllowedOrigins)
        {
            Console.WriteLine("Allowed origin:{0}", origin);
        }
    }
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

Delete CORS rules

For the complete code of deleting CORS rules, see *GitHub*.

Run the following code to delete all CORS rules for a specified bucket:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Delete CORS rules.
    client.DeleteBucketCors(bucketName);
    Console.WriteLine("Delete bucket:{0} Cors succeeded ", bucketName
);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0} ; Error info: {1}.
 \nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

5.13 Anti-leech

This topic describes how to use anti-leech.

To prevent your data on OSS from being leeched, OSS supports anti-leeching through the referer field settings in the HTTP header, including the following parameters:

- Referer whitelist: Used to allow access only for specified domains to OSS data.
- Empty referer: Determines whether the referer can be empty. If it is not allowed, only requests with the referer filed in their HTTP or HTTPS headers can access OSS data.

For more information about anti-leaching, see *Anti-leeching settings*.

For the complete code of configuring anti-leech, see *GitHub*.

Configure the referer whitelist

Run the following code to configure the referer whitelist:

```
using Aliyun.OSS;
```

```
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var refererList = new List<string>();
    // Add the referer field. The referer field allows question marks
 (?) and asterisks (*) for wildcard use.
    refererList.Add(" http://www.aliyun.com");
    refererList.Add(" http://www. *.com");
    refererList.Add(" http://www.?.aliyuncs.com");
    var srq = new SetBucketRefererRequest(bucketName, refererList);
    // Configure the referer list for a bucket.
    client.SetBucketReferer(srq);
    Console.WriteLine("Set bucket:{0} Referer succeeded ", bucketName
);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Obtain a referer whiltelist

For the complete code of obtaining a referer whitelist, see *GitHub*.

Run the following code to obtain a referer whiltelist:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Obtain the referer list for a bucket.
    var rc = client.GetBucketReferer(bucketName);
    Console.WriteLine("Get bucket:{0} Referer succeeded ", bucketName
);
    Console.WriteLine("allow?" + (rc.AllowEmptyReferer ? "yes" : "no
"));
    if (rc.RefererList.Referers ! = null)
    {
        for (var i = 0; i < rc.RefererList.Referers.Length; i++)
            Console.WriteLine(rc.RefererList.Referers[i]);
    }
    else
    {
```

```
        Console.WriteLine("Empty Referer List");
    }
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
finally
{
    client.SetBucketReferer(new SetBucketRefererRequest(bucketName));
}
```

Clear a referer whitelist

Run the following code to clear a referer whitelist:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // You cannot clear a referer whitelist directly. To clear a
referer whitelist, you need to create the rule that allows an empty
referer field and replace the original rule with the new rule.
    var srq = new SetBucketRefererRequest(bucketName);
    client.SetBucketReferer(srq);
    Console.WriteLine("Set bucket:{0} Referer succeeded ", bucketName
);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

5.14 Image processing

Image Processing (IMG) is a massive, secure, cost-effective and highly reliable image processing service. After source images are uploaded to OSS, you can process images on any Internet device at any anytime, from anywhere through simple RESTful APIs.

For more information about IMG, see *Image Processing*.

Basic features

OSS offers the following IMG features:

- *Retrieve dominant image tones*
- *Format conversion*
- *Resize images*
- *Incircle*
- *Adaptive orientation*
- *Brightness*
- *Add watermarks*: allows you to add images, texts, or image-text watermarks to another image.
- *Image processing*
- *Image processing access rules*: calls multiple IMG functions.

Usage

IMG uses standard HTTP GET. You can configure IMG parameters in QueryString of a URL.

If the ACL of an image object is private read and write, only authorized users are allowed for access.

- Anonymous access

  You can use the following format in level-3 domains to access a processed image:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-
process=image/<yourAction>,<yourParamValue>
```

Parameter	Description
bucket	Specifies the name of a bucket.
endpoint	Specifies endpoint used to access a region.
object	Specifies the name of an image object.

Parameter	Description
image	Specifies the reserved identifier for IMG.
action	Specifies the operations on an image, such as scaling, cropping, and rotating.
param	Specifies the parameter that indicates the operation on an image.

- Basic operations

  For example, scale an image to a width of 100 px. Adjust the height based on the ratio.

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=image/resize,w_100
```

- Customized image styles

  You can use the following format in level-3 domains to access a processed image:

```
http://<yourBucketName>.<yourEndpoint>/<yourObjectName>?x-oss-process=style/<yourStyleName>
```

  - style: specifies a reserved identifier of a customized image style.
  - yourStyleName: specifies the name of a custom image style. It is the name specified in the rule that is created on the OSS console.

  Example:

```
http://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-process=style/oss-pic-style-w-100
```

- Cascade operations

  Cascade operations allow you to perform multiple operations on an image sequentially.

```
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
```

- HTTPS access

IMG supports access through HTTPS. Example:

```
https://image-demo.oss-cn-hangzhou.aliyuncs.com/example.jpg?x-oss-
process=image/resize,w_100
```

- Authorized access

  Authorized access allows you to customize an image style, HTTPS access, and cascade operations.

  Run the following code to generate a signed URL for IMG:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
var objectName = "<yourObjectName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    var process = "image/resize,m_fixed,w_100,h_100";
    var req = new GeneratePresignedUriRequest(bucketName, objectName
, SignHttpMethod.Get)
    {
        Expiration = DateTime.Now.AddHours(1),
        Process = process
    };
    // Generat a signed URI.
    var uri = client.GeneratePresignedUri(req);
    Console.WriteLine("Generate Presigned Uri:{0} with process:{1}
succeeded ", uri, process);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error code: {0}; Error info: {1}.
 \nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

- Access with SDK

  You can use SDK to access and process an image object.

  SDK allows you to customize image styles, HTTPS access, and cascade operations.

  — Basic operations

  Run the following code to perform basic operations on an image:

```
using System;
```

```
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace ImageProcess
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.ImageProcess();
            Console.ReadKey();
        }
        public static void ImageProcess()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localImageFilename = "<yourLocalImageFilename>";
            var downloadDir = "<yourDownloadDir>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            try
            {
                client.PutObject(bucketName, objectName,
localImageFilename);
                // Scale the image.
                var process = "image/resize,m_fixed,w_100,h_100";
                var ossObject = client.GetObject(new GetObjectR
equest(bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-resize.jpg",
ossObject.Content);
                // Crop the image.
                process = "image/crop,w_100,h_100,x_100,y_100,r_1
";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-crop.jpg",
ossObject.Content);
                // Rotate the image.
                process = "image/rotate,90";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-rotate.jpg",
ossObject.Content);
                // Sharpen the image.
                process = "image/sharpen,100";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-sharpen.jpg",
ossObject.Content);
                // Add watermarks.
                process = "image/watermark,text_SGVsbG8g5Zu-
54mH5pyN5YqhIQ";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-watermark.jpg
", ossObject.Content);
                // Convert the image format.
```

```
                process = "image/format,png";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-format.png",
ossObject.Content);
                // Obtain image information.
                process = "image/info";
                ossObject = client.GetObject(new GetObjectRequest(
bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-info.txt",
ossObject.Content);
                Console.WriteLine("Get Object:{0} with process:{1
} succeeded ", objectName, process);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0};
Error info: {1}. \nRequestID:{2}\tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.
HostId);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}",
ex.Message);
            }
        }
        private static void WriteToFile(string filePath, Stream
stream)
        {
            using (var requestStream = stream)
            {
                using (var fs = File.Open(filePath, FileMode.
OpenOrCreate))
                {
                    IoUtils.WriteTo(stream, fs);
                }
            }
        }
    }
}
```

- Customize an image style

  Use the following code to customize an image style:

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
using Aliyun.OSS.Util;
namespace ImageProcessCustom
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.ImageProcessCustomStyle();
            Console.ReadKey();
        }
        public static void ImageProcessCustomStyle()
```

```
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localImageFilename = "<yourLocalImageFilename>";
            var downloadDir = "<yourDownloadDir>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            try
            {
                client.PutObject(bucketName, objectName,
localImageFilename);
                // Customize the image style: "style/<yourCustom
StyleName>".
                var process = "style/oss-pic-style-w-100";
                var ossObject = client.GetObject(new GetObjectR
equest(bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-style.jpg",
ossObject.Content);
                Console.WriteLine("Get Object:{0} with process:{1
} succeeded ", objectName, process);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0};
Error info: {1}. \nRequestID:{2}\tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.
HostId);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}",
ex.Message);
            }
        }
        private static void WriteToFile(string filePath, Stream
stream)
        {
            using (var requestStream = stream)
            {
                using (var fs = File.Open(filePath, FileMode.
OpenOrCreate))
                {
                    IoUtils.WriteTo(stream, fs);
                }
            }
        }
    }
}
```

- Cascade operations

  Use the following code to perform cascade operations on an image:

```
using System;
using System.IO;
using Aliyun.OSS;
using Aliyun.OSS.Common;
```

```
using Aliyun.OSS.Util;
namespace ImageProcessCascade
{
    class Program
    {
        static void Main(string[] args)
        {
            Program.ImageProcessCascade();
            Console.ReadKey();
        }
        public static void ImageProcessCascade()
        {
            var endpoint = "<yourEndpoint>";
            var accessKeyId = "<yourAccessKeyId>";
            var accessKeySecret = "<yourAccessKeySecret>";
            var bucketName = "<yourBucketName>";
            var objectName = "<yourObjectName>";
            var localImageFilename = "<yourLocalImageFilename>";
            var downloadDir = "<yourDownloadDir>";
            // Create an OSSClient instance.
            var client = new OssClient(endpoint, accessKeyId,
accessKeySecret);
            try
            {
                client.PutObject(bucketName, objectName,
localImageFilename);
                // Perform cascade operations.
                var process = "image/resize,m_fixed,w_100,h_100/
rotate,90";
                var ossObject = client.GetObject(new GetObjectR
equest(bucketName, objectName, process));
                WriteToFile(downloadDir + "/sample-style.jpg",
ossObject.Content);
                Console.WriteLine("Get Object:{0} with process:{1
} succeeded ", objectName, process);
            }
            catch (OssException ex)
            {
                Console.WriteLine("Failed with error code: {0};
Error info: {1}. \nRequestID:{2}\tHostID:{3}",
                    ex.ErrorCode, ex.Message, ex.RequestId, ex.
HostId);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed with error info: {0}",
ex.Message);
            }
        }
        private static void WriteToFile(string filePath, Stream
stream)
        {
            using (var requestStream = stream)
            {
                using (var fs = File.Open(filePath, FileMode.
OpenOrCreate))
                {
                    IoUtils.WriteTo(stream, fs);
                }
            }
        }
    }
```

```
}
```

IMG tools

You can use the IMG viewer *ImageStyleViever* to directly view the IMG result.

5.15 Static website hosting

You can set your bucket configuration to the static website hosting mode. After the configuration takes effect, you can access this static website with the bucket domain and be redirected to a specified index page or error page.

For more information about static website hosting, see *Static website hosting*.

Configure static website hosting

Run the following code to configure static website hosting:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Configure static website hosting.
    var request = new SetBucketWebsiteRequest(bucketName, "index.html
", "error.html");
    client.SetBucketWebsite(request);
    Console.WriteLine("Set bucket:{0} Wetbsite succeeded ", bucketName
);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

View static website hosting configurations

Run the following code to view static website hosting configurations:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
```

```
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // View static website hosting configurations.
    var result = client.GetBucketWebsite(bucketName);
    Console.WriteLine("Get bucket:{0} Wetbsite succeeded, index doc:{1
}, error doc:{2}",
                      bucketName, result.IndexDocument, result.
ErrorDocument);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
}
```

Delete static website hosting configurations

Run the following code to delete static website hosting configurations:

```
using Aliyun.OSS;
using Aliyun.OSS.Common;
var endpoint = "<yourEndpoint>";
var accessKeyId = "<yourAccessKeyId>";
var accessKeySecret = "<yourAccessKeySecret>";
var bucketName = "<yourBucketName>";
// Create an OSSClient instance.
var client = new OssClient(endpoint, accessKeyId, accessKeySecret);
try
{
    // Delete static website hosting configurations.
    client.DeleteBucketWebsite(bucketName);
    Console.WriteLine("Delete bucket:{0} Wetbsite succeeded ",
bucketName);
}
catch (OssException ex)
{
    Console.WriteLine("Failed with error info: {0}; Error info: {1}. \
nRequestID:{2}\tHostID:{3}",
        ex.ErrorCode, ex.Message, ex.RequestId, ex.HostId);
}
catch (Exception ex)
{
    Console.WriteLine("Failed with error info: {0}", ex.Message);
```

```
}
```

5.16 Exception handling

OSS .NET SDK has two types of exceptions: ClientException and OSSException. Both share the properties of RuntimeException.

ClientException

The ClientException exception occurs when the client attempts to send requests and when the data is transmitted to OSS. For example, when a request is sent, ClientException may occur due to unavailable network connections. ClientException may occur during file uploads due to IO exceptions.

OSSException

OSSException indicates a server exception. The exception occurs because a server error message is parsed. OSSException includes the error code and information returned from OSS. It is used to locate and handle problems.

The following table describes common OSSException information.

Parameter	Description
Code	Specifies the error code returned by OSS.
Message	Specifies the detailed error information returned by OSS.
RequestId	Specifies the UUID. It is unique and used to identify a request. If a problem persists, send the RequestId to OSS developers for help.
HostId	Identifies an OSS cluster. Ensure that the value of HostId is consistent with the Host (endpoint to access a bucket) used in the request.

OSS error codes

Error code	Description
AccessDenied	Access denied
BucketAlreadyExists	The bucket already exists.
BucketNotEmpty	The bucket is not empty.
EntityTooLarge	The entity size exceeds the maximum limit.

Error code	Description
EntityTooSmall	The entity size is below the minimum limit.
FileGroupTooLarge	The total file group size exceeds the maximum limit.
FilePartNotExist	No such part exists.
FilePartStale	The part has expired.
InvalidArgument	The parameter format is invalid.
InvalidAccessKeyId	No such AccessKeyId exists.
InvalidBucketName	The bucket name is invalid.
InvalidDigest	The digest is invalid.
InvalidObjectName	The object name is invalid.
InvalidPart	The part is invalid.
InvalidPartOrder	The part order is invalid.
InvalidTargetBucketForLogging	The buckets that store log files are invalid.
InternalError	Internal OSS error
MalformedXML	The XML format is invalid.
MethodNotAllowed	The method is not allowed.
MissingArgument	Parameters are not configured.
MissingContentLength	The content length is not configured.
NoSuchBucket	No such bucket exists.
NoSuchKey	No such object exists.
NoSuchUpload	No such uploadId exists.
NotImplemented	The methods are not implemented.
PreconditionFailed	Precondition error
RequestTimeTooSkewed	The local time set for OSSClient is deviated from the time set for the OSS server by over 15 minutes.
RequestTimeout	Request timeout
SignatureDoesNotMatch	Signature mismatch
TooManyBuckets	The number of buckets exceeds the limit.