1 Object Storage Service ................................................................................................................................................. 1
  1.1 Overview ........................................................................................................................................................................... 1
  1.2 Basic Concepts .................................................................................................................................................................. 1
    1.2.1 Accounts ....................................................................................................................................................................... 2
    1.2.2 Bucket ......................................................................................................................................................................... 2
    1.2.3 Object ............................................................................................................................................................................ 3
    1.2.4 Endpoints and Access Domain Names .......................................................... 3
    1.2.5 REST API ..................................................................................................................................................................... 6
  1.3 Comparison Between OBS and File Systems .................................................................................................................. 8
  1.4 List of Features .................................................................................................................................................................. 9

2 Creating Requests .......................................................................................................................................................... 12
  2.1 Obtaining Permanent Access Keys (AK/SK) ................................................................................................................... 12
  2.2 Obtaining an Endpoint ...................................................................................................................................................... 13
  2.3 Constructing a Request ................................................................................................................................................... 13
    2.3.1 Creating a Request Using the REST API .................................................................................................................... 13

3 Using Buckets ............................................................................................................................................................... 17
  3.1 Use Restrictions .............................................................................................................................................................. 17
  3.2 Creating a Bucket ............................................................................................................................................................. 18
  3.3 Deleting a Bucket ............................................................................................................................................................. 23
  3.4 Managing Bucket Quotas ................................................................................................................................................ 23
  3.5 Managing Bucket Tags ................................................................................................................................................... 24

4 Using Objects ................................................................................................................................................................. 25
  4.1 Basic Concepts of Objects ................................................................................................................................................ 25
  4.2 Object Metadata ............................................................................................................................................................... 25
  4.3 Uploading an Object ......................................................................................................................................................... 29
    4.3.1 Overview of Object Upload Overview ....................................................................................................................... 29
    4.3.2 Uploading an Object Using the PUT Method ............................................................................................................ 30
    4.3.3 Uploading an Object Using the POST Method ....................................................................................................... 30
  4.4 Copying an Object ........................................................................................................................................................... 31
  4.5 Obtaining an Object ......................................................................................................................................................... 32
  4.6 Deleting an Object ........................................................................................................................................................... 33
  4.7 Batch Deleting Objects ................................................................................................................................................... 33
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8 Multipart Upload</td>
<td>34</td>
</tr>
<tr>
<td>4.9 Appending Data to an Object</td>
<td>39</td>
</tr>
<tr>
<td><strong>5 Permission Control</strong></td>
<td>41</td>
</tr>
<tr>
<td>5.1 Overview of Permission Control</td>
<td>41</td>
</tr>
<tr>
<td>5.2 OBS Permission Control Model</td>
<td>41</td>
</tr>
<tr>
<td>5.3 ACL</td>
<td>44</td>
</tr>
<tr>
<td>5.3.1 Introduction to the ACL</td>
<td>44</td>
</tr>
<tr>
<td>5.3.2 Configuring ACLs for Existing Resources</td>
<td>45</td>
</tr>
<tr>
<td>5.3.3 Configuring an ACL Using Header Fields</td>
<td>46</td>
</tr>
<tr>
<td>5.3.4 Conversion Between Two ACL Formats</td>
<td>48</td>
</tr>
<tr>
<td>5.3.5 ACL Application Examples</td>
<td>51</td>
</tr>
<tr>
<td>5.4 Bucket Policy</td>
<td>52</td>
</tr>
<tr>
<td>5.4.1 About Bucket Policy</td>
<td>52</td>
</tr>
<tr>
<td>5.4.2 Policy Format</td>
<td>53</td>
</tr>
<tr>
<td>5.4.3 Bucket Policy Examples</td>
<td>61</td>
</tr>
<tr>
<td>5.4.4 URL Validation Settings</td>
<td>66</td>
</tr>
<tr>
<td>5.5 IAM User Policy (IAM Policy)</td>
<td>67</td>
</tr>
<tr>
<td>5.5.1 Introduction to IAM Policy</td>
<td>67</td>
</tr>
<tr>
<td>5.5.2 Common Policies</td>
<td>74</td>
</tr>
<tr>
<td>5.6 Examples About Permission Control</td>
<td>76</td>
</tr>
<tr>
<td><strong>6 Data Transmission</strong></td>
<td>79</td>
</tr>
<tr>
<td><strong>7 Data Protection Mechanisms</strong></td>
<td>80</td>
</tr>
<tr>
<td>7.1 Versioning</td>
<td>80</td>
</tr>
<tr>
<td><strong>8 Storage Classes</strong></td>
<td>85</td>
</tr>
<tr>
<td>8.1 Introduction to Storage Classes</td>
<td>85</td>
</tr>
<tr>
<td>8.2 Bucket Storage Classes</td>
<td>85</td>
</tr>
<tr>
<td>8.3 Object Storage Classes</td>
<td>86</td>
</tr>
<tr>
<td><strong>9 Object Lifecycle Management</strong></td>
<td>87</td>
</tr>
<tr>
<td>9.1 Lifecycle Management Overview</td>
<td>87</td>
</tr>
<tr>
<td>9.2 General Precautions</td>
<td>88</td>
</tr>
<tr>
<td>9.3 Response Elements for Lifecycle Configuration</td>
<td>90</td>
</tr>
<tr>
<td>9.4 Lifecycle Configuration Examples</td>
<td>97</td>
</tr>
<tr>
<td>9.5 Setting Lifecycle Configuration</td>
<td>99</td>
</tr>
<tr>
<td><strong>10 Static Website Hosting</strong></td>
<td>100</td>
</tr>
<tr>
<td>10.1 Principles of static website hosting</td>
<td>100</td>
</tr>
<tr>
<td>10.2 Configuring static website hosting</td>
<td>101</td>
</tr>
<tr>
<td>10.3 Configuring Flash for Cross-domain Access</td>
<td>110</td>
</tr>
<tr>
<td><strong>11 Cross-Region Replication</strong></td>
<td>113</td>
</tr>
<tr>
<td><strong>12 Event notification</strong></td>
<td>124</td>
</tr>
</tbody>
</table>
Object Storage Service

1.1 Overview

1.1 Overview

Object Storage Service (OBS) provides stable, secure, efficient, and easy-to-use cloud storage services. It also provides the REST, which enables you to store unstructured data of any amount and form.

- Stable
  With reliability built into multiple levels of the architecture, OBS ensures data durability and service continuity.

- Secure
  OBS uses trusted cloud authentication to secure data with support for multiple protection mechanisms, including server-side encryption, URL validation, log auditing, and fine-grained permission control.

- Efficient
  OBS offers intelligent scheduling capabilities with accelerated transfer speeds and vertically optimized big data analytics. It improves experience by maximizing concurrency access to data at enlarged bandwidth while minimizing latency.

- Easy to Use
  OBS supports the REST APIs which are compatible with all mainstream client tools. Furthermore, OBS gives you the freedom to upload, download, and manage your data anytime, anywhere.

1.2 Basic Concepts
1.2.1 Accounts

To access OBS, you need to register an account with the public cloud service and obtain the access keys. For details, see 2.1 Obtaining Permanent Access Keys (AK/SK). A Tenant (an account) is the owner of a bucket or object. The owner of a bucket or object created by an account is also an account.

Tenants are also the entities that bear fees.

A Tenant account can have a maximum of 100 buckets across all regions. For details about the relationship between tenants and buckets, see 5.2 OBS Permission Control Model. Each bucket has its belonging account (tenant), the creator. User permissions can be granted to other accounts, the creator account bears all the incurred costs.

When making an API call, you need to enter the account ID in some XML files. account ID, domain ID, and account ID in this document are the same because the account ID in the token is domain. Therefore, obtain the account ID on the management console first. Obtain the account ID as follows:

1. Log in to the Console.
2. Click the username and select Basic Information from the drop-down list.
3. On the displayed page, click Manage my credentials. On the My Credential page, the account ID is displayed.

Figure 1-1 My credential

1.2.2 Bucket

- Bucket and Object
  A bucket is a top-level data structure in OBS. It is a container for objects stored in OBS. Each object is contained in a bucket. The number of objects that can be contained in a bucket is not limited. Objects in a bucket do not have a hierarchical structure similar to the directory structure in a file system. They are managed in a flattened structure. The folders that you see on the OBS Console are simulations from object names. They are not true and the directory structure does not exist in OBS.

- Bucket and account
  A bucket name is globally unique and a bucket can be owned by only one account. By default, a bucket is owned by the account that created the bucket.
Bucket and Region
A bucket resides in a region. You can specify the region when creating the bucket. Once the bucket is created, its region cannot be changed. For details about how to create a bucket, see 3.2 Creating a Bucket.

Bucket Features
You can modify the properties of a bucket to control the access permission of the bucket, set object lifecycle rules, and manage versioning for the bucket.

Bucket Naming
For details about the bucket naming rules, see 3.1 Use Restrictions.

1.2.3 Object
An object is a fundamental entity in OBS. A bucket can store multiple objects, which may be text files or videos. An object consists of data and metadata. Metadata is used to describe the object data. Metadata stores system information such as the object creation time and object size. It also stores information set by users. In addition, if a user can access the data of an object, the user has the metadata access permission.

A single object can be uploaded with the maximum size of 5 GB. If the size of an object has exceeded 5 GB, multipart upload is required. OBS supports the data size ranging from 0 KB to 48.8 TB.

For details about object operations, see 4 Using Objects.

For details about object naming rules, see 4.2 Object Metadata.

1.2.4 Endpoints and Access Domain Names
1. Concepts about endpoints and access domain names
## Table 1-1 Basic concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td><code>obs.region.myhuaweicloud.com</code>. In this address, <code>region</code> means the area where the service is available. [Example] obs.ap-southeast-1.myhuaweicloud.com obs.ap-southeast-2.myhuaweicloud.com obs.ap-southeast-3.myhuaweicloud.com obs.af-south-1.myhuaweicloud.com</td>
<td>OBS provides an endpoint for each region. An endpoint can be considered as the domain name of OBS in a region, and is used to process access requests from the region. For details about endpoints of each region, see Regions and Endpoints.</td>
</tr>
<tr>
<td>Bucket access domain name</td>
<td><strong>BucketName.Endpoint</strong> [Example] For example, if bucket <code>examplebucket</code> is in the <code>ap-southeast-1</code> region, you can access the bucket through the following access domain names: examplebucket.obs.ap-southeast-1.myhuaweicloud.com</td>
<td>OBS allocates a default access domain name to each bucket. The access domain name is the Internet address of the bucket. You can directly access the bucket through the domain name when, for example, developing cloud applications or sharing data.</td>
</tr>
</tbody>
</table>

2. **Endpoint access requests**

OBS provides different endpoints for different regions. How does a user select an appropriate endpoint?

Generally, the endpoint carried in a request for accessing OBS must be the endpoint of the region where the requested resource resides. However, in some special cases, you can use any endpoint.

a. **Scenarios when the endpoint in request must be consistent with the endpoint of the region where the requested resources reside**

   When accessing a bucket or an object, the endpoint in the request must be the endpoint of the region where the bucket or object resides. For example, if bucket `mybucket` is in region `cn-north-4`, you can list objects in the bucket by referring to the following example:

   **A correct example of request and response for listing objects:**
If the endpoint in the request is not consistent with the endpoint of the requested region, an error message is returned indicating that the bucket does not exist.

In the preceding example, bucket `mybucket` is in region `cn-north-4`. If you use the `cn-south-1` endpoint (`mybucket.obs.cn-south-1.myhuaweicloud.com`) to access the bucket, HTTP 404 is returned, indicating that the bucket does not exist. In this case, you can call the API of **Obtaining Bucket Location** to obtain the bucket's region ID, and then re-send the request.

**An incorrect example of request and response for listing objects:**

**[Request]**

GET / HTTP/1.1  
Host: mybucket.obs.cn-south-1.myhuaweicloud.com  
Accept: */*  
Date: Thu, 10 Mar 2016 08:51:25 GMT  
Authorization: authorization

**[Response]**

HTTP/1.1 404 NoSuchBucket  
x-obs-request-id: 0001EF710C000001S36176DA465E4E6G  
x-obs-id-2: Rdj0ZzVcRkikHcjcCQqjKDTG8JuAgj2GuLIP7Pv/cYYpIsS0xTFJQHPSv5g5yOYC  
Date: Thu, 10 Mar 2016 08:51:30 GMT  
Content-Length: 0

b. **Scenarios when any endpoint can be used in a request**

For APIs to obtain the list of buckets and obtain a bucket region information, the endpoint in the request can be any regional endpoint, because these two APIs search for the requested buckets in all regions.
An example of request and response for obtaining the bucket region information:
For example, if bucket `mybucket` is in region `cn-north-4` and the endpoint of the `cn-south-1` region is used in the request, the bucket location information can still be obtained.

[Request]
```
GET /?location HTTP/1.1
Host: mybucket.obs.cn-south-1.myhuaweicloud.com
Accept: */*
Date: Thu, 10 Mar 2016 08:51:25 GMT
Authorization: authorization
```

[Response]
```
HTTP/1.1 200 OK
x-obs-request-id: 0001EF710C000001536176DA465E4E6G
x-obs-id-2: Rdj0zVrkhi8cijcQqjkdGt8JuAgi12CGUij7Pv/cYYp9S0XTFIO5Pv5Sg5yOYC
Content-Type: application/xml
Date: Thu, 10 Mar 2016 16:58:12 GMT
Content-Length: length

<?xml version="1.0" encoding="UTF-8" standalone="yes">
<Location xmlns="http://obs.myhuaweicloud.com/doc/2015-06-30/">
  cn-north-4
</Location>
```

1.2.5 REST API

REST stands for Representational State Transfer. API stands for application programming interface. The REST API is a communication interface that is in compliance with the HTTP protocol and REST architecture style. With the REST architecture, each Uniform Resource Identifier (URI) represents a resource, so the web address must not contain verbs. A web resource can be a segment of text, an image, an audio file, a service, or any resource entity.

Characteristics of the REST architecture are as follows:

1. Each URI represents a resource.
2. The resource is transferred between a client and the server in a certain form.
3. The client uses four HTTP request methods (PUT, GET, POST, and DELETE) to perform operations on server resources, and thus to implement the Representational State Transfer.

In REST, specific information or data on a network is represented by a resource, which is referenced with a uniform resource identifier (URI). Clients on a network can locate resources using uniform resource locators (URLs).

OBS uses buckets, objects, and their parameters to locate specific URL. Use URLs when you want to operate resources.

The following provides a common URL format. The parameters in square brackets ([ ]) are optional.

```
protocol://[bucket.]domain[:port][/object]?param
```
Table 1-2 URL parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>Protocol used for sending requests, which can be either HTTP or HTTPS. HTTPS is a protocol that ensures secure access to resources. OBS supports both HTTP and HTTPS.</td>
<td>Yes</td>
</tr>
<tr>
<td>bucket</td>
<td>Resource path of a bucket, identifying only one bucket in OBS.</td>
<td>No</td>
</tr>
<tr>
<td>domain</td>
<td>Domain name or IP address of the server for saving resources</td>
<td>Yes</td>
</tr>
<tr>
<td>port</td>
<td>Port enabled for protocols used for sending requests. The value varies with different servers deployed. By default, the default port is used. Each transmission protocol has a default port. For example, the default HTTP port number is 80, and the default HTTPS port number is 443. In OBS, HTTP port number is 5080 and that of HTTPS is 5443.</td>
<td>No</td>
</tr>
<tr>
<td>object</td>
<td>An object path used in the request.</td>
<td>No</td>
</tr>
<tr>
<td>param</td>
<td>A specific resource contained by a bucket or object. If this parameter is omitted, the bucket or object itself is obtained.</td>
<td>No</td>
</tr>
</tbody>
</table>

A request must be in HTTP 1.1 format. A request header must contain OBS-defined parameters, for example, signature information.

HTTP supports several HTTP request methods, such as GET, PUT, POST, DELETE, HEAD, and OPTIONS. A request method indicates how to access specific resources. Table 1-3 describes the request methods supported by the OBS REST API.

Table 1-3 HTTP request methods supported by the OBS REST API

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Requests the server to return a specific resource, for example, a bucket list or object.</td>
</tr>
<tr>
<td>PUT</td>
<td>Requests the server to store a specific resource, for example, a bucket or object.</td>
</tr>
<tr>
<td>POST</td>
<td>Requests the server to store a special resource or perform a special operation, for example, part uploading or merging.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Requests a server to delete specified resources, for example, an object.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD</td>
<td>Requests the server to return the digest of a specific resource, for example, object metadata.</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>The request server checks whether the user has the operation permission for a resource. The CORS needs to be configured for the bucket.</td>
</tr>
</tbody>
</table>

The server returns a status code and reminder information to the user. **Table 1-4** describes the common error codes:

<table>
<thead>
<tr>
<th>Response Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2xx</td>
<td>Indicates that the server has successfully returned the requested data.</td>
</tr>
<tr>
<td>4xx</td>
<td>Indicates that the request sent from the client is incorrect, so the server does not create or modify data.</td>
</tr>
<tr>
<td>5xx</td>
<td>Indicates that an error occurs on the server, and the user does not know whether the request has been successfully sent.</td>
</tr>
</tbody>
</table>

For details about OBS errors, see the *Object Storage Service API Reference*.

### 1.3 Comparison Between OBS and File Systems

This section compares the features and applicable scenarios of OBS and file systems, helping you select the service appropriately.

- **Features and Application Scenarios**
  The file system has the following features:
  a. Standard file protocol: You can mount file systems to servers, the same as using the local directory.
  b. Data sharing: The same file system can be mounted to multiple servers, so that access to data can be shared.
  c. Private network: Data access must be performed in the internal network of the data center.
  d. The capacity of a single file system is high (at PB level) and the performance is excellent (at ms level), mainly for media editing, HPC, and file sharing scenarios.

OBS has the following features:
  a. Provides the standard REST API. You need to use programming or third-party tools to access object storage.
b. Data sharing: servers, embedded devices, IoT devices can use the same path to access shared object data in OBS.

c. Public/private networks: Allows data to be accessed from public networks to meet Internet application requirements.

d. Unlimited capacity and high performance (10ms): Most suited for Web/Mobile, backup/archiving, and big data/IoT scenarios.

- API Function Comparison

The following table compares a bucket in OBS with a directory in the file system. OBS does not have the hierarchical directory structure.

Table 1-5 Comparison of the API between OBS and the file system

<table>
<thead>
<tr>
<th>OBS</th>
<th>File System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Service</td>
<td>Obtain the directory list.</td>
</tr>
<tr>
<td>Get Bucket</td>
<td>Obtain the file list.</td>
</tr>
<tr>
<td>Put Object</td>
<td>Write data to a file.</td>
</tr>
<tr>
<td>Append Object</td>
<td>Append data to a file.</td>
</tr>
<tr>
<td>Get Object</td>
<td>Read a file.</td>
</tr>
<tr>
<td>Delete Object</td>
<td>Delete a file.</td>
</tr>
<tr>
<td>Not supported</td>
<td>Modify the file contents.</td>
</tr>
<tr>
<td>Copy Object (copy the same name object in the same bucket), Put ACL</td>
<td>Modify properties of a file.</td>
</tr>
<tr>
<td>Copy Object</td>
<td>Copy a file.</td>
</tr>
</tbody>
</table>

- Precautions for Mapping File Systems to OBS

Many tools can map buckets to file systems. An object name can be mapped to a multi-level directory. For example, the object name test/ refers to a directory, and the object name test/test1.jpg refers to a file in the directory. Function simulation (such as renaming a directory) is resource consuming. Therefore, avoid performing such operations in a simulated file system. Try to simply write new files, delete files, read and write files.

1.4 List of Features

The requests that are sent by users to OBS must comply with REST specifications and contain required header parameters. If a request is successfully processed, OBS returns a success response. If the request fails to be processed, OBS returns an error response that contains the cause of the error. As containers for data storage in OBS, buckets have the following features:
Table 1-6 OBS features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic bucket operations</td>
<td>Creating buckets, deleting buckets, obtaining bucket lists, and listing objects in a bucket</td>
</tr>
<tr>
<td>Basic object operations</td>
<td>uploading objects, downloading objects, copying objects, and uploading an object in multiple parts.</td>
</tr>
<tr>
<td>Control permissions</td>
<td>Object ACL, bucket ACL, and bucket policies</td>
</tr>
<tr>
<td>Data transmission</td>
<td>OBS supports the data transfer protocols.</td>
</tr>
<tr>
<td>Data protection mechanisms</td>
<td>Bucket versioning control and server-side encryption</td>
</tr>
<tr>
<td>Storage classes</td>
<td>This section describes the storage classes of buckets and objects.</td>
</tr>
<tr>
<td>Object lifecycle management</td>
<td>Guides and instructions on object lifecycle management</td>
</tr>
<tr>
<td>Static website hosting</td>
<td>Guides and instructions on static website hosting</td>
</tr>
<tr>
<td>Cross-region replication</td>
<td>Guides and instructions on cross-region replication</td>
</tr>
<tr>
<td>Event notification</td>
<td>Sends event notifications through SMN.</td>
</tr>
<tr>
<td>Logging</td>
<td>This topic describes how to manage bucket logs.</td>
</tr>
<tr>
<td>Monitoring</td>
<td>This topic describes how to view OBS monitoring metrics.</td>
</tr>
</tbody>
</table>

OBS has two generations of hardware architecture, OBS 2.0 and OBS 3.0. Any newly created bucket will be stored in OBS 3.0 by default, but the previously created buckets are still stored in OBS 2.0. OBS 3.0 provides more features than the OBS 2.0, Table 1-7 shows the differences.

Table 1-7 Comparison of features between OBS 3.0 and OBS 2.0

<table>
<thead>
<tr>
<th>Feature</th>
<th>OBS 3.0</th>
<th>OBS 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object storage classes</td>
<td>Available</td>
<td>N/A</td>
</tr>
<tr>
<td>Federated authentication</td>
<td>Available</td>
<td>N/A</td>
</tr>
<tr>
<td>IAM agency</td>
<td>Available</td>
<td>N/A</td>
</tr>
<tr>
<td>Image processing</td>
<td>Available</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-region replication</td>
<td>Available</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Other features and basic functions that are not listed here are available with both OBS 2.0 and OBS 3.0.
You can use OBS Console or the HeadBucket operation to check whether the bucket is stored in OBS 2.0 or OBS 3.0. The checking methods are listed as follows:

Method 1: Log in to OBS Console and check the bucket summary.

If the **Bucket Version** is **3.0**, the bucket is stored in OBS 3.0. If no value is displayed for **Bucket Version**, the bucket is stored in OBS 2.0.

Method 2: Send the API request (HeadBucket) to check the bucket version.

Example request

```
HEAD / HTTP/1.1
Host: bucketname.obs.region.myhuaweicloud.com
Accept: */*
Date: Thu, 07 Jun 2018 02:23:25 GMT
Authorization: auth string
```

Example response:

```
HTTP/1.1 200 OK
Server: OBS
x-obs-request-id: BF2600000163D80E4C5F20FDD5BD0085
Content-Type: application/xml
x-obs-version: 3.0
x-obs-id-2: 32AAAQAAEAABABAABAAAQAAEAABAAAQAAEAABCS8w5mI00I4oMWhmdnIV7XmAvfewrQq
Date: Thu, 07 Jun 2018 02:23:25 GMT
Content-Length: 0
```

In this response, **x-obs-version: 3.0** indicates that the bucket is stored in the OBS 3.0. If this header does not exist or the value of this header is displayed otherwise, the bucket is stored in the OBS 2.0.
2 Creating Requests

2.1 Obtaining Permanent Access Keys (AK/SK)

2.2 Obtaining an Endpoint

2.3 Constructing a Request

2.1 Obtaining Permanent Access Keys (AK/SK)

OBS provides a set of REST API operations. You can develop your own applications to access OBS by using the software development kits (SDKs) or the REST API provided by OBS. It is strongly recommended that the SDK be used to complete client interconnection development, which simplifies client software implementation and provides better interface compatibility and stability.

The REST API of OBS supports both authenticated requests and anonymous requests. Anonymous requests are typically used only for scenarios that require public access, such as static website hosting. In most scenarios, authenticated requests can successfully access OBS resources. An authenticated request must contain a signature value. The signature value is calculated based on the requestor's access key as the encryption factor and the specific information carried in the request body. The process of calculating the signature is included in the SDK. You only need to set the access key in the SDK initialization phase. The signature calculation is implemented automatically. However, if the client uses the REST API to develop a program to access OBS, the client needs to calculate the signature based on the signature algorithm defined by the OBS and add the signature to the request.

In public cloud services, users are managed in a unified manner. Register an account on IAM and create a pair of access keys. The following is an example of a pair of access keys:

- **AK**: indicates the ID of the access key. It is the unique ID associated with a secret access key. The AK and SK are together used to obtain an encrypted signature for a request. Format example: \textbf{HCY8BGCN1YM5ZYYOK1MH}

- **SK**: indicates the secret access key used together with the access key ID to sign requests. AK and SK can be used together to identify a request sender to prevent the request from being modified. Format example: \textbf{9zywfluabSQQY0JTNfJQDG7vffqYBaTdXde2GUcq}
The AK uniquely identifies an IAM user of the public cloud. The OBS determines the identity of the user based on the AK and checks the permission.

For details about how to obtain the permanent access key, see Creating Access Keys (AKs and SKs).

Accessing OBS Through Temporary Authentication

Create a pair of temporary AK/SK and security token to access OBS:

You can obtain temporary AK/SK and security token on IAM, and then access OBS. In this way, you can assign a third-party application and a sub-user the temporary credentials within a validity period to access OBS.

You can obtain the temporary AK/SK and security token by calling the IAM API Obtaining a Temporary AK/SK.

This API uses the user's token or an agency token to obtain temporary AK/SK and security token. The authorization complies with the least privilege principle. If you want to access OBS through temporary authentication, you need to use the temporary AK/SK and security token together, and add the x-obs-security-token field to the request header.

2.2 Obtaining an Endpoint

For details about the concept of endpoints, see 1.2.4 Endpoints and Access Domain Names.

Endpoints information:

For details about OBS regions and endpoints, see Regions and Endpoints.

2.3 Constructing a Request

2.3.1 Creating a Request Using the REST API

Constructing an HTTP Request

An HTTP request consists of four parts: request method, request URL, request header, and request body.

Method

The following table describes the request methods supported by OBS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Requests the server to return a specific resource, for example, a bucket list or object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PUT</td>
<td>Requests the server to store a specific resource, for example, a bucket or object.</td>
</tr>
<tr>
<td>POST</td>
<td>Requests the server to store a special resource or perform a special operation, for example, part uploading or merging.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Requests a server to delete specified resources, for example, an object.</td>
</tr>
<tr>
<td>HEAD</td>
<td>Requests the server to return the digest of a specific resource, for example, object metadata.</td>
</tr>
<tr>
<td>OPTION</td>
<td>The request server checks whether the user has the operation permission for a resource. The CORS needs to be configured for the bucket.</td>
</tr>
</tbody>
</table>

**URL**

Example URL: `<schema>://<bucketname>.<endpoint>/<objectname>?querystrings`

- Schema: HTTP or HTTPS
- bucketname: name of the bucket
- Endpoint: Domain name of the region where the bucket resides
- objectname: name of the object
- querystrings: query parameters

---

**NOTICE**

All API requests except those for the bucket list must contain the bucket name. Based on the DNS resolution performance and reliability, OBS requires that the bucket name must be placed in front of the **EndPoint** when a request carrying a bucket name is constructed to form a three-level domain name, also mentioned as virtual hosting access domain name.

For example, you have a bucket named **test-bucket** in the **ap-southeast-1** region, and you want to access the ACL of an object named **test-object** in the bucket. The endpoint of the **ap-southeast-1** region is **obs.ap-southeast-1.myhuaweicloud.com**, the correct URL is **https://test-bucket.obs.ap-southeast-1.myhuaweicloud.com/test-object?acl**.

**Request Header**

HTTP headers used by OBS are classified into the following types:

- Standard HTTP header
- OBS-defined header

The user-defined header that is always prefixed with **x-obs-**, such as **x-obs-date** and **x-obs-acl**.
Each REST API operation defines some optional headers and mandatory headers. Some headers need to be involved in the request signature calculation. For details about the differences API operations, see the Object Storage Service API Reference.

Request Message Body

Not all API operations need to carry the request body. Generally, only the operations defined by the PUT and POST methods need to carry the request body. The request body defined by the REST API of OBS is generally expressed in the format of XML, JSON, or binary data. For details about the differences between operations, see the Object Storage Service API Reference.

Adding a Signature to a Request

A signature can be added to a request in three forms.

Table 2-2 Forms of a signature in a request

<table>
<thead>
<tr>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
</table>
| Signature contained in the header | Usually, a signature is contained in a header. In this case, an authentication header field needs to be added to the HTTP request header in the following format: Authorization: OBS AccessKeyId:signature  
Example: Authorization: OBS HCY8BGCN1YM5ZWYOK1MH:51KOKYY9CVvk/DtUt4W8aFHVfsc= |
| Signature contained in the URL | The querystring of the URL can contain a signature, which is generally used for time-limited sharing links. For example:  
https://bucketname.obs.region.myhuaweicloud.com/objectname?AccessKeyId=HCY8BGCN1YM5ZWYOK1MH&Expires=1528857302&Signature=OgynyFAkBRU6FUHkEqpMVsPolok= |
| Signature contained in the POST table | The POST table can contain a signature, which is generally used in browser upload scenarios. For example:  
-----WebKitFormBoundaryx4WluntRxFeG3GAD  
Content-Disposition: form-data; name="AccessKeyId"  
HCY8BGCN1YM5ZWYOK1MH  
-----WebKitFormBoundaryx4WluntRxFeG3GAD  
Content-Disposition: form-data; name="signature"  
Bpj82MEqa23d7GEot7LCqJY8Zdg= |

For details about how to calculate the signature, see Signing a Request.

STS Temporary Access to OBS

- For details about how to authenticate the signature in a header, see Authentication of Signature in a Header.
- For details about how to authenticate the signature in a query parameter, see Authentication of Signature Carried in a Query Parameter.
• For details about how to authenticate the signature in the table uploaded by a browser, see Authentication of Signature Carried in the Table Uploaded Through a Browser.
3 Using Buckets

3.1 Use Restrictions

3.2 Creating a Bucket

3.3 Deleting a Bucket

3.4 Managing Bucket Quotas

3.5 Managing Bucket Tags

3.1 Use Restrictions

Bucket Names

The bucket name is used as a part of the access domain name and needs to be parsed. Therefore, the bucket name must meet the DNS domain name specifications. When receiving a request for creating a bucket, the OBS strictly checks the bucket name. A bucket name must meet the following rules:

- Contains only lowercase letters, digits, hyphens (-), and periods (.)
- Starts with a digit or a letter.
- Ranges from 3 to 63 characters.
- Cannot be an IP address.
- Cannot end with a hyphen (-) or period (.)
- Cannot contain two consecutive periods (..).
- Cannot contain a combination of a period and a hyphen (- or -). For example, my-.bucket and my.-bucket are invalid.
When the HTTPS protocol is used for OBS access, the SSL wildcard certificate matches only the bucket that does not contain a period (.). As a result, when a bucket whose name contains a period (.) accesses the OBS system, the client displays a message indicating that the bucket is risky. For example, a red alarm may be displayed in the browser security prompt. Therefore, do not include a period (.) in the bucket name. Alternatively, you can rewrite the certificate verification logic on the client to avoid this problem.

**Bucket Name Uniqueness**

Bucket names are globally unique. OBS does not allow two buckets with the same name in any region. If a bucket does not include any object or part, you can delete this bucket and create another one with the same name. However, because the bucket name is a part of the access domain name and the cache expiration time may be different for each level of domain name resolution, the bucket name may be available after a period of time. In addition, if someone uses the bucket name before you, you cannot use this name any longer. Therefore, exercise caution when deleting a bucket and re-creating a bucket with the same name.

**Notice**

After a bucket is created, the bucket name cannot be changed.

**Bucket Region**

Bucket regions are planned based on geographical proximity, cost reduction, and legal compliance. A bucket must belong and can only belong to one region.

**Notice**

After a bucket is created, its region cannot be modified. If you select an incorrect region during initial planning, you must delete the bucket and then create it again, or enable a new bucket. Therefore, before creating a bucket, plan the bucket region with caution.

**Maximum Number of Buckets**

A maximum of 100 buckets can be created for each account. Buckets created by different users of an account occupy the quota of the account.

### 3.2 Creating a Bucket

After being authorized, a user can create buckets in OBS. A bucket belongs to only one account. If a bucket owner repeatedly creates the bucket, success messages are returned but no actions are performed. In other scenarios, if a bucket is created repeatedly, a message indicating a conflict is returned.
Before creating a bucket, plan the following:

- Bucket name
- Bucket region
- Default storage class
- Bucket access permissions
- Bucket cluster type

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket name</td>
<td>Select a bucket name by referring to Bucket Names.</td>
<td>Yes</td>
</tr>
<tr>
<td>AZ mode of a bucket</td>
<td>You can enable or disable the multi-AZ mode when creating an OBS bucket.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>After the multi-AZ mode is enabled, data is stored in multiple AZs with redundancy, improving reliability. If the multi-AZ mode is disabled, data is stored only in a single AZ, which is cheaper than the storage in the multi-AZ mode. Buckets created earlier without specifying the AZ mode adopt the single-AZ storage by default.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For details about multi-AZ pricing, see Product Pricing Details.</td>
<td></td>
</tr>
<tr>
<td>Bucket region</td>
<td>Select a region by referring to Regions and Endpoints.</td>
<td>Yes</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Bucket cluster type</td>
<td>If you have purchased the dedicated OBS cluster, you can specify the cluster type when creating a bucket. You can specify the dedicated cluster for a new bucket. Then the bucket and all objects in the bucket are stored in the dedicated cluster. You can also create a bucket in a public cluster. Then the bucket and all objects in the bucket are stored in the public cluster shared by all users. <strong>NOTE</strong> If the cluster type is not specified in a bucket creation request, the following bucket creation rules are applied: The bucket will be created in the dedicated cluster, if you have a dedicated cluster in the region where the bucket is to be created; The bucket will be created in a public cluster, if you do not have any dedicated cluster in the region where the bucket is to be created. <strong>NOTICE</strong> The cluster type of the bucket cannot be changed once the bucket is created.</td>
<td>No</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Default storage class</td>
<td>OBS allows you to specify the default storage class of a bucket when creating the bucket. If no storage class is specified for an object, OBS uses the default storage class of the bucket as that of the object. The charging rates and costs vary according to the storage class. You need to select a storage class based on service models. For details about storage classes, see <a href="#">8 Storage Classes</a>. <strong>NOTE</strong> If you do not specify the default storage class for a bucket, the standard storage class is used by default. <strong>NOTICE</strong> The default storage class of a bucket can be modified, but the modification takes effect only for newly uploaded objects.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Creating a Bucket Using OBS Console

For details about how to create a bucket using OBS Console, see the Console Operation Guide.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket access permissions</td>
<td>The access permission of a bucket can be controlled by bucket policies and ACL rules. When creating a bucket, you can use the predefined header field of OBS to set a default ACL. However, bucket policies cannot be set during bucket creation. For details about the functions of and differences between bucket policies and bucket ACLs, see 5 Permission Control. A predefined ACL is used in scenarios where public access or authorization to other specified tenants is required. For example, • x-obs-acl: public-read indicates that the read-only permission is granted to all users by default after the bucket is created. • x-obs-grant-read: id=bbac5b3bdc04674bad8b60a1eb885c4 indicates that the read-only permission is granted to the account whose ID is bbac5b3bdc04674bad8b60a1eb885c4 by default after the bucket is created. • For more predefined header fields, see Creating a Bucket. NOTE If you do not specify the access permission, the bucket is a private bucket and only the owner of the bucket has access permissions.</td>
<td>No</td>
</tr>
</tbody>
</table>
Creating a Bucket Using OBS API

Although using the API for development is not recommended, you can choose to do so in special scenarios. For example, when running the environment on which the SDK depends on is impossible on embedded board software, you can use the API.

Comply with the API description and signature rules when using the OBS API for development. For details about the interface description and usage, see User Signature Authentication and Creating a Bucket in the Object Storage Service API Reference.

**NOTICE**

If a 5xx error is returned from the server or the request times out during bucket creation, the system takes approximately 30 minutes to make bucket information consistent. During the process, bucket information may be inaccurate.

### 3.3 Deleting a Bucket

If a bucket is empty, you can delete the bucket. If the bucket is not empty, deleting the bucket is not allowed.

An empty bucket must meet the following requirements:

- There is no object or any historical version of an object in the bucket.
- There is no unmerged multipart upload task in the bucket.

**NOTE**

Before deleting a bucket, you can check whether the bucket is empty by listing objects and multipart tasks in the bucket. OBS supports the versioning feature. Therefore, before deleting a bucket, check whether the bucket is enabled with versioning and whether historical versions and delete marks exist. (A delete mark is considered as a historical version.)

### 3.4 Managing Bucket Quotas

By default, a newly created bucket has no quota. You can set a quota limit for a bucket to control the total capacity of objects that can be uploaded in the bucket. If the total capacity exceeds the configured object capacity, objects fail to be uploaded.

**NOTE**

A bucket quota can control object uploading only after the quota is set. If the bucket quota is smaller than the capacity of uploaded objects, existing objects will not be deleted, but new objects cannot be uploaded. In this case, you can upload new objects only after deleting some existing objects until the used space is less than the quota limit.

A bucket quota must be a non-negative integer expressed in bytes. The maximum value is $2^{63} - 1$.

OBS does not provide an API for deleting bucket quotas. You can set the bucket quota to 0 to cancel the quota limit.
Using OBS API to Manage Bucket Quotas

You can configure bucket storage quotas by calling the API. For details, see Configuring Bucket Storage Quota.

You can query bucket storage quotas by calling the API. For details, see Querying Bucket Storage Quota.

3.5 Managing Bucket Tags

Generally, your service system may use multiple cloud services of HUAWEI CLOUD. You can set tags for different resource instances of these cloud services (for OBS, a resource instance is a bucket). The CDRs generated by those services reflect these resource instances and the tags set on the instances. If your service system is composed of multiple applications, setting a unified tag for resource instances owned by an application can help you easily analyze the usage and cost of different applications.

Tag Usage Specifications

- A bucket can have up to 10 tags.
- Each tag contains a key and a value. The key of each tag in a bucket must be unique.
- The key contains 1 to 36 characters, including A to Z, a to z, 0 to 9, hyphens (-), underscores (_), and Unicode(\u4E00-\u9FFF) characters.
- The value contains 0 to 43 characters, including A to Z, a to z, 0 to 9, hyphens (-), underscores (_), periods (.), and Unicode(\u4E00-\u9FFF) characters.

NOTICE

After a bucket tag is set, the setting may take effect after several minutes due to the cache.
4 Using Objects

4.1 Basic Concepts of Objects

Objects are basic storage units in OBS. You can store multiple objects in one or more buckets. Relevant concepts of an object are as follows:

- **Object key**: Name of an object. You can obtain an object using the object name.
- **Object value**: data uploaded by users and stored in OBS
- **Metadata**: description information about object names and sizes
- **Version ID**: In a bucket with versioning enabled, there may be multiple objects with the same name but different versions. Each object is identified by the name and version ID.
- **Access control information**: access permission information about objects

4.2 Object Metadata

Each object has the data, key, and metadata. An object key (or key name) can uniquely identify an object in a bucket. When you create an object, specify an object key because it uniquely identifies the object in the bucket. A key name is a sequence of Unicode characters, and its UTF-8 code contains up to 1024 bytes.
Object metadata is a key-value pair. You can set the metadata of an object when you upload the object. After the object is uploaded, you cannot modify its metadata. The only way to modify object metadata is to create an object copy and set metadata.

**Guidelines on Naming Object Keys**

Although any UTF-8 characters can be used in an object key name, naming object keys according to the following guidelines can help maximize the object keys' compatibility with other applications. Ways to analyze special characters vary with applications. The following guidelines help object key names substantially meet the requirements of DNS, web security characters, XML analyzers and other APIs.

The following character sets can be freely used in key names.

- Uppercase (A to Z) and lowercase (a to z) letters and digits (0 to 9)
- Special characters including !-_.*'

The following are examples of valid object key names:

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

Avoid using the following characters in key names, because they require a lot of special processing to keep consistency across all applications.

- Backslash (`\`)
- Left brace (`{`) and right brace (`}`)
- Non-printable ASCII characters (128–255 decimal characters)
- Insert symbol (`^`)
- Percentage character (`%`)
- Accent/Untick (````) and tilde (`~`)
- Less than sign (`<`) and greater than sign (`>`)
- Hashtag (`#`) and vertical bar (`|`)

Note that the OBS data model is a layer structure. You create a bucket and the bucket stores objects. No sub-buckets or sub-folders exist in the structure. However, you can use key name prefixes and separators to deduce the logical structure like OBS Console does. The folder concept is available on OBS Console.

Assume that your bucket (`companybucket`) contains four objects with the following object keys:

- `bucket-log/log01.txt`
- `cgvbs/test1.txt`
- `2015-10-14_111756.png`
- `test1.txt`

OBS Console uses the key name prefix (`bucket-log/` and `cgvbs/`) and separator (`/`) to display the folder structure, as shown in the following figure.
The **2015-10-14_111756.png** and **key test1.txt** keys do not have prefixes. Therefore, the objects appear at the root level of the bucket. If you open the **cgvbs/** folder, you will see that it contains the **test1.txt** object.

### Object Metadata

Metadata includes system metadata and user-defined metadata.

System metadata: OBS reserves a set of system metadata for each object in a bucket. OBS deals with the system metadata based on requirements. For example, OBS reserves object metadata such as creation time and size, and the metadata is part of object management.

1. System metadata is classified into the following two types:
   - System-controlled metadata (such as the object creation date). Only OBS can change its value.
   - Other system metadata (such as the storage class configured for an object and the data that indicates whether encryption is enabled at the server end for an object). You can control values of such metadata. Assume that you own a bucket that is configured as a website and you may want to redirect a page.
request to another page or an external URL. In this example, the web page is the object in your bucket. The OBS stores the page redirection value as such system metadata whose value is controlled by you.

When creating an object, you can configure the values of these system metadata projects or update the values when needed.

### Table 4-1 Object metadata defined by the system

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Whether a User Can Change the Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Length</td>
<td>Object size, in bytes</td>
<td>No</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Date when the object was modified last time</td>
<td>No</td>
</tr>
<tr>
<td>Content-MD5</td>
<td>128-bit MD5 digest of the Base64 code of an object</td>
<td>No</td>
</tr>
<tr>
<td>x-obs-version-id</td>
<td>Object version. If versioning is enabled for a bucket, the bucket will specify a version ID for each added object. For more information, see content about object versioning.</td>
<td>No</td>
</tr>
<tr>
<td>x-obs-delete-marker</td>
<td>In the bucket where versioning is enabled, this Boolean value marks whether the object is a delete mark.</td>
<td>No</td>
</tr>
<tr>
<td>x-obs-storage-class</td>
<td>Storage class</td>
<td>No</td>
</tr>
<tr>
<td>x-obs-website-redirect-location</td>
<td>Redirecting requests of associated objects to other objects in the same bucket or external URLs</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2. Use-defined metadata

When uploading an object, you can also specify the metadata for the object. When you send a PUT or POST request to create an object, you provide the optional information in the form of key-value pairs. When you use REST APIs to upload objects, the optional user-defined metadata names must start with `x-obs-meta-` to differentiate them from other HTTP headers. When you use REST APIs to search for data elements, the prefix `x-obs-meta-` is returned.
NOTICE

- When you use REST APIs to search for metadata, OBS lists namesake (case insensitive) headers and separates them with commas (,) in the list. If some metadata contains unprintable characters, the characters are not returned. However, the header `x-obs-missing-meta` and the number of metadata items that cannot be printed are returned together.
- Do not use non-US-ASCII characters to avoid unnecessary troubles.

- OBS uses lowercase letters to store user-defined metadata. When you use REST and UTF-8 or send a POST request to upload objects based on browsers, key-value pairs must comply with US-ASCII.
- The maximum size of a PUT request header is 8 KB. In a PUT request header, the maximum size of user-defined metadata is 2 KB. User-defined metadata is a group of key-value pairs. You can calculate the size of user-defined metadata by summarizing the total number of bytes in the UTF-8 codes of all keys and values.

**OBS Storage Class**

User-defined metadata `x-obs-storage-class` reserved by the system is used to set and record object storage classes. OBS supports three storage classes. For details about the application scenarios of three storage classes, see [8 Storage Classes](#).

The tree storage classes are as follows:

- Standard
- Infrequent Access
- Archive

In OBS, you can specify a storage class when uploading and copying an object. Use `x-obs-storage-class` in the request header to set a storage class for the object.

You can also modify the storage class of an existing object. Modifying the storage class of an object is the same as that of modifying the user-defined metadata.

OBS also allows you to modify the storage class of an object by configuring a lifecycle management task. For details, see [9 Object Lifecycle Management](#).

### 4.3 Uploading an Object

#### 4.3.1 Overview of Object Upload Overview

Uploading an object is to add an object to a bucket. This operation requires the write permission. Objects uploaded by users are stored in buckets. Only the users who have the write permission can upload objects to buckets. The name of each object in a bucket must be unique.

OBS provides the following uploading modes based on the size of the object that you want to upload:
- Uploading an object in a single operation: You can upload an object not more than 5 GB by a PUT or POST operation.
- Multipart upload: You can upload an object up to 48.8 TB using multipart upload APIs.

Multipart upload APIs are used to improve the experience of uploading large objects. You can upload objects in multiple parts. These parts can be concurrently and independently uploaded in any sequence. Multipart upload suits objects smaller than 48.8 TB. You are advised to upload objects that are larger than 100 MB in multiple parts.

For more information about multipart upload, see 4.8 Multipart Upload. This section mainly describes uploading an object in a single operation.

After creating a bucket in OBS, you can upload objects to the bucket using a PUT or POST operation. A POST operation is mainly used for web browser-based upload.

### 4.3.2 Uploading an Object Using the PUT Method

Uploading an object is to add an object to a bucket. This operation requires the write permission. Objects uploaded by users are stored in buckets. Only the users who have the write permission can upload objects to buckets.

The name of each object in a bucket must be unique. If you upload an object with the same key as an existing object in the bucket, the new object will replace the existing object. To ensure that no data is corrupted during transmission, you can specify the value of the `Content-MD5` parameter in the request header. After receiving the request, the OBS will perform an MD5 consistency check. If the two MD5 values are inconsistent, the system returns an error message. You can also specify the value of the `x-obs-acl` parameter to configure an access control policy for the object.

After creating a bucket in OBS, you can upload an object to the bucket using the PUT method.

If a bucket has versioning enabled, the system automatically generates a unique version ID for the requested object in this bucket and returns the version ID in response header `x-obs-version-id`. If versioning is suspended for the bucket, the object version number is null.

The format of a PUT upload request is as follows:

```plaintext
PUT /ObjectName HTTP/1.1
Host: Host Server
Content-Type: type
Content-Length: length
Authorization: authorization
Date: date
<Optional Additional Header>
<object Content>
```

### 4.3.3 Uploading an Object Using the POST Method

You can also upload an object using the POST method. If the bucket versioning state is set to Enabled, the system automatically generates a unique version ID for the object. If the bucket versioning state is set to Suspended, the version ID of the object is null, which is returned in the response header `x-obs-version-id`.
The format of a POST upload request is as follows:

```plaintext
POST / HTTP/1.1
Host: bucketname.obsdomainName
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=-9431149156168
Content-Length: length
--9431149156168
Content-Disposition: form-data; name="key"
acl
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"
success_redirect
--9431149156168
Content-Disposition: form-data; name="content-Type"
content_type
--9431149156168
Content-Disposition: form-data; name="x-obs-meta-uuid"
uuid
--9431149156168
Content-Disposition: form-data; name="x-obs-meta-tag"
metadata
--9431149156168
Content-Disposition: form-data; name="AccessKeyID"
access-key-id
--9431149156168
Content-Disposition: form-data; name="policy"
encoded_policy
--9431149156168
Content-Disposition: form-data; name="signature"
signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename"
Content-Type: image/jpeg
file_content
--9431149156168
Content-Disposition: form-data; name="submit"
Upload to OBS
--9431149156168--
```

4.4 Copying an Object

This section describes how to copy an object in OBS. You can copy an object of up to 5 GB by an operation. To copy an object of larger than 5 GB, you must use multipart upload APIs. Specifically, the CopyObject operation allows you to:

- Create a copy for an object.
- Rename an object by creating a copy for it and deleting the source object.
- Change the metadata of an object. Every object has metadata which is a field-value pair. When you upload an object, you can set its metadata. After the object is uploaded, you cannot modify its metadata. The only way to modify object metadata is to create an object copy and set metadata. In the copy operation, set the target object to the same as the source object.

Each object contains metadata, including system metadata and user-defined metadata. Users can control some system metadata. In the event of copying an object, user-controlled system metadata and user-defined metadata are copied
too. OBS resets the metadata controlled by the system. For example, when you copy an object, the OBS resets the creation date of the copied object. In the copy request, you do not need to set such value.

When copying an object, you may want to update some metadata. For example, when the source object is configured to use standard storage, you may want to configure the object copy to using low-redundancy storage. You may also want to modify some user-defined metadata of the source object. If you want to modify metadata, even only one metadata, that can be configured by users (defined by users or the system), specify all the metadata that can be configured by users on the source object in the request.

**NOTICE**

When versioning is disabled for a bucket, if you copy object a as object b and an object named object b already exists, the new object b will overwrite the old object b. After the copying is executed successfully, only new object b can be downloaded because old object b has been deleted. Therefore, before copying an object, ensure that there is no object with the same name as the object copy to prevent data from being deleted mistakenly. During the copying, object a has no changes. You cannot determine whether a request is executed successfully only using **status_code** in the header returned by HTTP. If 200 in **status_code** is returned, the server has received the request and starts to process the request. The body in the response shows whether the request is executed successfully. The request is executed successfully only when the body contains Etag. Otherwise, the request fails to be executed.

For more information, see section **Copying Objects**.

### 4.5 Obtaining an Object

An OBS registered user or anonymous user with the read permission on an object can download the object content and metadata. You can download either an object or specified parts of an object.

- Downloading an object: Perform a single GET operation and the object stored in OBS can be returned.
- Downloading specified parts of an object: Use the Range HTTP header in the GET request to download a specific byte range of an object stored in OBS.

Once your application is ready, other parts of the object can be downloaded. When you only need parts of an object, this recoverable download function is very useful. This function is also convenient for you to respond to download failures due to poor network connections.

When you download an object, the metadata of the object is returned in the response header. Sometimes, you may want to overwrite a specific header value in the GET response. For example, you may want to overwrite the Content-Disposition response header in the GET request. The REST GET Object API allows you to specify the query string parameters in the GET request to overwrite the values.
If you want to obtain an object of a specific version from a bucket with the versioning function enabled, add the `versionId` parameter to the request.

For more information, see section Downloading Objects in the Object Storage Service API Reference.

### 4.6 Deleting an Object

You can delete one or more objects from the OBS system. You can delete either an object or multiple objects at once.

- Deleting an object: Use the deletion API to delete an object in an HTTP request.
- Deleting multiple objects: Use the batch deletion API to delete up to 100 objects in an HTTP request.

To delete an object from a bucket with versioning disabled, you need to specify the object name only. To delete an object in a specific version from a bucket with versioning enabled, you need to specify the version ID as well as the name of the object.

**Deleting an object from a bucket with versioning enabled**

If versioning is enabled for a bucket, an object in the bucket may have multiple versions. For such a bucket, the following options are provided by the object deletion API:

- A deletion request without specifying the version ID: The object name is specified but not the version ID.
  
  In this case, OBS only deletes the latest version of the object from the bucket. For a deleted version, OBS creates a Delete Marker for it and returns a new version ID. When an object of the latest version is listed, it cannot be queried because of its Delete Marker.

- A deletion request with the version ID specified: The object name is specified, as well as the version ID. In this case, either of the following results may occur:
  
  a. If the version ID is mapped to a specific object version, OBS deletes the object of that version.
  
  b. If the version ID maps to an object delete mark, OBS deletes this delete mark. After this operation, the object will reappear in the bucket.

If you want to make an API call to delete objects in the deleted object list, you need to make an API call to list objects that have multiple versions, and then specify the version IDs that you want to delete.

For more information, see section Deleting an Object in the Object Storage Service API Reference.

### 4.7 Batch Deleting Objects

You can send a `DELETE Multiple Objects` request to batch delete multiple objects from a bucket. OBS returns the deletion result of each object. Deleted objects cannot be restored.
The response modes for batch deleting objects are as follows:

- **Verbose:** The returned response includes the deletion result of each requested object in an XML file.
- **Quiet:** The returned response includes only results of objects that failed to be deleted.

OBS uses verbose mode by default and you can specify the quiet mode in the request body.

The request header of the batch deletion request must contain the Content-MD5 and Content-Length fields so that the server can detect whether an error occurs in the request message body during network transmission.

For more information, see section **Deleting Objects** in the *Object Storage Service API Reference*.

### 4.8 Multipart Upload

Multipart upload allows uploading a single object as a group of parts separately. Each part is a part of consecutive object data. You can upload object parts in any sequence or independently upload them. A part can be reloaded after an uploading failure, without affecting other parts. After all parts are uploaded, OBS merges these parts to create the object. Generally, if the size of an object reaches 100 MB, multipart upload is recommended. For example, you want to upload an object (500 MB) to an OBS bucket. In this case, you can use the tool OBS Browser to upload the object in multipart. The tool can automatically divide the object into multiple parts for uploading. Alternatively, you can make an API call for multipart upload, improving upload efficiency and reducing failures.

Multipart upload provides the following benefits:

- **Improving throughput:** You can upload parts in parallel to improve throughput.
- **Quick recovery from any network failures:** Small-size parts can minimize the impact of failed uploading caused by network errors.
- **Convenient suspension and resuming of object uploading:** You can upload parts at any time. A multipart upload does not have a validity period. You must explicitly complete or cancel the multipart upload.
- **Starting uploading before knowing the size of an object:** You can upload an object while creating it.

The multipart upload API allows uploading a large-size object in multiple parts. You can upload a new large-size object or create a copy of an existing object using this API.

The procedure for uploading multiple sections is as follows: Starting uploading (initializing the upload task), uploading parts, and completing uploading (merging the uploaded parts). Upon receiving a part merging request, OBS merges the uploaded parts to create an object. The object can be accessed like other objects.

You can list all the ongoing multipart upload tasks or obtain the list of uploaded parts of a specified multipart upload task. The following describes the detailed operations.
Initiating a multipart upload task

When you send a request to start multipart upload, OBS returns a response with the upload ID, which is the unique identifier of the multipart upload. This ID must be included in the request for uploading parts, listing uploaded parts, completing a multipart upload, or canceling a multipart upload.

Uploading parts

When uploading parts, you must specify the upload ID and part numbers. You can select any part number between 1 and 10,000. A part number uniquely identifies a part and its location in the object you are uploading. If the number of an uploaded part is used to upload a new part, the uploaded part will be overwritten. Whenever you upload a part, OBS returns the ETag header in the response. For each part upload task, you must record the part numbers and ETag values. These part numbers and ETag values are required in subsequent operations of completing the multipart upload task.

---

**NOTICE**

After the multipart upload task is initialized and one or more parts are uploaded, you must merge the parts or cancel the multipart upload task. Otherwise, you have to pay for the storage fee of the uploaded parts. OBS releases the storage and stops charging the storage fee only after the uploaded parts are merged or the multipart upload task is canceled.

When multiple concurrent upload operations are performed for the same part of an object, the server complies with the Last Write Win policy, but the time referred in Last Write is the time when the part metadata is created. To ensure data accuracy, the client must be locked during the concurrent upload for the same part of an object. Concurrent upload for different parts of an object does not require the client to be locked.

Copying parts

After creating a multipart upload job, you can specify upload IDs and upload parts for the specified upload task. You can also call the API for part copying to add parts. A part of an object or the whole object can be copied as a part.

---

**NOTICE**

You cannot determine whether a request is executed successfully only using status_code in the header returned by HTTP. If 200 in **status_code** is returned, the server has received the request and starts to process the request. The body in the response shows whether the request is executed successfully. The request is executed successfully only when the body contains Etag. Otherwise, the request fails to be executed.

If you copy the source object as a part called part1 and another part1 already exists before the copy operation, the original part1 will be overwritten by the new one after the copy operation. After the copy succeeds, only the new part1 is
displayed. Data of the old part 1 will be deleted. Therefore, ensure that the target part does not exist or has no value when calling the interface for part copying. Otherwise, data may be deleted by mistake. The source object in the copy process does not change.

**Merging parts and canceling a multipart upload task**

When merging parts, OBS creates an object by standardizing multiple parts in ascending order. If any object metadata is provided in the initialization of a part upload task, OBS associates the metadata with the object. After the multipart upload is complete, the parts will no longer exist. A part merging request must contain the upload ID, part numbers, and a list of corresponding ETag values. OBS responses include the ETag that uniquely identifies composite object data. The ETag is not the MD5 hash value of the object data. You can cancel a multipart upload task. After a multipart upload task is canceled, the upload ID cannot be used to upload any part. Then, OBS releases the storage of all uploaded parts. If you stop an ongoing multipart upload, the uploading will still complete (the result can be successful or failed). To release the storage capacity occupied by all uploaded parts, cancel the multipart upload after the entire task is complete.

---

**NOTICE**

If 10 parts are uploaded but only nine parts are merged. The part that has not been merged will be automatically deleted by the system. Parts that are not merged cannot be restored after being deleted. Before merging the parts, adopt the interface used to list the parts that have been uploaded to check all parts to ensure that no part is missed.

**Listing uploaded parts**

You can list a specified multipart upload task or the parts of all the multipart upload tasks in progress. Information about uploaded parts in a specific multipart upload will be returned for a request to list uploaded parts. For each request to list uploaded parts, OBS returns information about uploaded parts in the specific multipart upload. Information about a maximum of 1000 parts can be returned. If more than 1000 parts are uploaded in a multipart upload, you need to send multiple requests to list all uploaded parts. It is worth mentioning that the list of uploaded parts does not include merged parts.

---

**NOTICE**

A returned list can only be used for verification. After a multipart upload is complete, the result in the list is no longer valid. However, when part numbers and the ETag values returned by OBS are uploaded, the list of part numbers specified by the user will be reserved.

**Listing multipart upload tasks**

You can list all the ongoing multipart upload tasks. Ongoing multipart upload tasks refer to those tasks that have been started but not completed or aborted. For each request, a maximum of 1000 multipart upload tasks will be returned. If
there are more than 1000 ongoing multipart upload tasks, a user needs to send multiple requests to list all tasks.

<table>
<thead>
<tr>
<th>Item</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum object size</td>
<td>48.8 TB</td>
</tr>
<tr>
<td>Maximum number of parts for each upload task</td>
<td>10 000</td>
</tr>
<tr>
<td>Part number</td>
<td>1 to 10 000 (included)</td>
</tr>
<tr>
<td>Part size</td>
<td>The part size is between 5 MB to 5 GB. The size of the last part is between 0 bytes to 5 GB.</td>
</tr>
<tr>
<td>Maximum number of parts in a returned list</td>
<td>1 000</td>
</tr>
<tr>
<td>Maximum number of parts in a request</td>
<td>1 000</td>
</tr>
</tbody>
</table>

**Multipart upload operations and permissions**

You can perform multipart upload only after being granted with the permission. You can use ACLs, bucket policies, or user policies to grant users the permission. The following table lists multipart upload operations and the required permissions that can be granted by ACLs, bucket policies, or user policies.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Required Permission</th>
</tr>
</thead>
</table>
| Initiating a multipart upload task | To perform this operation, you need to have the PutObject permission.  
You must have the PutObject bucket owner to allow others to perform the PutObject operation. A bucket owner can allow trustees to perform the PutObject operation. |
| Uploading parts                  | To perform this operation, you need to have the PutObject permission.  
Only the initiator of a multipart upload can upload parts. The bucket owner must grant the multipart upload initiator the PutObject permission so that the initiator can upload parts of the object. |
<table>
<thead>
<tr>
<th>Operation</th>
<th>Required Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copying parts</td>
<td>To perform this operation, you need to have the PutObject permission as well as the GetObject permission on the object to be copied. Only the initiator of a multipart upload can copy parts. The bucket owner must grant the multipart upload initiator the PutObject permission so that the initiator can upload parts of the object.</td>
</tr>
<tr>
<td>Merging parts</td>
<td>To perform this operation, you need to have the PutObject permission. Only the initiator of a multipart upload can merge parts. The bucket owner must grant the multipart upload initiator the PutObject permission so that the initiator can complete multipart upload.</td>
</tr>
<tr>
<td>Canceling a multipart upload task</td>
<td>To perform this operation, you need to have the AbortMultipartUpload permission. By default, only the bucket owner and multipart upload initiator have this permission. In addition to the default configuration, the bucket owner can allow trustees to perform this operation. The bucket owner can also deny any trustees performing this operation.</td>
</tr>
<tr>
<td>Listing uploaded parts</td>
<td>To perform this operation, you need to have the ListMultipartUploadParts permission. By default, the bucket owner can list the uploaded parts of any multipart upload to the bucket. The multipart upload initiator can list the uploaded parts of a specific multipart upload. In addition to the default configuration, the bucket owner can allow trustees to perform this operation. The bucket owner can also deny any trustees performing this operation.</td>
</tr>
<tr>
<td>Operation</td>
<td>Required Permission</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Listing multipart upload tasks</td>
<td>To list multipart upload tasks to the bucket, you need to have the ListBucketMultipartUploads permission. In addition to the default configuration, the bucket owner can allow trustees to perform this operation.</td>
</tr>
</tbody>
</table>

**REST APIs applicable to multipart upload**

The following sections in the *Object Storage Service API Reference* describe REST API operations relevant to multipart upload.

- ListBucketMultipartUpload
- InitiateMultipartUpload
- UploadPart
- UploadPart-Copy
- ListParts
- CompleteMultipartUpload
- AbortMultipartUpload

### 4.9 Appending Data to an Object

The AppendObject operation adds data to the end of an object in a specified bucket. If there is no namesake object in the bucket, a new object is created. The object created using the AppendObject operation is an appendable object, and the object uploaded using the PUT operation is a normal object. Objects uploaded by users are stored in buckets. Only the users who have the write permission can upload objects to buckets. The name of each object in a bucket must be unique. To ensure that data is not damaged during transmission, you can add the `Content-MD5` parameter to the request header. After receiving the data, OBS performs MD5 verification for the data. If the data is inconsistent, OBS returns an error message. This operation allows you to specify the `x-obs-acl` parameter when creating an appendable object and set the permission control policy for the object. This operation supports server-side encryption.

**Relationship with other operations:**

1. If you perform the PUT operation on an existing appendable object, the appendable object is overwritten by the newly uploaded object and the object type changes to normal. If you perform the other way around, an error occurs.

2. An appendable object will be changed to a normal object after being copied. An appendable object cannot be copied and saved as an appendable object.
1. The last modification time of the object is updated each time an appending upload is performed.

2. If the SSE-C encryption mode is used on the server side, the appending upload is the same as the initialization segment. In this case, the request headers such as `x-obs-server-side-encryption` must be carried.

3. For the server-side encryption (SSE-KMS), the request header such as `x-obs-server-side-encryption` is specified only when the file is uploaded for the first time and no object with the same name exists in the bucket.

The format of an AppendObject request is as follows:

```
POST /ObjectName?append&position=Position HTTP/1.1
Host: bucketname.obs.region.myhuaweicloud.com
Content-Type: application/xml
Content-Length: length
Authorization: authorization
Date: date
<Object Content>
```

NOTE

1. The last modification time of the object is updated each time an appending upload is performed.

2. If the SSE-C encryption mode is used on the server side, the appending upload is the same as the initialization segment. In this case, the request headers such as `x-obs-server-side-encryption` must be carried.

3. For the server-side encryption (SSE-KMS), the request header such as `x-obs-server-side-encryption` is specified only when the file is uploaded for the first time and no object with the same name exists in the bucket.
5 Permission Control

5.1 Overview of Permission Control
An OBS bucket belongs to a specific account. The OBS performs permission verification on each access request received, in order to ensure that the bucket and object can be accessed only by authorized users. You can control requesters' permissions to access requested resources (buckets or objects). Requesters can only access resources after being granted permissions.

A bucket or object is created by a user who is the owner of the bucket or object. By default, only the owner has the permission to access the bucket or object. To access the bucket or object, other users must obtain the access permission of the bucket or object. The owner can manage permissions for the bucket or object using the ACL or bucket policy.

5.2 OBS Permission Control Model
OBS resources (buckets and objects) are private resources by default. Only owners have the permission to access their resources in OBS. The resource owner is the account who creates the resource. As illustrated in the following figure, an account has multiple IAM users. The bucket created by IAM user1 belongs to the account. If an IAM user2 uploads an object to the bucket, the object also belongs to the account.
OBS permission control refers to granting permissions to other tenants or IAM users by editing access policies. For example, if you have a bucket, you can authorize another IAM user to upload objects to your bucket. You can also open buckets to non-public cloud users, that is, buckets are public resources that can be accessed by anyone on the Internet.

Access control policies describe the resources that can be accessed by users. You can use the following two access control policies to control the permissions for your resources in OBS:

- OBS provides ACLs and bucket policies to implement resource-based permission control.
Each bucket and object has an ACL. Even if no ACL is configured, the OBS generates a default ACL for each object. The following is an example:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <Owner>
    <ID>resource owner id</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>resource owner id</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
      <Delivered>false</Delivered>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The ACL displays the information of the resource owner, as well as the list of grantee and the granted permissions. As described in the preceding example, the default ACL of a resource only grants the full control permission to the resource owner, and does not grant any permission to any other users.

In addition, you can leverage the bucket policy to grant other tenants or IAM users the permissions to access your buckets and objects. With the bucket policy, you can grant accounts or users with permissions for only objects created by the bucket owner. The bucket policy, as a supplement to ACL (or replace the ACL in many scenarios), allows more flexible and fine-grained permission control.

```json
{
  "Statement":[
    {
      "Sid": "granteReadObject",
      // More permissions...
    }
  ]
}
```
The preceding is an example of a bucket policy. You need to use the JSON format to compile the bucket policy. This policy grants all users with the permission to read all objects in a bucket named `bucketname`.

The account or IAM user ID in the resource policy can be queried through the public cloud console or the API operation of **Listing Users**. To query the user ID on the public cloud console, perform the following steps:

**Step 1** Log in to the console, click the username in the upper right corner, and select **My Credential**.

**Step 2** On the **My Credential** page, you can view information such as the username and user ID.

---End

- IAM provides user permission control based on IAM user roles.

After an IAM user is created, you need to add the user to an IAM group. IAM can grant the required permissions to the user group. OBS related permissions are classified into the following types (see **5.5 IAM User Policy (IAM Policy)**):

### 5.3 ACL

#### 5.3.1 Introduction to the ACL

An ACL can be used to manage bucket and object access permissions. A default ACL is generated when a bucket or object is created. You can make an API call to generate a new ACL. Each item in the ACL list contains the permissions granted to the grantee. To implement simple and practical authorization for users, the OBS ACL has the following features:

- The ACL takes effect for both the account and the users under the account.
- When the owner of a bucket is the same as the owner of an object, the ACL configured on the bucket takes effect on the bucket and objects in the bucket by default.
- An ACL can be carried when a bucket is created, or an ACL can be configured after a bucket is created. An object can carry an ACL when it is uploaded. You can also configure the ACL after the object is uploaded successfully.

ACLs of buckets and objects can be set in either of the following ways:

- Call a separate ACL API to configure existing resources (buckets or objects).
- When creating a bucket or uploading an object, you can set the ACL of the bucket or object based on the header.
5.3.2 Configuring ACLs for Existing Resources

The OBS allows you to set an ACL for an existing bucket or object. After the ACL is configured successfully, the previous ACL will be overwritten and the new ACL takes effect. The ACL format of an existing resource is XML. The format is as follows:

- Setting the ACL Format for an Existing Bucket

When configuring an ACL for an existing bucket, ensure that the request body contains an ACL in XML format. The ACL format is as follows:

```xml
<AccessControlPolicy>
  <Owner>
    <ID>domainId</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>domainId</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

**Delivered** in **Grant** indicates whether the ACL permission of the bucket is passed to the object in the bucket. If the value is **true**, the ACL permissions of the object are the same as the ACL permissions of the bucket unless the object displays that the ACL permissions of the bucket is not inherited.

- Setting the ACL Format for an Existing Object

When you set an ACL for an existing object, the request message body must contain the ACL list in XML format. The format of the ACL is as follows:

```xml
<AccessControlPolicy>
  <Owner>
    <ID>domainId</ID>
  </Owner>
  <Delivered>true</Delivered>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>domainId</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

**Delivered** in the ACL list indicates whether the object ACL inherits the ACL permission of the bucket. The default value is **true**. The object inherits the bucket ACL permissions by default. If you do not want to inherit the bucket ACL permissions, set this parameter to **false**.

The ACL check for buckets is simple, but complex for object. Therefore, **Delivered** needs to be considered. Check the object ACL. If the ACL permission of the object does not meet the requirement and is **<Delivered>true</Delivered>**, continue to check the bucket ACL. If the ACL permission of the bucket is **<Delivered>true</Delivered>**, the ACL can be applied to the object. If the object ACL is configured to **<Delivered>false</Delivered>**, the ACL permission of the bucket cannot be inherited. If the ACL permission of the bucket is **<Delivered>false</Delivered>**, the bucket permission cannot be applied to the object. (The permission inheritance must be the **<Delivered>true</Delivered>** of the object, and the permission of the bucket is **<Delivered>true</Delivered>**.)
OBS supports five types of permissions (for details, see Table 5-1). \(<\text{Grantee}>\) indicates the target of permission granting, including OBS users and all users. The format of permission is slightly different.

- Grantee is an OBS user:
  \(<\text{Grantee}>
  \<\text{ID}>\text{domainId}\</\text{ID}>
  \</\text{Grantee}>\)

- Grantee is everyone:
  \(<\text{Grantee}>
  \<\text{Canned}>\text{Everyone}\</\text{Canned}>
  \</\text{Grantee}>\)

### Table 5-1 Permissions for an OBS bucket or object

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
</table>
| READ           | If you have the read permission for a bucket, you can obtain the list of objects, multipart tasks, and multiple object versions in the bucket, as well as metadata of the bucket.  
A grantee with such a permission for an object can obtain the object content and metadata. |
| WRITE          | A grantee with such a permission on a bucket can upload, overwrite, and delete any object or part in the bucket.  
This permission is not applicable to objects. |
| READ_ACP       | A grantee with this permission can obtain the ACL of a bucket or object.  
A bucket or object owner has this permission permanently. |
| WRITE_ACP      | A grantee with this permission can update the ACL of a bucket or object.  
A bucket or object owner has this permission permanently.  
A grantee with such a permission can modify the access control policy to obtain desired access permission. |
| FULL_CONTROL   | A grantee with such a permission for a bucket has READ, WRITE, READ_ACP, and WRITE_ACP permissions.  
A grantee with such a permission for an object has READ, READ_ACP, and WRITE_ACP permissions. This restriction does not apply to users with the read-only permission. |

**NOTE**

A request supports a maximum of 100 permissions.
 Instead of being added with new permissions, the ACL of a bucket or object is overwritten after a grant of permissions for the bucket or object is performed.

### 5.3.3 Configuring an ACL Using Header Fields

**Access Control Policy**
You can set an access control policy when creating a bucket or uploading an object using the header field `x-obs-acl`. Available access control policies are predefined.

The `x-obs-acl` is special, which can be configured with six types of permissions. No matter what type of permissions is configured, the owner has full control permission for the resource. For details, see the following table.

**Table 5-2 Description of pre-defined access control policies in OBS**

<table>
<thead>
<tr>
<th>Pre-Defined Access Control Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>Indicates that the owner of a bucket or object has the full control permission for the bucket or object. Other users have no permission to access the bucket or object.</td>
</tr>
<tr>
<td>public-read</td>
<td>If this permission is set for a bucket, everyone can obtain the list of objects, multipart tasks, and multiple object versions in the bucket, as well as metadata of the bucket. If this permission is set for an object, everyone can obtain the content and metadata of the object.</td>
</tr>
<tr>
<td>public-read-write</td>
<td>If this permission is set for a bucket, everyone can obtain the object list in the bucket, multipart tasks in the bucket, the bucket metadata; and the bucket versions, and can upload objects; delete objects; initialize multipart upload tasks; upload parts; merge parts; copy parts; and cancel multipart upload tasks. If this permission is set for an object, everyone can obtain the content and metadata of the object.</td>
</tr>
<tr>
<td>public-read-delivered</td>
<td>If this permission is set for a bucket, everyone can obtain the object list, multipart tasks, bucket metadata, and bucket versions, and obtain the content and metadata of the objects in the bucket. It cannot be applied to objects.</td>
</tr>
<tr>
<td>public-read-write-delivered</td>
<td>If this permission is set for a bucket, everyone can obtain the object list in the bucket, multipart tasks in the bucket, the bucket metadata; and the bucket versions, and can upload objects; delete objects; initialize multipart upload tasks; upload parts; merge parts; copy parts; and cancel multipart upload tasks. Users can also obtain content and metadata of objects in the bucket. It cannot be applied to objects.</td>
</tr>
</tbody>
</table>
Pre-Defined Access Control Policy | Description
--- | ---
bucket-owner-full-control | If this permission is set for a bucket, only the bucket owner has the full control over the bucket, and the bucket cannot be accessed by other users. If this permission is set for an object, only the bucket owner and the object owner have the full control over the object.

**NOTE**

By default, the access control policy is private.

When creating a bucket or uploading an object, you can set other header fields as follows:

**Table 5-3 Header fields for setting bucket or object ACLs**

<table>
<thead>
<tr>
<th>Header Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-obs-grant-read</td>
<td>Grant the READ permission to all users in a specified account.</td>
</tr>
<tr>
<td>x-obs-grant-write</td>
<td>Grant the WRITE permission to all users in a specified account.</td>
</tr>
<tr>
<td>x-obs-grant-read-acp</td>
<td>Grant the READ_ACP permission to all users in a specified account.</td>
</tr>
<tr>
<td>x-obs-grant-write-acp</td>
<td>Grant the WRITE_ACP permission to all users in a specified account.</td>
</tr>
<tr>
<td>x-obs-grant-full-control</td>
<td>Grant the FULL_CONTROL permission to all users in a specified account.</td>
</tr>
<tr>
<td>x-obs-grant-read-delivered</td>
<td>Grant the READ permission for buckets and objects in the bucket to all users in a specified account, and objects inherit the permission of the bucket. It cannot be applied to objects.</td>
</tr>
<tr>
<td>x-obs-grant-full-control-delivered</td>
<td>Grant the FULL_CONTROL permission to all users in a specified account, and objects inherit the bucket permission. It cannot be applied to objects.</td>
</tr>
</tbody>
</table>

**5.3.4 Conversion Between Two ACL Formats**

The following uses an example to describe the relationship between ACL formats. The example for a bucket is as follows:
- **x-obs-acl: private**
  <AccessControlPolicy>
    <Owner>
      <ID>domainId</ID>
    </Owner>
    <AccessControlList>
      <Grant>
        <Grantee>
          <ID>domainId</ID>
        </Grantee>
        <Permission>FULL_CONTROL</Permission>
        <Delivered>false</Delivered>
      </Grant>
    </AccessControlList>
  </AccessControlPolicy>

- **x-obs-acl: public-read-write**
  <AccessControlPolicy>
    <Owner>
      <ID>domainId</ID>
    </Owner>
    <AccessControlList>
      <Grant>
        <Grantee>
          <ID>domainId</ID>
        </Grantee>
        <Permission>FULL_CONTROL</Permission>
        <Delivered>false</Delivered>
      </Grant>
      <Grant>
        <Grantee>
          <Canned>Everyone</Canned>
        </Grantee>
        <Permission>READ</Permission>
        <Delivered>false</Delivered>
      </Grant>
      <Grant>
        <Grantee>
          <Canned>Everyone</Canned>
        </Grantee>
        <Permission>WRITE</Permission>
        <Delivered>false</Delivered>
      </Grant>
    </AccessControlList>
  </AccessControlPolicy>

- **x-obs-acl: public-read-write-delivered**
  <AccessControlPolicy>
    <Owner>
      <ID>domainId</ID>
    </Owner>
    <AccessControlList>
      <Grant>
        <Grantee>
          <ID>domainId</ID>
        </Grantee>
        <Permission>FULL_CONTROL</Permission>
        <Delivered>false</Delivered>
      </Grant>
      <Grant>
        <Grantee>
          <Canned>Everyone</Canned>
        </Grantee>
        <Permission>READ</Permission>
        <Delivered>true</Delivered>
      </Grant>
      <Grant>
        <Grantee>
          <Canned>Everyone</Canned>
        </Grantee>
        <Permission>WRITE</Permission>
        <Delivered>true</Delivered>
      </Grant>
    </AccessControlList>
  </AccessControlPolicy>
The example for an object is as follows:

- **x-obs-acl : private**
  ```xml
  <AccessControlPolicy>
  <Owner>
    <ID>domainId</ID>
  </Owner>
  <Delivered>true</Delivered>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>domainId</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
      <Delivered>false</Delivered>
    </Grant>
    <Grant>
      <Grantee>
        <ID>domainId1</ID>
      </Grantee>
      <Permission>READ</Permission>
      <Delivered>false</Delivered>
    </Grant>
    <Grant>
      <Grantee>
        <ID>domainId1</ID>
      </Grantee>
      <Permission>WRITE</Permission>
      <Delivered>false</Delivered>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
  ```
5.3.5 ACL Application Examples

Example 1: Create a bucket for storing static web pages.

When you make the API call to create a bucket, the ACL header field is set to `x-obs-acl: public-read-delivered`. After the bucket is created, you do not need to set an ACL or an independent ACL header when uploading objects.

Example 2: Some files in the bucket need to be shared to users 1 and user 2. A small number of files are shared to user 3. Some files cannot be shared.
Create a bucket and set the ACL of the bucket to `x-obs-grant-read-delivered:id=domainID1, id=domainID2`; when a user uploads an object, the object shared to user 1 and user 2 does not carry the ACL by default. When the object to be shared to user 3 is uploaded, the `x-obs-grant-read:id=domainID3` is carried; After the objects that are not shared are uploaded, set the acl to `<Delivered>false</Delivered>.

### 5.4 Bucket Policy

#### 5.4.1 About Bucket Policy

Bucket policies provide centralized access control to buckets and objects based on a variety of conditions, including OBS operations, authorizers, resources, and elements (for example IP addresses) of the request. The permissions attached to a bucket apply to all of the objects in that bucket. Tenants have the power to grant bucket policy permissions and assign employees permissions based on a variety of conditions. For example, an account can create a policy that gives a user the write permission:

- To a particular bucket for a user.
- For a user from a specified account's corporate network.
- For an account's custom application.

Different from ACLs which can add (grant) permissions only on individual objects, bucket policies can either add or deny permissions on all objects within a bucket. With one request, an account can set permissions for any number of objects in a bucket. In addition, an account can add the wildcard (similar to regular expression operators) to resource names and other values to control access to a set of objects.

A bucket owner can perform the PutBucket policy operation to set a policy on bucket access. A new policy will overwrite the existing one. A bucket owner can also perform the GetBucket Policy or DeleteBucket Policy operation to obtain or delete the existing bucket policy. After a policy is set for a bucket, all subsequent accesses to the bucket are controlled by the policy. The description of a policy determines whether a request will be accepted or rejected. The following policy allows `783fc6652cf246c096ea836694f71855` (account ID) and `219d520ceac84c5a98b237431a2cf4c2` (account ID) to perform the GetObject operation on all the objects in bucket mybucket.

```json
{
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "1",
            "Principal": {
                "ID": ["domain/783fc6652cf246c096ea836694f71855:user/*",
                        "domain/219d520ceac84c5a98b237431a2cf4c2:user/*"],
                "ID": ["domain/783fc6652cf246c096ea836694f71855:user/*",
                        "domain/219d520ceac84c5a98b237431a2cf4c2:user/*"],
                "ID": ["domain/783fc6652cf246c096ea836694f71855:user/*",
                        "domain/219d520ceac84c5a98b237431a2cf4c2:user/*"]
            },
            "Action": ["GetObject"],
            "Resource": ["mybucket/*"]
        }
    ]
}
```
5.4.2 Policy Format

A policy is described in the JSON format, as described in the following syntax:

```
{
  "Statement": [
    {
      "Sid": "ExampleStatementID1",
      "Principal": "*",
      "Effect": "Allow",
      "Action": "ListBucket",
      "Resource": "examplebucket",
      "Condition": "some conditions"
    },
    {
      "Sid": "ExampleStatementID2",
      "Principal": "*",
      "Effect": "Allow",
      "Action": "PutObject",
      "Resource": "examplebucket",
      "Condition": "some conditions"
    },
    ....
  ]
}
```

Example:

```
{
  "Statement": [
    {
      "Sid": "ExampleStatementID1",
      "Principal": "*",
      "Effect": "Allow",
      "Action": "ListBucket",
      "Resource": "examplebucket",
      "Condition": "some conditions"
    },
    {
      "Sid": "ExampleStatementID2",
      "Principal": "*",
      "Effect": "Allow",
      "Action": "PutObject",
      "Resource": "examplebucket",
      "Condition": "some conditions"
    },
    ....
  ]
}
```

A policy can consist of multiple statements or one statement. The structure of each statement contains the following contents:

**Table 5-4 Statement elements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sid</td>
<td>ID of a statement. The value is a string that describes the statement.</td>
<td>No</td>
</tr>
<tr>
<td>Principal</td>
<td>Optional keyword. Grantee of a statement. The value can be a wildcard character (<em>) that indicates all domains and users. When authorizing permissions to all users under a domain, the format of Principal is domain/domainiduser/</em>. When authorizing permissions to a specific user under a domain, the format of Principal is domain/domainiduser/userid or domain/domainiduser/userName.</td>
<td>No. Select either Principal or NotPrincipal.</td>
</tr>
<tr>
<td>NotPrincipal</td>
<td>An exception to a list of principals in the statement. You can deny access to all principals except the ones named in the NotPrincipal element. The value format is the same as Principal.</td>
<td>No. Select either NotPrincipal or Principal.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>Action</td>
<td>Specifies the operation to which the statement applies. This parameter is optional. The Action field is a set of all operations supported by OBS. It is a string of case-insensitive characters, and supports a wildcard character (<em>) that indicates all operations. For example: &quot;Action&quot;: [&quot;List</em>&quot;, &quot;Get*&quot;].</td>
<td>No. Select either Action or NotAction.</td>
</tr>
<tr>
<td>NotAction</td>
<td>An exception to a list of actions in the statement. All actions are performed except the one specified in NotAction. The value of this element is similar to Action.</td>
<td>No. Select either Action or NotAction.</td>
</tr>
<tr>
<td>Effect</td>
<td>Indicates whether permission in a statement is Allow or Deny.</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource</td>
<td>Specifies a group of resources on which the statement takes effect. The wildcard * is supported, indicating all resources.</td>
<td>No. Select either Resource or NotResource.</td>
</tr>
<tr>
<td>NotResource</td>
<td>An exception to a list of resources in the statement. A policy is not applied to resources specified in NotResource. The value of this parameter is similar to that of Resource.</td>
<td>No. Select either Resource or NotResource.</td>
</tr>
<tr>
<td>Condition</td>
<td>Indicates the conditions for a statement to take effect.</td>
<td>No</td>
</tr>
</tbody>
</table>

**NOTE**

A statement contains either Action or NotAction, either Resource or NotResource, and either Principal or NotPrincipal.

**Principal / NotPrincipal**

The Principal or NotPrincipal supported by OBS includes anonymous users, specific tenants, specific users, federated users, and entrusted users.

- **All (anonymous users)**
  
  "Principal": {"ID": "*"}

  In the example, the wildcard (*) is used as a placeholder for Everyone/Anonymous. We also strongly recommend that you do not use wildcards in the Principal element in the role’s trust policy unless you restrict access by using the Condition element in the policy.

- **Specific account**

  When the account identifier is used as the authorizer in the policy, the permission in the policy statement can be granted to all roles contained in the account. This includes all subscribers under the account. The following example demonstrates how to specify an account as an authorized person.

  "Principal": { "ID": "domain/domainIdxxxx:user/*" }
You can grant permissions to multiple tenants, as described in the following example:

```
"Principal": {
  "ID": [
    "domain/domainIDxx1:user/useridxxxx",
    "domain/domainIDxx2:user/**"
  ]
}
```

- **Specific user**

  In the `Principal` element, the user name is case sensitive.

  ```
  "Principal": {"ID": "domain/domainIDxxx:user/user-name" }
  "Principal": {
    "ID": ["domain/domainIDxxx:user/UserID1",
    "domain/domainIDxxx:user/UserID2"
  ]
  }
  ```

- **Federated user (using SAML identity provider)**

  ```
  "Principal": {"Federated": "domain/domainIDxxx:identity-provider/provider-name" }
  "Principal": {"Federated": "domain/domainIDxxx:group/groupname" }
  ```

- **Agencies**

  ```
  "Principal": {"ID": "domain/domainIDxxc:agency/agencyname" }
  ```

**Action / NotAction**

OBS supports the following bucket **Action** or **NotAction**:

- CreateBucket
- DeleteBucket
- HeadBucket
- ListBucket
- ListBucketVersions
- ListBucketMultipartUploads
- GetBucketAcl
- PutBucketAcl
- GetBucketCORS
- PutBucketCORS
- GetBucketVersioning
- PutBucketVersioning
- GetBucketLocation
- GetBucketPolicy
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketLogging
- PutBucketLogging
- GetBucketWebsite
- PutBucketWebsite
- DeleteBucketWebsite
The object **Action** or **NotAction** supported by OBS is as follows:

- GetObject (applicable to GET Object and HEAD Object)
- GetObjectVersion
- PutObject (applicable to PUT Object, POST Object, Initiate Multipart Upload, Upload Part, and Complete Multipart Upload)
- GetObjectAcl
- GetObjectVersionAcl
- PutObjectAcl
- PutObjectVersionAcl
- DeleteObject
- DeleteObjectVersion
- ListMultipartUploadParts
- AbortMultipartUpload
- RestoreObject
- ReplicateObject
- ReplicateDelete
- ModifyObjectMetaData

**Resource / NotResource**

The resources supported by the OBS are as follows:

- **bucketname** (bucket operation): The **Action** drop-down list box contains the list of supported bucket actions. If you want to perform the operations on the bucket, set **Resource** to the bucket name.

- **bucketname/objectname** (object operation): The **Action** drop-down list box contains the **Supported Object Action** list. If you want to respond to an object in a bucket, set **Resource** to **bucketname/objectname**. **objectname** supports wildcards. For example, if you have permission on the directory object in a bucket, set **Resource** to "bucketname/director/**". If you have permission on all objects in a bucket, set **Resource** to "bucketname/**". If permissions for both buckets and objects in the bucket need to be granted, set **Resource** to ["examplebucket/**", "examplebucket"].

The following example policy grants all operation permissions (including bucket operations and object operations) of **examplebucket** to user1 whose user ID is 71f3901173514e6988115ea2c26d1999 in account b4bf1b36d9ca43d984fbc99491b6f9e9 (account ID).

```json
{
  "Statement": [
    {
      "Sid": "test",
      "Action": [*],
      "Effect": "Allow",
      "Resource": ["examplebucket/**", "examplebucket"]
    }
  ]
}
```
"Effect":"Allow",
"Principal": {"ID": ["domain/b4bf1b36d9ca43d984fbc9491b6fce9:user/
71f3901173514e6988115ea2c26d1999"],
"Action": ["*"]
"Resource": ["examplebucket/**","examplebucket"]
}
]
}

**Condition**

The following table lists the general types of **Condition** that you can specify.

**Table 5-5 Condition**

<table>
<thead>
<tr>
<th>Type</th>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>StringEquals</td>
<td>Strict matching. Short version: streq</td>
</tr>
<tr>
<td></td>
<td>StringNotEquals</td>
<td>Strict negated matching. Short version: strneq</td>
</tr>
<tr>
<td></td>
<td>StringEqualsIgnoreCase</td>
<td>Strict matching, ignoring case. Short version: streqi</td>
</tr>
<tr>
<td></td>
<td>StringNotEqualsIgnoreCase</td>
<td>Strict negated matching, ignoring case. Short version: strneqi</td>
</tr>
<tr>
<td></td>
<td>StringLike</td>
<td>Loose case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: strl</td>
</tr>
<tr>
<td></td>
<td>StringNotLike</td>
<td>Negated loose case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: strnl</td>
</tr>
<tr>
<td>Numeric</td>
<td>NumericEquals</td>
<td>Strict matching. Short version: numeq</td>
</tr>
<tr>
<td></td>
<td>NumericNotEquals</td>
<td>Strict negated matching. Short version: numneq</td>
</tr>
<tr>
<td></td>
<td>NumericLessThan</td>
<td>&quot;Less than&quot; matching. Short version: numlt</td>
</tr>
<tr>
<td></td>
<td>NumericLessThanEquals</td>
<td>&quot;Less than or equals&quot; matching. Short version: numlteq</td>
</tr>
<tr>
<td></td>
<td>NumericGreaterThan</td>
<td>&quot;Greater than&quot; matching. Short version: numgt</td>
</tr>
<tr>
<td></td>
<td>NumericGreaterThanEquals</td>
<td>&quot;Greater than or equals&quot; matching. Short version: numgteq</td>
</tr>
<tr>
<td>Type</td>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Date</td>
<td>DateEquals</td>
<td>Strict matching. Short version: dateeq</td>
</tr>
<tr>
<td></td>
<td>DateNotEquals</td>
<td>Strict negated matching. Short version: dateneq</td>
</tr>
<tr>
<td></td>
<td>DateLessThan</td>
<td>Indicates that the date is earlier than a specific date. Short version: datelt</td>
</tr>
<tr>
<td></td>
<td>DateLessThanEquals</td>
<td>A point in time at which a key stops taking effect. Short version: dateltq</td>
</tr>
<tr>
<td></td>
<td>DateGreaterThan</td>
<td>A point in time at which a key starts taking effect. Short version: dategt</td>
</tr>
<tr>
<td></td>
<td>DateGreaterThanEquals</td>
<td>A point in time at which a key starts taking effect. Short version: dategteq</td>
</tr>
<tr>
<td>Boolean</td>
<td>Bool</td>
<td>Strict Boolean matching</td>
</tr>
<tr>
<td>IP address</td>
<td>IpAddress</td>
<td>Approved based IP address or range</td>
</tr>
<tr>
<td></td>
<td>NotIpAddress</td>
<td>All IP addresses exclude the specified IP address or IP address range</td>
</tr>
</tbody>
</table>

**NOTE**

Elements in Condition are case sensitive. The date format complies with the ISO 8601 standard. For example: `2015-07-01T12:00:00Z`

Each **Condition** block can contain multiple key-value combinations. The **Condition** combination in the following figure indicates that the request time ranges from `2015-07-01T12:00:00Z` to `2018-04-16T15:00:00Z` and the request IP address range is "192.168.176.0/24" or "192.168.143.0/24".

```
"Condition" : {
  "DateGreaterThan" : {
    "CurrentTime" : "2015-07-01T12:00:00Z"
  },
  "DateLessThan" : {
    "CurrentTime" : "2018-04-16T15:00:00Z"
  },
  "IpAddress" : {  
    "SourceIp" : ["192.168.176.0/24","192.168.143.0/24"]
  }
}
```

A Condition block can contain two types of keys: general keys that have nothing to do with **Action** and **Action** related keys.

Keys irrelevant to **Action** are listed as follows:
Table 5-6 Common Condition Key

<table>
<thead>
<tr>
<th>Condition Key</th>
<th>Condition Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentTime</td>
<td>Date</td>
</tr>
<tr>
<td>EpochTime</td>
<td>Numeric</td>
</tr>
<tr>
<td>SecureTransport</td>
<td>Bool</td>
</tr>
<tr>
<td>SourceIp</td>
<td>IP address</td>
</tr>
<tr>
<td>UserAgent</td>
<td>String</td>
</tr>
<tr>
<td>Referer</td>
<td>String</td>
</tr>
</tbody>
</table>

Keys in **Condition** must be used in certain actions. The following table lists the key pairs in **Action** and **Condition**.

Table 5-7 OBS Action Condition Key

<table>
<thead>
<tr>
<th>Action</th>
<th>Optional Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateBucket</td>
<td>x-obs-acl</td>
<td>When a bucket is created, the value range of <strong>Canned ACL</strong> in the <strong>x-obs-acl</strong> header field is <em>private</em></td>
</tr>
<tr>
<td>ListBucket</td>
<td>prefix</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>delimiter</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>max-keys</td>
<td>Numeric</td>
</tr>
<tr>
<td>ListBucketVersions</td>
<td>prefix</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>delimiter</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>max-keys</td>
<td>Numeric</td>
</tr>
<tr>
<td>PutBucketAcl</td>
<td>x-obs-acl</td>
<td>When the bucket ACL is modified, the value range of <strong>Canned ACL</strong> in the <strong>x-obs-acl</strong> header field is <em>private</em></td>
</tr>
<tr>
<td>PutObject</td>
<td>x-obs-acl</td>
<td>When an object is uploaded, the value range of <strong>Canned ACL</strong> in the <strong>x-obs-acl</strong> header field is <em>private</em></td>
</tr>
<tr>
<td></td>
<td>x-obs-copy-source</td>
<td>Type: String. Format: <code>/bucketname/keyname</code></td>
</tr>
<tr>
<td>Action</td>
<td>Optional Key</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>x-obs-metadata-directive</td>
<td>Valid values: COPY</td>
<td>REPLACE</td>
</tr>
<tr>
<td>PutObjectAcl</td>
<td>x-obs-acl</td>
<td>When an object is modified, the value range of Canned ACL in the x-obs-acl header field is private</td>
</tr>
<tr>
<td>GetObjectVersion</td>
<td>VersionId</td>
<td>String</td>
</tr>
<tr>
<td>GetObjectVersionAcl</td>
<td>VersionId</td>
<td>String</td>
</tr>
<tr>
<td>PutObjectVersionAcl</td>
<td>VersionId</td>
<td>String</td>
</tr>
<tr>
<td>x-obs-acl</td>
<td>When an object version is modified, the value range of Canned ACL in the x-obs-acl header field is private</td>
<td>public-read</td>
</tr>
<tr>
<td>DeleteObjectVersion</td>
<td>VersionId</td>
<td>String</td>
</tr>
</tbody>
</table>

**Policy Permission Determination Logic**

A policy may pose any of the three results for each statement: explicit deny, allow, and default deny. If a bucket policy contains multiple statements, the policy determines which statement prevails according to the following rules:

1. If conditions in any statement of a policy are not met, the policy poses a default deny result.
2. An explicit deny overrides allows.
3. An allow overrides default denies.
4. Statements can be in any order in a policy.

**Table 5-8 Statement Result**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit deny</td>
<td>A statement defines effect=&quot;deny&quot;. All requests for resources to which the statement applies are denied. No permission is returned.</td>
</tr>
<tr>
<td>allow</td>
<td>A statement defines effect=&quot;allow&quot;. All requests for resources to which the statement applies are allowed.</td>
</tr>
<tr>
<td>default deny</td>
<td>Conditions defined in a statement are not met. Requests are denied.</td>
</tr>
</tbody>
</table>
If an ACL and a bucket policy are applied together to an account, an explicit deny in the bucket policy overrides allows in the ACL.

If a bucket policy and an IAM policy are applied together to an account, an explicit deny overrides allows, and an allow overrides default denies.

SSE-KMS server-side encrypted object does not support Bucket ACL/Policy for cross-account authorization.

5.4.3 Bucket Policy Examples

Examples about typical bucket policy use cases are provided as follows: The policy uses the examplebucket as the bucket name in the examples. To test these policies, you need to replace these strings with your actual bucket name and change the authorized user to the user you want to authorize.

Granting Permissions to OBS Users

The following example policy grants the PutObject and PutObjectAcl permissions to the user whose ID is 71f3901173514e6988115ea2c26d1999 under account b4bf1b36d9ca43d984fbc9491b6fce9 (account ID).

```json
{
    "Statement": [
        {
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "ID": "domain/b4bf1b36d9ca43d984fbc9491b6fce9:user/71f3901173514e6988115ea2c26d1999"},
            "Action": ["PutObject", "PutObjectAcl"],
            "Resource": ["examplebucket/*"]
        }
    ]
}
```

Grant All Operation Permissions for a Specified Bucket to an OBS User.

The following example policy grants all operation permissions (including bucket operations and object operations) of examplebucket to the user whose ID is 71f3901173514e6988115ea2c26d1999 in account b4bf1b36d9ca43d984fbc9491b6fce9 (account ID).

```json
{
    "Statement": [
        {
            "Sid": "test",
            "Effect": "Allow",
            "Principal": {
                "ID": "domain/b4bf1b36d9ca43d984fbc9491b6fce9:user/71f3901173514e6988115ea2c26d1999"},
            "Action": ["*"],
            "Resource": ["examplebucket/*", "examplebucket"]
        }
    ]
}
```
Granting All Permissions Excluding the Permission for Deleting Objects to an OBS User

The following example policy grants a user (user ID 71f3901173514e6988115ea2c26d1999) of an account (ID b4bf1b36d9ca43d984fbc9491b6fce9) all permissions for the examplebucket bucket, excluding the permission to delete objects.

```json
{
    "Statement": [
        {
            "Sid": "test1",
            "Effect": "Allow",
            "Principal": {
                "ID": 
                    "domain/b4bf1b36d9ca43d984fbc9491b6fce9:user/71f3901173514e6988115ea2c26d1999"},
            "Action": ["*"],
            "Resource": ["examplebucket/*"]
        },
        {
            "Sid": "test2",
            "Effect": "Deny",
            "Principal": {
                "ID": 
                    "domain/b4bf1b36d9ca43d984fbc9491b6fce9:user/71f3901173514e6988115ea2c26d1999"},
            "Action": ["DeleteObject"],
            "Resource": ["examplebucket/*"]
        }
    ]
}
```

Granting Permissions to Multiple Tenants and Specifying Conditions

The following example policy grants the PutObject and PutObjectAcl permissions to user 783fc6652cf246c096ea836694f71855 (account ID) and 219d520ceac84c5a98b237431a2cf4c2 (account ID), and requires that any request for these operations include the standard ACL permission of public-read.

```json
{
    "Statement": [
        {
            "Sid": "AddAcl",
            "Effect": "Allow",
            "Principal": {
                "ID": 
                    "domain/783fc6652cf246c096ea836694f71855:user/*","domain/219d520ceac84c5a98b237431a2cf4c2:user/*"},
            "Action": ["PutObject","PutObjectAcl"],
            "Resource": ["examplebucket/*"],
            "Condition": {"StringEquals":{"x-obs-acl": ["public-read"]}}
        }
    ]
}
```

Granting the Read-Only Permission to Anonymous Users

The following example policy grants GetObject (download object) permission to anonymous users (non-public cloud users). This permission allows anyone to read all object data uploaded by the owner of the bucket. This is useful when you configure the bucket as a website and expect everyone to read objects in the bucket.

```json
{
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": {
                "ID": "*"},
            "Action": ["GetObject"],
            "Resource": ["examplebucket/*"]
        }
    ]
}
```
Granting the Read-Only Permission for Specific Objects to Anonymous Users

The following example policy grants the **GetObject** (download object) permission of **exampleobject** in bucket **examplebucket** to anonymous users, allowing everyone to read data of the exampleobject object.

```json
{
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["GetObject"],
            "Resource": ["examplebucket/exampleobject"]
        }
    ]
}
```

Restricting Access to Specific IP Addresses

The following policy grants all users the permission to perform any OBS operation. However, the requests must be from the specified IP address range. The IP address range that is allowed by the statement is 192.168.0.* with an exception of 192.168.0.1.

**Condition** uses **IpAddress**, **NotIpAddress**, and **SourceIp** (in OBS range). The value of **SourceIp** is the CIDR notation described in RFC 4632.

```json
{
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["Get*"]
            "Resource": ["examplebucket/*"],
            "Condition": {
                "IpAddress": {"SourceIp": "192.168.0.0/24"},
                "NotIpAddress": {"SourceIp": "192.168.0.1/32"}
            }
        }
    ]
}
```

Restrict Accesses to Specified HTTP Website

Assume that you have a website whose domain name is **www.example.com** or **example.com** with a link **examplebucket** pointing to photos and videos in the OBS bucket. By setting the permission, all objects in a bucket can be accessed by anonymous users. However, you can only read and access these objects from your website. You can add a bucket policy that allows the **GetObject** condition, that is, the request must come from a specific web page. This function is called anti-leeching. For details, see **5.4.4 URL Validation Settings**.
Grant the PutObject permission to the copied object from the specified replication source.

For PutObject requests, when a user specifies a source object, this is a copy operation. Correspondingly, the bucket owner grants the permission to copy the object to the user, but has restrictions on the source object, for example:

- Only objects in bucket `sourcebucket` can be copied.
- Objects starting with `public/` in bucket `sourcebucket` can be copied. For example: `sourcebucket/public/*`
- Only a specific object in bucket `sourcebucket` can be copied. For example: `sourcebucket/example.jpg`.

The following example bucket policy grants PutObject (upload object and copy object) permission to user `71f3901173514e6988115ea2c26d1999` (user ID) in account `b4bf1b36d9ca43d984fbc9491b6f6ce9` (account ID), but can copy only objects whose name is prefixed with `public/` in the `sourcebucket` bucket.

```json
{
    "Statement": [
        {
            "Sid": "putObject",
            "Effect": "Allow",
            "Principal": {
                "ID": "domain/b4bf1b36d9ca43d984fbc9491b6f6ce9:user/71f3901173514e6988115ea2c26d1999"
            },
            "Action": ["PutObject"],
            "Resource": "examplebucket/*"
        },
        {
            "Sid": "Deny copy /bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "ID": "domain/b4bf1b36d9ca43d984fbc9491b6f6ce9:user/71f3901173514e6988115ea2c26d1999"
            },
            "Action": ["PutObject"],
            "Resource": "examplebucket/*",
            "Condition": {
                "StringNotLike": {
                    "x-obs-copy-source": "sourcebucket/public/*"
                }
            }
        }
    ]
}
```

Granting the Permission to Access the Specified Version of the Object

Assume that the multi-version function is enabled for the bucket of account A. The bucket contains multiple versions of object `obj01`. The account administrator wants to grant the user `71f3901173514e6988115ea2c26d1999` (user ID) only to obtain the permission of the specified version of the object. The account administrator only needs to grant the permission to download specified version objects as required, described as follows: The key of the `VersionId` condition is specified in the key value pair of Condition.

```json
{
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "ID": "domain/b4bf1b36d9ca43d984fbc9491b6f6ce9:user/71f3901173514e6988115ea2c26d1999"
            },
            "Action": ["GetObject"],
            "Resource": ["examplebucket/obj01@v1"],
            "Condition": {
                "VersionId": "v1"
            }
        }
    ]
}
```
Allowing Users to Obtain the Object List in a Bucket Based on a Specified Prefix

The bucket owner can limit the content of a specified folder in a bucket. It is useful if the object in the bucket is organized based on the prefix of the key value. The OBS Console displays the folder level based on the prefix.

This condition limits the user to listing object keys with the examplefolder prefix. The added explicit rejection will deny the user to list the keys with any other prefix, regardless of what permission the user might have. For example, the user may obtain the permission to list the object keys that are not limited by updating a previous user policy or by using a bucket policy. However, because explicit rejection always replaces any other permission, user requests listing non-examplefolder prefixes will be rejected.
5.4.4 URL Validation Settings

OBS is charged based on usage. To prevent user data from being stolen, OBS supports URL validation based on HTTP headers. OBS also supports both whitelist and blacklist settings.

- **Whitelist settings**

  Users can set a whitelist to allow requests from the websites added in the whitelist and deny requests from any other website.

  For the requests that are initialized from browsers’ address boxes, that is, those HTTP requests with a blank `referer`, users can add the `$null` field to "Referer" of Condition to specify whether to allow the requests with a blank referer.

  Set a whitelist based on the following policy setting:

  ```json
  "Statement":[
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {"ID": "*"},
      "Action": "*",
      "Resource": ["bucket/*"],
    },
    {
      "Sid": "2",
      "Effect": "Deny",
      "Principal": {"ID": "*"},
      "Action": "*",
      "Resource": ["bucket/*"],
      "Condition": {
        "StringNotEquals": {
          "Referer": ["http://www.example01.com","$null"]
        }
      }
    }
  ]
  ```

  If you set a whitelist in this way, you can perform operations on resources in buckets only when the value of the referer parameter is `www.example01.com` or is blank.

- **Blacklist settings**

  You can refer to the following policy settings to set a blacklist for access.

  ```json
  "Statement":[
    {"Sid": "1",
      "Effect": "Deny",
      "Principal": {"ID": "*"},
      "Action": "*",
      "Resource": ["bucket/*"],
      "Condition": {
        "StringEquals": {
          "Referer": ["http://www.example01.com","http://www.example02.com"]
        }
      }
    }
  ]
  ```

  Issue 04 (2019-07-16)
If you set a blacklist in this way, you cannot perform operations on resources in buckets when the value of the referer parameter is www.example01.com or www.example02.com.

5.5 IAM User Policy (IAM Policy)

5.5.1 Introduction to IAM Policy

Identity and Access Management (IAM) provides permissions management to secure access to your HUAWEI CLOUD services and resources. IAM Policy is a user-specific authorization policy. With IAM Policy, you can manage user accounts (such as employee, system, and application accounts), and assign resource operation permissions to these accounts. For example, you can restrict your users to have only the read permission on a specific OBS bucket.

There are fine-grained policies and Role-Based Access Control (RBAC policies).

- An RBAC policy consists of permissions for an entire service. Users in a group with such a policy attached are granted all of the permissions required for accessing and managing resources of that service. RBAC policies do not support operation-specific permissions control.
- A fine-grained policy consists of API-based permissions for operations on specific resource types. Fine-grained policies, as the name suggests, allow for more refined permissions control. For example: You can restrict a sub-user to access only the objects in a specific directory of an OBS bucket.

System-Defined Policies

IAM supports both system-defined and custom policies. System-defined policies contain the common permissions required to access various services. They are classified into RBAC policies and fine-grained policies. System-defined policies can be directly used to assign permissions to user groups but cannot be modified.

Table 5-9 RBAC policies related to OBS in system-defined policies

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant Administrator</td>
<td>Users with this permission can perform any operation on OBS resources.</td>
</tr>
<tr>
<td>Tenant Guest</td>
<td>Users with this permission can query the usage of OBS resources. Specifically, a user with this permission can only read OBS resources.</td>
</tr>
<tr>
<td>OBS Buckets Viewer</td>
<td>Users with this permission can list buckets, obtain basic bucket information, and obtain bucket metadata.</td>
</tr>
</tbody>
</table>
Table 5-10  Fine-grained policies related to OBS in system-defined policies

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBS ReadOnlyAccess</td>
<td>Users with this permission can list buckets, obtain basic bucket information, obtain bucket metadata, and list objects. This policy is a system-defined policy of fine-grained authorization. Users with fine-grained authorization can use this policy and can create custom policy template based on this policy.</td>
</tr>
<tr>
<td>OBS OperateAccess</td>
<td>Users with this permission can perform all OBS ReadOnlyAccess operations and perform basic object operations, such as uploading objects, downloading objects, deleting objects, and obtaining object ACLs. This policy is a system-defined policy of fine-grained authorization. Users with fine-grained authorization can use this policy and can create custom policy template based on this policy.</td>
</tr>
</tbody>
</table>

Administrators can set user group policies on the IAM console and add IAM users to different user groups so that users have different permissions.

After an administrator attaches a system-defined policy to a user group, sub-users in the user group have the corresponding permissions to access OBS resources. For details about operations that can be performed on OBS, see Table 5-11.

**NOTE**

When an administrator attaches a policy to a user group, OBS is deployed separately from other services. Authorization must be performed in the OBS project in the Global service area.

Table 5-11  Permissions and the allowed operations on OBS resources

<table>
<thead>
<tr>
<th>Operation</th>
<th>Tenant Administrator Permission</th>
<th>Tenant Guest Permission</th>
<th>OBS Buckets Viewer Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listing buckets</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Creating a bucket</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deleting a bucket</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Obtaining basic bucket information</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Obtaining bucket metadata</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Obtaining monitoring statistics</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Operation</td>
<td>Tenant Administrator Permission</td>
<td>Tenant Guest Permission</td>
<td>OBS Buckets Viewer Permission</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------</td>
<td>-------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Managing bucket access permissions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing bucket policies</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Modifying bucket storage classes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Listing objects</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Listing object versions</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Uploading files</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Creating a folder</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deleting files</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deleting folders</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Downloading files</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deleting object versions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Downloading object versions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Modifying object storage classes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Restoring files</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Undoing a file deletion</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deleting fragments</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing object access permissions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Configuring object metadata</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing versioning</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing logging</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing event notifications</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing tags</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing lifecycle rules</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Operation</td>
<td>Tenant Administrator Permission</td>
<td>Tenant Guest Permission</td>
<td>OBS Buckets Viewer Permission</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------</td>
<td>-------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Managing static website hosting</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing CORS rules</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing URL validation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing domain names</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing cross-region replication</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Managing image processing</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Custom Policies**

Custom policies can be created using the actions supported by cloud services. They will be used as a supplement to system-defined policies to implement more refined access control. You can create custom policies in the visual editor or in JSON view.

Users with the fine-grained authorization permission can customize user policies based on the **OBS ReadOnlyAccess** or **OBS OperateAccess** policy to implement refined user permission management. For details, see [Custom Policies](#).

A policy consists of a Version and a Statement. Each policy can have multiple statements.
Policy Content

The following uses a custom policy as an example to describe the syntax of a fine-grained policy.

```
{
  "Version": "1.1",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "obs:bucket:HeadBucket",
        "obs:bucket:ListBucket",
        "obs:bucket:GetBucketLocation"
      ],
      "Condition": {
        "StringEndsWithIfExists": {
          "g:UserName": "specialCharacter"
        },
        "Bool": {
          "g:MFAPresent": "true"
        }
      },
      "Resource": ["obs:*:*:bucket:*"]
    },
    {
      "Effect": "Allow",
      "Action": ["obs:bucket:ListAllMybuckets"],
      "Resource": ["*"]
    }
  ]
}
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Policy version</td>
<td>● 1.0: RBAC policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 1.1: Fine-grained policy</td>
</tr>
<tr>
<td>Statement</td>
<td>Operations to be performed on OBS</td>
<td>Format: <em>Service name</em>.Resource type*.Operation.</td>
</tr>
<tr>
<td>Permissions</td>
<td></td>
<td>OBS has two resource types: bucket and object. The wildcard character (*) is supported, which indicates either resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For lists of actions supported by OBS, see Bucket-Related Actions and Object-Related Actions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(obs:bucket:ListAllMybuckets): Permissions for listing all OBS buckets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Action obs:bucket:ListAllMybuckets is for all OBS buckets. Therefore, the Resource is * in the example. During visualized configuration, an independent permission needs to be configured for this action, and all resources need to be selected.</td>
</tr>
<tr>
<td>Effect</td>
<td>Determines whether to allow or deny the operation defined in an action.</td>
<td>● Allow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Deny</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the policies contain both Allow and Deny effects for the same action, the Deny effect takes the priority.</td>
</tr>
<tr>
<td>Resource</td>
<td>Resources on which the policy takes effect</td>
<td>Format: <em>Service name</em>.Region*.Domain ID*.Resource type*.Resource path*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The wildcard character (*) is supported, which indicates any options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● obs:<em>:</em>:bucket:*: Any OBS buckets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● obs:<em>:</em>:object:my-bucket/my-object/*: Any objects in the my-object directory of the my-bucket bucket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● The value of Resource supports uppercase (A to Z), lowercase (a to z) letters, digits (0 to 9), and the following characters: _-/<em>/. If the value contains invalid characters, use the wildcard character (</em>).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● OBS is a global service. Set Region to *</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>Condition</td>
<td>Conditions for the policy to take effect</td>
<td>Format: \textit{Condition operator:}{\textit{Condition key:}[\textit{Value 1, Value 2}]} The condition includes the global service condition name and cloud service condition name. The condition names supported by OBS are the same as those in the bucket policy. When configuring in IAM, add \textit{obs:}. For details, see \textit{5.4.2 Policy Format}. Examples: \begin{itemize} \item StringEndWithIfExists&quot;: {&quot;g:UserName&quot;: [&quot;specialCharactor&quot;]}: The statement is valid for users whose names end with \textit{specialCharactor}. \item &quot;StringLike&quot;:{&quot;obs:prefix&quot;: [&quot;private/&quot;]}: When listing objects in a bucket, you need to set prefix to \textit{private/} or include \textit{private/}. \end{itemize} NOTE The value of \textit{Condition} can contain only uppercase (A to Z), lowercase (a to z) letters, digits (0 to 9), and the following characters: \texttt{-./@#$%&amp;}. If the value contains unsupported characters, consider using the condition operator for fuzzy match, such as StringLike, StringStartWith.</td>
</tr>
</tbody>
</table>
You can only set fine-grained policies on the Enterprise Management console. If an RBAC policy is also set for a user group on the IAM console, the result of listing all buckets is determined by the policy configured on the Enterprise Management console.

- When you set a policy for a user group on the IAM console, the policy takes effect for all enterprise projects.
- When you set a policy in Enterprise Project Management and Personnel Management on the Enterprise Management console, the policy takes effect only for the associated enterprise project.

5.5.2 Common Policies

OBS Full-Control Policy

A full-control policy allows sub-users to perform any operations on OBS.

```json
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["OBS:*:*"]
    }
  ]
}
```

Read-only Policy for Any Directory

This policy allows sub-users to list and download all objects in bucket `my-company`.

```json
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["obs:object:GetObject", "obs:bucket:ListBucket"],
      "Resource": ["OBS:*::*:object:my-company/*", "OBS:*::*:bucket:my-company"]
    }
  ]
}
```

Read-only Policy for a Specific Directory

This policy allows sub-users to only download all objects in the `my-project` directory of bucket `my-company`. Objects in other directories can be listed but cannot be downloaded.

```json
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["obs:object:GetObject", "obs:bucket:ListBucket"],
      "Resource": ["OBS:*::*:object:my-company/my-project/*"]
    }
  ]
}
```
Read-Write Policy for a Specific Directory

This policy allows sub-users to list, download, upload, and delete all objects in the `my-project` directory of bucket `my-company`.

```json
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "obs:object:GetObject",
                "obs:object:ListMultipartUploadParts",
                "obs:bucket:ListBucket",
                "obs:object:DeleteObject",
                "obs:object:PutObject"
            ],
            "Resource": [
                "OBS::*:*:object:my-company/my-project/*",
                "OBS::*:*:bucket:my-company"
            ]
        }
    ]
}
```

Full-Control Policy for a Specific Bucket

This policy indicates that an IAM user always has full management permission on a specific bucket.

```json
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:object:*",
                "obs:bucket:*"
            ],
            "Resource": [
                "OBS::*:*:bucket:my-company",
                "OBS::*:*:object:my-company/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

OBS Console Full-Control Policy

This policy indicates that all operations can be performed on buckets on OBS Console. Assign the OBS full-control policy to sub-users, and then configure the Tenant Guest policy for global services and regions.
When a user logs in to OBS Console, the user accesses resources of other services, such as audit information in CTS, acceleration domain names in CDN, and keys in KMS. Therefore, in addition to the OBS permissions, you need to configure the permissions for other services.

In the example, Tenant Guest is configured for the global service and the CN North-Beijing1 region respectively. CDN is a global service, so you need to add permissions to the global service. Cloud Eye, KMS, and SMN are regional services, so you need to add the Tenant Guest permission to the corresponding regions.

### 5.6 Examples About Permission Control

This section describes common permission authorization scenarios and various combinations of authorization schemes.

**Scenario 1: An account can have multiple IAM users. Each user can access only its own bucket.**

If you want to list all buckets owned by an account, you can use only OBS SDK, OBS Browser, or obscmd to implement this function, instead of operating on OBS Console.

**Step 1** Log in to IAM using a tenant that can access OBS, and create an IAM user. After creating an IAM user, add the user to the user group. For details about how to create an IAM user and user group, see [Creating an IAM User](#). Do not grant the OBS permission to the user group.

After the user is created, record the username. The username is used for authorization.

**Step 2** In IAM, create a credential (AK/SK) for the newly created user.
Step 3 Log in to OBS Console as a tenant (an IAM user with the permission to access OBS), and create a bucket. You can also use the access keys (AK and SK) of a tenant (or an IAM user with the permission to access OBS) to log in to OBS SDK or OBS Browser to create a bucket. After the bucket is created, add a policy for the bucket. Do not select any resource for the policy, so that it indicates the policy is applicable to the entire bucket. You can configure bucket related permissions through the policy, such as bucket configuration, querying bucket logs, versioning management, and lifecycle management.

The authorized user is the user created in step 1.

A new policy is added. The action is * (indicating that all objects in the bucket are authorized and objects can be listed, uploaded, downloaded, and deleted).

If you use the OBS API to configure the bucket policy, see Configuring a Bucket Policy.

Step 4 Add an external bucket to OBS Browser+. Skip this step if you use SDK to access OBS.

Open OBS Browser+ and enter the credential created in step 2 for login.

Then add the bucket created in step 3 to OBS Browser+ as an external bucket.

Step 5 After the bucket is added successfully, users who have been granted permissions to this bucket (see step 3) can upload to and download objects from this bucket through OBS Browser+.

---End

Scenario 2: After you have registered with the public cloud, the user has only the read and write permissions for objects in OBS.

Step 1 Log in to IAM using a tenant that can access OBS, and create an IAM user. After creating an IAM user, add the user to the user group. For details about how to create an IAM user and user group, see Creating an IAM User. Do not grant the OBS permission to the user group.

After the user is created, record the user ID. The user ID is used for authorization.

Step 2 Log in to OBS Console as a tenant (an IAM user with the permission to access OBS), and create a bucket. You can also use the access keys (AK and SK) of a tenant (an IAM user with the permission to access OBS) to log in to OBS SDK or OBS Browser to create a bucket. After the bucket is created, a policy is added. The resource is * (indicating that all objects in the bucket are authorized and objects can be listed, uploaded, downloaded, and deleted). The authorized user is the user created in step 1.

If you use the OBS API to configure the bucket policy, see Configuring a Bucket Policy.

Step 3 Log in to OBS Console using the user created in Step 1, enter the bucket, and upload or download objects. Alternatively, you can use OBS SDK to upload and download objects.

---End
Scenario 3: Allow users to display only authorized buckets on the bucket list page.

Users, who have enabled enterprise projects and fine-grained permission control, can implement this scenario.

**Step 1** Log in to IAM using a tenant that can access OBS, and create an IAM user. After creating an IAM user, add the user to the user group. For details about how to create an IAM user and user group, see Creating an IAM User.

**Step 2** Create a user group and add a user to the group on the Enterprise Project page, create an enterprise project, and add a user group to the enterprise project. Select OBS ReadOnlyAccess when configuring the permission control policy, or configure custom policy with the ListAllMyBucket permission. See the following figure:

**Step 3** Log in to OBS Console and open the page where buckets are listed. You can see only buckets in the enterprise project.

- **NOTE**
  1. If the enterprise project information of some buckets in the list is displayed as --, it indicates that the region does not support the functions for enterprise project management and fine-grained permission control. When creating a bucket in an enterprise project, ensure that the region supports the function for enterprise projects management.
  2. A console page usually requires more than one operation permissions. If you have insufficient permissions, information on the page may be incomplete or even the entire page cannot be displayed. Therefore, do not restrict the GET permissions of a user.

----End

Scenario 4: Use a sub-user to upload and download KMS-encrypted objects.

**Step 1** Log in to IAM using a tenant that can access OBS, and create an IAM user. After creating an IAM user, add the user to the user group. For details about how to create an IAM user and user group, see Creating an IAM User. Select the OBS OperateAccess permission when editing user group permissions. This step is to grant sub-users the permission to upload and download objects. You can also configure custom policies.

**Step 2** Edit the user group permissions to add the KMS Administrator permission. KMS is a project-level service. Therefore, you need to select the region where your bucket is located.

---End
Data Transmission

Before arriving at OBS, data needs to be transmitted over the network, which may cause security risks, such as intercepting, tampering, and replay. Therefore, OBS provides the transmission encryption technology.

Transmission encryption technologies include: communication peer authentication, key negotiation, data encryption, and integrity protection, which effectively authenticate the peer end, encrypt transmission data, and ensure data integrity.

OBS supports the HTTPS (HTTP over SSL) protocol to support transmission encryption.

SSL has experienced multiple versions in the development process, including: SSL2.0, SSL3.0, TLS1.0, TLS1.1, and TLS1.2. Considering security and compatibility, OBS supports TLS1.0, TLS1.1, and TLS1.2 3 protocol versions.

If you use an up-to-date browser to access OBS, you do not need to perform any configuration. The browser will automatically select the protocol of a later version. Therefore, it is recommended that you use the latest version of the browser for all the time.

If you use the SDK or customized client to access OBS, you do not need to configure it in most scenarios. However, if your terminal is outdated and does not support the preceding three protocol versions, you cannot connect to OBS. If the connection fails, you are advised to replace the terminal or upgrade the system and try again.
7 Data Protection Mechanisms

7.1 Versioning

Versioning ensures security of objects in a bucket by preventing unpredictable loss due to accidental object deletions or overwrites.

Versioning Status

The status of versioning can be disabled, enabled, or suspended.

**NOTICE**

Once the versioning status of a bucket is enabled, it cannot be restored to the disabled status. However, you can set the versioning status of the bucket to suspended.

Each time an object is created, OBS assigns a version number to the object. The version number generation rule varies with the versioning state of the bucket.

- When the versioning state of a bucket is Disabled or Suspended, the version number of a newly created object is null.
- When the versioning state of a bucket is Enabled, the version number of a newly created object is a randomly generated 32- to 48-byte character string and it is globally unique.

Data Protection Mechanism

Any object saved in OBS, unless being deleted or overwritten by another object with the same name, can always be obtained. However, sometimes you may accidentally delete or overwrite an object and data cannot be restored. Or you may want to save copies for data in different periods and keep a unified object name for such copies. In these two scenarios, you are advised to enable the versioning feature of the bucket.
Table 7-1 Rules for deleting and overwriting objects in different versioning states

<table>
<thead>
<tr>
<th>Versioning Status</th>
<th>Deleting an Object</th>
<th>Overwriting an Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>The object is directly deleted and cannot be obtained.</td>
<td>The object is directly overwritten and cannot be restored. Only the new object can be obtained.</td>
</tr>
<tr>
<td>Enabled</td>
<td>A delete marker is generated for the object. The delete marker also carries a globally unique version number. Historical versions are not affected. <strong>NOTE</strong> No matter whether the object has a historical version, OBS always generates a delete marker, indicating that a deletion has been performed at this time. In this case, If you try to obtain an object without specifying the version number, the system displays a message indicating that the object does not exist. If you try to obtain an object with the version number specified, the target object can be obtained.</td>
<td>A new version is generated for this object, and the historical version is not affected. In this case, if you try to obtain an object without specifying the version number, the object of the latest version is returned. If you try to obtain an object with the version number specified, the target object can be obtained.</td>
</tr>
</tbody>
</table>
### Versioning Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Deleting an Object</th>
<th>Overwriting an Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspended</td>
<td>A delete marker whose version number is <strong>null</strong> is generated for this object. If an object or delete marker whose version number is <strong>null</strong> already exists, it will be overwritten by the new delete marker. Objects or delete markers whose version numbers are not <strong>null</strong> are not affected. <strong>NOTE</strong> No matter whether the object has a historical version, OBS always generates a delete marker, indicating that a deletion has been performed at this time. In this case, if you try to obtain an object without specifying the version number, the system displays a message indicating that the object does not exist. If you try to obtain an object with the version number specified, the target object can be obtained.</td>
<td>An object whose version number is <strong>null</strong> is generated for this object. If an object or delete marker whose version number is <strong>null</strong> already exists, it will be overwritten by the new delete marker. Objects or delete markers whose version numbers are not <strong>null</strong> are not affected. In this case, if you try to obtain an object without specifying the version number, the object of the latest version is returned. If you try to obtain an object with the version number specified, the target object can be obtained.</td>
</tr>
</tbody>
</table>

**NOTE**

For ease of understanding, the version numbers in the following figures are all presented by character strings shorter than actual ones.

As shown in the following figure, when versioning is enabled and an object is repeatedly created, a new version number is generated upon each creation.
As shown in the following figure, when versioning is enabled and an object is deleted, the object is not actually deleted. Instead, a delete marker is generated to indicate that the current version of the object is deleted.

You can view delete markers in a bucket by calling Listing Objects in a Bucket (listing multi-version objects). When the latest version of an object is deleted and the object is uploaded again. The delete marker does not disappear. Instead, it is stored in OBS as a historical version.

As shown in the following figure, when versioning is suspended and an object is created. Historical versions of this object are retained, and the version number of the new object is null. If you create the object again, the latest object with the version null automatically overwrites the previous null version.

Deleting a Specified Version

Object version numbers are allocated by OBS. Therefore, you cannot specify the version number when creating an object. However, when you download or delete an object, you can specify the version number for such operations.

The following figure shows the changes of a multi-version object in a bucket after a specified version of the object is deleted.
If a specified version of an object is deleted by specifying the version number, OBS does not retain the specified historical version of the object and removes it from the object historical version list.

If the object of version **00003** shown in the figure is deleted, the latest version of the object becomes **00001**. In this case, if you try to obtain the object without specifying the version number, the object of version **00001** is obtained.

**Differences Between the Suspended State and the Disabled State**

In both the suspended and disabled states, the version number of a newly created object is **null** and it overwrites the original null version of the object regardless of whether the original null version is generated in the suspended or disabled state. That is, OBS saves only one null version for an object.

The difference lies in the methods of handling deletion operations. In the suspended state, if an object is deleted, a delete marker whose version number is null is generated no matter whether the object has a historical version. However, in the disabled state, the delete marker is not generated.

**NOTE**

After versioning is enabled, all historical versions of an object are stored and they occupy storage space. Storage fees of such historical versions will be charged by OBS. Exercise caution when enabling versioning to avoid extra storage fees.
8 Storage Classes

8.1 Introduction to Storage Classes

OBS has three storage classes: OBS Standard, OBS Infrequent Access, and OBS Archive. With the three storage classes, OBS comprehensively meets customers' requirements on storage performance and cost.

- OBS Standard features low access latency and high throughput. It is applicable to storing frequently accessed (multiple times per month) hot or small files (< 1 MB) requiring quick response. The application scenarios include big data, mobile applications, hot videos, and social media images.
- OBS Infrequent Access is applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response. The application scenarios include file synchronization or sharing, and enterprise-level backup. It provides the same durability, access latency, and throughput as the Standard but at a lower price. The main drawback, however, is that it has lower availability than the Standard.
- OBS Archive is applicable to archiving rarely-accessed (once a year) data. The application scenarios include data archiving and long-term data retention for backup. It provides secure, durable, and inexpensive storage, which can replace tape libraries. However, it may take hours to retrieve data from the Archive storage class.

8.2 Bucket Storage Classes

You can set the storage class for a bucket when creating it. Or the bucket defaults to be in the Standard storage class. The **x-obs-storage-class** header in a bucket creation request specifies the default storage class for a bucket. The storage class of the objects in a bucket defaults to be that of the bucket. OBS provides three bucket storage classes. Therefore, the possible values of the **x-obs-storage-class** header are **STANDARD** (OBS Standard), **WARM** (OBS Infrequent Access), and
COLD (OBS Archive). If this header is not contained, the storage class of the created bucket is Standard.

8.3 Object Storage Classes

When uploading an object, you can use header `x-obs-storage-class` to set the storage class for the object. If you do not use this header, the storage class of the object defaults to be that of its residing bucket.

OBS provides three storage classes. Therefore, the possible values of the `x-obs-storage-class` header are **STANDARD** (OBS Standard), **WARM** (OBS Infrequent Access), and **COLD** (OBS Archive).

The storage class of an object will change according to lifecycle rules if you have enabled lifecycle management. For details about lifecycle management, see 9 Object Lifecycle Management.

If an object is an Archive object, you must retrieve it before downloading it. For details about the API for retrieving an Archive object, see Restoring Archive Objects.
9 Object Lifecycle Management

9.1 Lifecycle Management Overview

Overview

You can configure the lifecycle management function to manage objects’ lifecycles. You can define how OBS manages an object in the lifecycle from creating or initially storing the object to deleting, migrating, or replacing the object. OBS provides many actions to simplify the lifecycle management of an object, which can be categorized as follows:

- Transition: specifies the time when objects will be transited from one storage class to another. For example, you can specify that objects will be transited to Infrequent Access 30 days after creation, or to Archive 1 year after creation.
- Expiration: specifies the expiration time of objects. After doing this, OBS deletes your expired objects.

You can configure lifecycle rules for the objects whose lifecycles are clearly defined. Many objects in a bucket may have a clearly defined lifecycle. The following is an example:

- If you periodically upload logs to your bucket, your applications may use these logs within one week or one month, and then you may want to delete these logs that may not be used again.
- Some documents may be frequently accessed within a specified period. Then you may not access these files in real time, but you need to archive them for a longer time and selectively delete them.
- You may upload data of different types for archive purposes, such as digital media, financial and medical records, original genome sequence data, long-
term database backup, and data that must be retained for regulatory compliance.

### Configuring the Lifecycle

The lifecycle configuration consists of a set of rules defining the actions that can be performed on objects. The lifecycle configuration must be in XML format. OBS saves the lifecycle configuration as a sub-resource attached to the bucket. You can use OBS APIs to initiate PUT, GET, and DELETE requests on lifecycle configuration.

### 9.2 General Precautions

#### About Transition

You can configure the transition function in the lifecycle configuration to allow OBS to transit objects from one storage class to another. For example:

- For infrequently accessed objects, transit them to Infrequent Access.
- For objects involving no real-time access, transit them to Archive.

The supported transition paths are as follows:

- **From Standard to Infrequent Access**
  - The Standard objects must be stored in OBS for at least 30 days before the transition. For example, if you set the transition time to be 1 day after object creation, the transition will fail.
  - If versioning control is enabled, noncurrent object versions can be transited after they have become noncurrent version for more than 30 days.

- **From Standard or Infrequent Access to Archive**
  - Archive objects cannot be accessed in real time.
  - To access an Archive object, you must retrieve it first to create a temporary copy. This copy is valid for a period that you have specified in the retrieval request. Once the copy becomes invalid, OBS deletes it while the source object remains in archive.
  - Archive objects cannot be transited to Standard or Infrequent Access objects.
  - If you have specified the transition times both to Infrequent Access and to Archive, the transition time to Infrequent Access must be at least 30 days earlier than that to Archive. For example, if the time for transition to Infrequent Access is set to 30 days after object creation, the time for transition to Archive cannot be shorter than 60 days after object creation. OBS does not support transition from Archive to Standard. If you want to change the storage class of an object from Archive to Standard or Infrequent Access, retrieve it first to generate a temporary copy. Then, use the copy to overwrite the current object version and set the storage class to Standard or Infrequent Access.

You can use these lifecycle rules together to manage the whole lifecycle of an object, including transition to Infrequent Access, transition to Archive, and expiration.
Suppose you have created objects with a specific lifecycle. At the first 30 days, these objects are frequently accessed. Then, in the next 60 days, these objects are infrequently accessed. And then, real-time access to these objects is no longer needed. You can choose to archive or delete them. In such cases, you can set the transition rules as follows:

- Transit objects to Infrequent Access 30 days after creation.
- Transit objects to Archive 90 days after creation.
- Specify the validity duration of these objects, after which they will expire.

Objects are transited to Archive objects in an asynchronous manner. The time when transition actually takes place may not be exactly the same as the transition time specified in the lifecycle rule.

**About Object Archiving (Transition to Archive)**

For objects requiring no real-time access, OBS provides the Archive storage class that is perfectly suitable for archiving objects.

Every object in OBS is classified into one storage class. When an object is uploaded, OBS associates it with the storage classes of the bucket by default. For example, if the storage class of the bucket is Standard, the object is also Standard by default. You can also specify the storage class of the object as Standard, Infrequent Access or Archive when uploading it.

The lifecycle configuration allows only unidirectional conversion from Standard or Infrequent Access to Archive storage class. If you want to change the storage class of an object from Archive to Standard or Infrequent Access, you must retrieve the object of Archive storage class, copy the object, and specify the storage type of the new object.

**About Expiration**

Some objects stored in a bucket may have defined survival periods. Suppose you periodically upload logs to your bucket and these logs only need to be retained for a certain period. You can set the expiration rules for these logs. After doing this, OBS will permanently or logically delete the expired ones based on the versioning status.

OBS provides the **Expiration** action which allows you to set the expiration time for objects. In addition, for buckets with versioning enabled, OBS provides the **NoncurrentVersionExpiration** operation which allows you to set the expiration time for noncurrent object versions.

For buckets with versioning disabled, the **Expiration** action allows you to delete the expired objects permanently.

If versioning control is enabled (or suspended) for a bucket, the **Expiration** action will logically delete the current version of an object by creating a delete marker for the current version and recognizing the delete marker as the new current version. The **NoncurrentVersionExpiration** action deletes objects of noncurrent versions permanently.

Beside buckets, you can also configure the expiration time for objects during object uploads. The expiration time of objects prevails against that of buckets.
After an object expires, OBS adds the object to the deletion queue and delete it asynchronously. This may cause the deletion time to be later than the expiration time. After an object expires, you will not be charged for the related storage duration fee.

To query the planned expiration time of an object, you can use the GET object API or HEAD object API. The APIs return response headers that provide expiration information about the object.

Lifecycle for Versioning Control

Lifecycle configuration can be added to a bucket with versioning control enabled or disabled. By default, versioning control is disabled. You can enable it. If versioning control is enabled for a bucket, the bucket will maintain the current object version and zero or more noncurrent object versions. You can manage object storage costs using versioning control and lifecycle rules. The predefined lifecycle configuration actions can facilitate management over the lifecycle of the current object versions and non-current object versions.

9.3 Response Elements for Lifecycle Configuration

You can set the lifecycle configuration format to XML. The configuration contains one or more lifecycle rules.

Each rule consists of the following contents:

- Metadata, specifying the rule ID and whether the rule is enabled or disabled. If the rule is disabled, OBS does not perform actions specified in the rule.
- Filtering criteria, specifying the objects to which the rule applies. You can set the object name prefix to be the filtering criteria.
- When (a date or a time period) will one transition or expiration action be performed on objects in the lifecycle.

The following are two examples:

Example 1: Lifecycle configuration of a bucket with versioning control disabled

By default, versioning control is disabled. Each object can have only one version. Assume that you have a bucket with versioning control disabled. You want objects whose name prefix is documents/ to be transited to Infrequent Access objects 30 days after creation, transited to Archive 60 days after creation, and permanently deleted 1 year after creation. To achieve these, you can add the following lifecycle configuration for your bucket. Define a rule that contains the Transition and Expiration actions. Set the Prefix element to documents.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
    <Transition>
      <Days>30</Days>
      <StorageClass>WARM</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```
Example 2: Lifecycle configuration of a bucket with versioning control enabled

You can enable versioning control. If versioning control is enabled for a bucket, the bucket will maintain the current object version and zero or more noncurrent object versions. For more information, see the section about versioning control. Versioning control enables you to maintain the history records of objects and lifecycle management allows you to control the retention of object versions as well as the storage class transition.

For a bucket with versioning control enabled, lifecycle configuration provides multiple predefined actions that can be used to manage noncurrent object versions. In the following example, the lifecycle configuration contains a rule, which specifies the following:

- This rule applies to noncurrent object versions whose name prefix is `logs/`.
- One `NoncurrentVersionTransition` action transits objects to Infrequent Access 30 days after they have become noncurrent versions.
- The other `NoncurrentVersionTransition` action transits objects to Archive 60 days after they have become noncurrent versions.
- The `NoncurrentVersionExpiration` action makes the objects expire 180 days after they have become noncurrent versions.

```
例 config2.xml contains a rule, which specifies the following:
- This rule applies to noncurrent object versions whose name prefix is `logs/`.
- One `NoncurrentVersionTransition` action transits objects to Infrequent Access 30 days after they have become noncurrent versions.
- The other `NoncurrentVersionTransition` action transits objects to Archive 60 days after they have become noncurrent versions.
- The `NoncurrentVersionExpiration` action makes the objects expire 180 days after they have become noncurrent versions.
```

You can use the predefined `NoncurrentVersionTransition` and `NoncurrentVersionExpiration` actions to manage noncurrent versions in your bucket. The following describes the rules and actions in details.

Generally, each lifecycle rule consists of the following parts:

- **Metadata**: specifying the rule ID (`<ID>` element) and whether the rule is enabled or disabled (`<Status>` element). If the rule is disabled, OBS does not perform actions specified in the rule.
- **Prefix** (`<Prefix>` element), which is used for identifying objects that the rule applies to.
- **Actions** that you want to perform on the specified objects (such as `NoncurrentVersionTransition` and `NoncurrentVersionExpiration` mentioned
in the previous example). Each action includes the date when or time period after which the action will be performed.

The following describes the elements in lifecycle rules in details.

- **Element ID**
  One lifecycle configuration can have a maximum of 1000 rules. Each rule has a unique ID.

- **Element Status**
  The value can be **Enabled** or **Disabled**. If a rule is disabled, OBS does not perform any actions defined in the rule.

- **Element Prefix**
  A lifecycle rule applies to one or more objects with the name prefix specified in the rule. Suppose you have the following objects:
  - logs/day1
  - logs/day2
  - logs/day3
  - ExampleObject.jpg
  If you set **Prefix** to **ExampleObject.jpg**, the rule applies only to object **ExampleObject.jpg**. If you set **Prefix** to **logs/**, the rule applies to the first three objects whose name starting with **logs/**. If you leave **Prefix** null, the rule applies to all objects in your bucket.

- **Element Action**
  You can specify predefined actions in the lifecycle rule to perform them on your buckets in the lifecycle. The predefined actions include **Transition**, **Expiration**, **NoncurrentVersionTransition**, and **NoncurrentVersionExpiration**. The action effect depends on the status of versioning control of a bucket.

**NOTE**

By default, bucket versioning control is disabled. You can enable bucket versioning control, so that each object has a concurrent version and may have one or more noncurrent versions. As well, you can disable versioning control. For more information about versioning control, see the section about versioning control.

- **Action Transition**
  This action transits objects to Infrequent Access or Archive. In the specified date or when a certain time period runs out, OBS transits objects that apply to the rule to Infrequent Access or Archive.
  - Bucket with versioning control disabled: The **Transition** action transits the objects to Infrequent Access or Archive.
  - Bucket with versioning control enabled: The **Transition** action transits objects of the current version to Infrequent Access or Archive. This action has no impact on noncurrent object versions.

- **Action Expiration**
  This action makes objects applying to the rule expire. Objects become unavailable once they expire. Whether the expired objects will be permanently deleted depends on the versioning control status of the bucket.
The **Expiration** action will not delete uploaded parts of an incomplete multipart upload.

- Bucket with versioning control disabled: The **Expiration** action deletes objects permanently and the deleted objects cannot be restored.
- Bucket with versioning control enabled: This action applies only to current object versions, instead of noncurrent object versions. This action logically deletes current object versions which are not delete markers, by adding delete markers for them as the new current object versions and saving the original ones as noncurrent object versions. This action will not be performed on current object versions who are delete markers. If the current object version is the only version of the object and has a delete marker, OBS will delete the current object version. Clearing a delete marker may take a while, because OBS needs to confirm that the delete marker is the only object version.

If you initiate a GET request on an object whose current version is a delete marker without specifying the version ID, OBS will identify the object as a deleted one and return error **404 Object Not Found**. But you can specify the version ID in the GET request to restore the deleted object.

For example, you can set a rule to make the object named `photo.gif` expire 5 days after creation. If `photo.gif` is created at 10:30 UTC on January 1, 2016, the expiration rule will be executed at a time point after 00:00 UTC (five days after object creation) on January 7, 2016. The time will not be later than 23:59 UTC on January 7, 2016. For a bucket with versioning disabled, a deletion operation permanently deletes `photo.gif`. For a bucket with versioning enabled, after the expiration rule is executed, `photo.gif` (version 111111) is still stored in the bucket and can be accessed if needed, but the current version (version 4857693) of the object has a delete marker. The original object `photo.gif` turns to be a non-current version. For details about principles of delete markers, see the section about versioning control.

Bucket with versioning control suspended: OBS will create delete markers for expired objects whose version ID is `null`. Any existing `null` versions will be overwritten by new `null` versions, and data associated with this version cannot be restored. For more information about deleting objects from a bucket with versioning control suspended, see the section about versioning control.

- **Actions specific to buckets with versioning control enabled (or suspended)**

  The **Transition** and **Expiration** lifecycle actions can manage the lifecycle of current object versions. The **NoncurrentVersionTransition** and **NonCurrentVersionExpiration** actions can manage the lifecycle of noncurrent object versions.

  The following lifecycle configuration actions can be performed only on buckets with versioning control enabled (or suspended). In a bucket with versioning control enabled, an object may have multiple versions, including a current version and zero or more noncurrent versions. You can use these actions to request the OBS to perform specific operations on noncurrent object versions. These actions do not affect current object versions.

  **NoncurrentVersionTransition**: Specifies the time period after which noncurrent versions will be transited from Standard to Infrequent Access or Archive.
**NoncurrentVersionExpiration**: Specifies the time period after which noncurrent object versions will be permanently deleted. A deleted object cannot be restored.

For example, if you want to enable a five-day period to correct any accidental deletion or overwriting, you can configure an expiration rule so that the object can be deleted 5 days after it has become a noncurrent version.

At 2016-01-01 10:30 (UTC time), you created an object named `photo.gif` whose version ID is 111111. At 2016-01-02 10:30 (UTC time), you accidentally deleted this object and OBS created a delete marker for this object using a new version ID, 4857693. In the next five days, you were allowed to restore the original object `photo.gif` whose version ID is 111111.

At 2016-01-08 00:00 (UTC time), the **NoncurrentVersionExpiration** action permanently deleted object `photo.gif` whose version ID is 111111 (after it became a noncurrent version for 5 days.)

**How does OBS verify the number of days an object has become a noncurrent version**

In a bucket with versioning control enabled, an object may have multiple versions, including a current version and zero or more noncurrent versions. Every time an object is uploaded, the existing current version is saved as a noncurrent version, while the uploaded version becomes the current version. To verify the days an object version has become a noncurrent version, OBS checks the creation time of the corresponding object of the current version. OBS uses the days that the object of the current version has been created as the days that the object of a noncurrent version has existed.

**Restoring a noncurrent version using the lifecycle configuration**

You can use either of the following methods to search for noncurrent object versions:

1. Copy a noncurrent object version to the same bucket. The copied version will become the current version and all object versions are reserved.
2. Delete the current object version permanently. After you delete the current version of an object, a noncurrent version of the object becomes the current version.

If the lifecycle rules are applied to a bucket with versioning control enabled, you are advised not to use the second method. Use method 1 instead.

Due to the consistency syntax of OBS, before the communication mode is changed, a current version that is permanently deleted may not disappear (OBS may be unaware of this deletion action). In addition, the expiration action for noncurrent versions, if you have configured, may delete the noncurrent object versions permanently, including those you want to restore. Therefore, method 1 is more secure.

The following table describes relationship between the actions executed on objects and versioning control status of the buckets to which the objects belong.
Table 9-1 Configuration actions and versioning control status of buckets

<table>
<thead>
<tr>
<th>Operation</th>
<th>Bucket with Versioning Disabled</th>
<th>Bucket with Versioning Enabled</th>
<th>Bucket with Versioning Suspended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>Objects can be transited to Infrequent Access or Archive.</td>
<td>If the current version is not a delete marker and is the latest version, the current version can be transited to Infrequent Access or Archive.</td>
<td>If the current version is not a delete marker and is the latest version, the current version can be transited to Infrequent Access or Archive.</td>
</tr>
<tr>
<td>Expiration</td>
<td>The specified object is deleted and cannot be restored.</td>
<td>If the current version is not a delete marker, a delete marker is created and is regarded as the current version. The existing current version becomes a noncurrent version.</td>
<td>A delete marker is created using the version ID and is regarded as the current version. If the version ID of the current version is null, the current ID is permanently deleted. Otherwise, the current version is saved as a noncurrent version.</td>
</tr>
<tr>
<td>NoncurrentVersionTransition</td>
<td>This action is invalid.</td>
<td>If the current version is not a delete marker and is not the latest version, the current version can be transited to Infrequent Access or Archive.</td>
<td>If the current version is not a delete marker and is not the latest version, the current version can be transited to Infrequent Access or Archive.</td>
</tr>
<tr>
<td>NoncurrentVersionExpiration</td>
<td>This action is invalid.</td>
<td>Noncurrent objects are deleted and cannot be restored once deleted.</td>
<td>Noncurrent objects are deleted and cannot be restored once deleted.</td>
</tr>
</tbody>
</table>
• **Date-based lifecycle rules**

You can specify the execution dates for **Transition** and **Expiration** actions. The dates must conform to the ISO8601 standards and the exact time is always 00:00 (UTC time). If a specified date in the past, the corresponding action will be immediately performed on all objects that apply to this lifecycle.

---

**WARNING**

A lifecycle action with a date specified is not a one-off action. Even if the date has passed, OBS will adopt this action as long as the lifecycle is enabled.

Assume you have specified a date for performing the **Expiration** action to delete all objects (without setting any filtering criteria). On the specified date, OBS makes all objects in the bucket expire. In addition, OBS continues to make all new objects created in the bucket expire. To terminate the **Expiration** action, you have to delete this action from your lifecycle configuration, disable the rule, or delete the rule from the lifecycle configuration.

You cannot create date-based lifecycle rule on OBS Console.

• **Time period-based lifecycle rule**

You can specify how many days after an object is created will the **Transition** or **Expiration** action will be performed on the object. After the days are specified, OBS adds the specified days when setting time for an object and starts calculating the time from 00:00 (UTC time) of the next day. For example, you created an object at 2016-01-15 10:30 (UTC time) and you specified that objects would be transited 3 days after creation, the object would be transited at 2016-01-19 00:00 (UTC time).

---

**NOTICE**

OBS only records the last modification date for each object. On OBS Console, you can view the last modification time (**LastModified**) of an object on the object properties page. After an object is created, the date is the creation date. If the object is replaced, the date will also change. Therefore, the creation date is a synonym of the last modification date.

When using the **NoncurrentVersionTransition** or **NoncurrentVersionExpiration** action, you can specify how many days after an object changes to be a noncurrent version (due to overwrite or deletion) will the action be performed on the object.

After the days are specified, OBS adds the specified days to the date when the creation date of the object version and starts calculating the time from 00:00 (UTC time) of the next day. For example, in your bucket, the current version of an object was created at 2016-01-01 10:30 (UTC time), the later version that replaced this version was created at 2016-01-15 10:30 (UTC time), and you specified that an object would be transited 3 days after becoming a noncurrent version, the object was transited at 2016-01-19 00:00 (UTC time).
When configuring the lifecycle rules, within a rule and for rules whose prefixes have inclusion relationship, Date or Days of Transition and Expiration must be identical.

### 9.4 Lifecycle Configuration Examples

This section provides an example of lifecycle configuration. Each example describes how to specify XML in each example solution.

Example 1: Specify a rule.

The following lifecycle configuration contains a rule with two actions, and requests OBS to transit objects whose name prefix is `projectdocs/` to Infrequent Access 30 days after creation, to Archive 60 days after creation, and delete them 365 days after creation.

```xml
LifecycleConfiguration>
  Rule>
    ID>Transition and Expiration Rule</ID>
    Prefix>projectdocs</Prefix>
    Status>Enabled</Status>
    Expiration>
      Days>365</Days>
    </Expiration>
    Transition>
      Days>30</Days>
      StorageClass>WARM</StorageClass>
    </Transition>
    Transition>
      Days>60</Days>
      StorageClass>COLD</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

Besides time period, you can also specify a date for performing an action. Note that a time period and a date cannot coexist.

---

**NOTICE**

If the lifecycle configuration contains more than one action rule, one object may match multiple rules. In this case, the rule priority on OBS is as follows:

- Permanent deletion takes precedence over transition.
- When an object can be transited to Infrequent Access and Archive at the same time, the object will be transited to Archive.

Example 2: Specify a rule applicable to all objects in a bucket.

In the following example, Prefix is left null and Days is set to 0. No prefix indicates that the rule applies to all objects in the bucket. Because Days is set to 0, every object will be transited to Archive at 00:00 (UTC) in the next day after creation.

```xml
LifecycleConfiguration>
  Rule>
    ID>Archive all object same-day upon creation</ID>
    Prefix></Prefix>
</LifecycleConfiguration>
```
Example 3: Disable a rule.

The following lifecycle configuration specifies two rules, in which one is disabled. OBS performs no action specified in a disabled rule.

```xml
<LifecycleConfiguration>
  <Rule>
    <ID>30 days log objects expire rule</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>30</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>1 year documents expire rule</ID>
    <Prefix>documents/</Prefix>
    <Status>Disabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Example 4: Prefixes can overlap with each other.

The same characters can be used in different prefixes of objects as long as the prefixes specify different actions. The following configuration contains one rule, which specifies that all noncurrent versions (no prefix is specified) in the bucket will be deleted 10 days after the creation. In this way, there are 10 days for a user to restore objects to an earlier version. In the other rule, objects whose name prefix is `logs/` are set to expire 30 days after creation.

```xml
<LifecycleConfiguration>
  <Rule>
    <ID>111</ID>
    <Prefix></Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>10</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>222</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>30</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Example 5: Specify a rule containing actions applicable to objects of noncurrent versions.

The following example contains an action applicable to objects of noncurrent versions in a bucket with versioning control enabled.

```xml
<LifecycleConfiguration>
  <Rule>
    <!-- Rule content -->
  </Rule>
</LifecycleConfiguration>
```
9.5 Setting Lifecycle Configuration

This section introduces how to manage object lifecycles on OBS Console.

You can set lifecycle rules for buckets on OBS Console. OBS Console provides the lifecycle management option on the bucket properties page, as illustrated in the following figure.

Using the RESTful APIs to Manage the Lifecycle of Objects

You can set lifecycle rules for a bucket on OBS Console. If your application needs it, you can send REST requests directly. For details about RESTful APIs related to lifecycle configuration, see Object Storage Service API Reference.

- Configuring Bucket Lifecycle Rules
- Obtaining Bucket Lifecycle Configuration
- Deleting Lifecycle Rules
10 Static Website Hosting

10.1 Principles of static website hosting

OBS allows static websites to be hosted on buckets and supports index page, error page display, and page redirection. After static website hosting is enabled, users can access objects in a bucket by using either of the following methods:

- **Resource management**: Users can access objects in a bucket through REST API using a common domain name. In this mode, users can perform uploading, downloading, deleting, and listing operations on buckets or objects.
- **Static website**: Users can access objects in a bucket by using a specific domain name. In this mode, display of the index page, error pages and requested pages is implemented according to the configured static domain name hosting rules.

The modification of static website hosting takes effect within two minutes. Then users can access static resources using the following access domain names:

- `https://bucket name.static website hosting domain name/object name`
- `http://bucket name.static website hosting domain name/object name`

**NOTE**

Avoid using periods (.) in the destination bucket name. Otherwise, failures in client authentication certificate may occur when users use HTTPS for access.

To allow your clients to access the content on the website terminal node, you must make all of your content public and accessible. You can use bucket policies or ACLs on objects to grant necessary permissions.

The differences between the resource management mode and static website mode are as follows:
### 10.2 Configuring static website hosting

**Overview**

To configure a bucket for static website hosting, add the site configuration to the bucket. The configuration includes the following information:

**Index document**

When you enter a URL such as `http://example.com`, you are not requesting a specific page. In this case, the Web server will provide a default page that contains the directory that stores the requested website content. This default page is called an index document, and in most cases it is named as `index.html`. When you configure a bucket for website hosting, you must specify an index document. When a request is sent to the root domain or any subfolder, OBS returns this index document.

**Error document**

If an error occurs, OBS returns an HTML error document. For 4XX errors, you can selectively provide your own customized error document. You can provide other guides to your users in this document.

**Redirection of all requests**

If the root domain is `example.com` and you need to respond to requests from `http://example.com` and `http://www.example.com`, you can create two buckets named `example.com` and `www.example.com`. Then you can retain the website content in only one bucket (such as `example.com`), and configure the other bucket to redirect all requests to the `example.com` bucket.

**Redirection based on advanced conditions**

You can redirect requests based on the specific object name or prefix in the request, or based on the response code. For example, assume that you delete or
rename an object in a bucket. You can add a routing rule that redirects requests to other objects.

The syntax format is as follows by setting redirection rules of specific requests:

```xml
<WebsiteConfiguration>
  <IndexDocument>
    <Suffix>index.html</Suffix></IndexDocument>
  <ErrorDocument>
    <Key>SomeErrorDocument.html</Key>
  </ErrorDocument>
  <RoutingRules>
    ..........RoutingRules......
  </RoutingRules>
</WebsiteConfiguration>
```

<table>
<thead>
<tr>
<th>Table 10-1 Elements for redirecting all website requests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>WebsiteConfiguration</td>
</tr>
<tr>
<td>RedirectAllRequestsTo</td>
</tr>
<tr>
<td>HostName</td>
</tr>
<tr>
<td>Protocol</td>
</tr>
</tbody>
</table>

The syntax format of **RoutingRules** is as follows. The content in square brackets is optional.

```xml
<RoutingRules> =
  <RoutingRules>
    <RoutingRule>...</RoutingRule>
    ..........RoutingRules......
    <RoutingRule>..."</RoutingRule> ...
  </RoutingRules>
</RoutingRules>
```
### Table 10-2 Elements for adding rules that redirect website requests

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebsiteConfiguration</td>
<td>Indicates the root element for the website configuration. Type: container</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Ancestor: none</td>
<td></td>
</tr>
<tr>
<td>IndexDocument</td>
<td>Indicates the container for the <strong>Suffix</strong> element. Type: container</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Ancestor: WebsiteConfiguration</td>
<td></td>
</tr>
<tr>
<td>Suffix</td>
<td>Indicates a suffix that is appended to a request initiated for a directory on the website endpoint. For example, if the suffix is <strong>index.html</strong> and you request for <strong>samplebucket/images/</strong>, the data that is returned will be for the object with the key name <strong>images/index.html</strong> in the samplebucket bucket. The suffix cannot be empty or contain slashes (/). Type: string</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Ancestor: IndexDocument</td>
<td></td>
</tr>
<tr>
<td>ErrorDocument</td>
<td>Indicates the container for the <strong>Key</strong> element. Type: container</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Ancestor: WebsiteConfiguration</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Key</td>
<td>Indicates the object name that is used when a 4XX error occurs. This element identifies the page that is returned when a 4XX error occurs. Type: string Ancestor: ErrorDocument Condition: Required when ErrorDocument is specified.</td>
<td>No</td>
</tr>
<tr>
<td>RoutingRules</td>
<td>Indicates the container for the RoutingRule element. Type: container Ancestor: WebsiteConfiguration</td>
<td>No</td>
</tr>
<tr>
<td>RoutingRule</td>
<td>Indicates the container for a routing rule. A redirection rule contains a Condition and a Redirect. When the Condition is matched, Redirect takes effect. Type: container Ancestor: RoutingRules Condition: In a RoutingRules container, there must be at least one RoutingRule element.</td>
<td>Yes</td>
</tr>
<tr>
<td>Condition</td>
<td>Indicates the container for describing a condition that must be met for the specified redirect to apply. Type: container Ancestor: RoutingRule</td>
<td>No</td>
</tr>
<tr>
<td>KeyPrefixEquals</td>
<td>Indicates the object name prefix when the redirection is applied. For example: To redirect the request for object ExamplePage.html, the key prefix is set to ExamplePage.html. Type: string Ancestor: Condition Condition: Required when the ancestor element Condition is specified and sibling HttpErrorCodeReturnedEquals is not specified. If two conditions are specified, both conditions must be true for the Redirect to be applied.</td>
<td>No</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Mandatory</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>HttpErrorCodeReturnedEquals</td>
<td>Indicates the HTTP error code returned when the Redirect is applied. The specified redirect is applied only when the error code returned equals this value. For example: If you want to redirect to NotFound.html when HTTP error code 404 is returned, set HttpErrorCodeReturnedEquals to 404 in Condition and ReplaceKeyWith to NotFound.html in Redirect.</td>
<td>No</td>
</tr>
<tr>
<td>Redirect</td>
<td>Indicates the container for redirection information. You can redirect requests to another host, to another web page, or with another protocol. You can specify an error code to be returned after an error.</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>Indicates the protocol used in the redirection request.</td>
<td>No</td>
</tr>
<tr>
<td>HostName</td>
<td>Indicates the host name used in the redirection request.</td>
<td>No</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Mandat ory</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>ReplaceKeyPrefixWith</td>
<td>Indicates the object name prefix used in the redirection request. For example: To redirect all requests for (objects under) docs to (objects under) documents, set KeyPrefixEquals to docs in Condition and ReplaceKeyPrefixWith to documents in Redirect.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ancestor: Redirect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Condition: Not required if one of the siblings is present. Can be present only if ReplaceKeyWith is not provided.</td>
<td></td>
</tr>
<tr>
<td>ReplaceKeyWith</td>
<td>Indicates the object name used in the redirection request. For example, redirect requests to error.html. Type: string</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Ancestor: Redirect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Condition: Not required if one of the siblings is present. Can be present only if ReplaceKeyPrefixWith is not provided.</td>
<td></td>
</tr>
<tr>
<td>HttpRedirectCode</td>
<td>Indicates the HTTP status code returned after the redirection request. Type: string</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Ancestor: Redirect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Condition: Not required if one of the siblings is present.</td>
<td></td>
</tr>
</tbody>
</table>

Examples:

Example 1: Modifying the object name prefix for redirection.

Assume that your bucket contains the following objects:

- index.html
- docs/article1.html
- docs/article2.html

You decide to change the folder name from docs/ to documents/. After the modification, the request for an object with prefix /docs needs to be redirected to another with documents/. For example, the request for docs/article1.html needs to be redirected to documents/article1.html.

In this case, you can add the following routing rules to the website configuration:
Example 2: Redirecting requests destined for deleted folders to a specified page

Assume that you have deleted the images/ folder, that is, you have deleted all objects whose object name prefix is images/. You can add a routing rule that redirects the requests of all objects whose prefix is images/ to the page named folderdeleted.html.

Example 3: Redirecting for HTTP errors

Assume that the requested object is not found, and the request needs to be redirected to www.example.com. You can add redirection rules so that site visitors can redirect to www.example.com when HTTP status code 404 (not found) is returned. The following example also inserts the object name prefix report-404/ into redirection. For example, if you request the page ExamplePage.html and it causes an HTTP 404 error, the request will be redirected to the page report-404/ExamplePage.html on the www.example.com. If there is no routing rule and HTTP error 404 occurs, the error document specified in the configuration is returned.

Index document support

An index document is a web page returned when a request is sent to the root of a site or any subfolder. For example, if the user enters http://www.example.com in the browser, the user does not request any specific page. In this case, OBS provides index documents, which are sometimes referred to as default pages.

When configuring your bucket as a website, provide the name of the index document. You must upload the object with this name and configure it to be public and accessible.
Trailing slash in the root URL is optional. For example, if you configure an index.html website as an index document, any of the following URLs will return to index.html.

http://bucketname.obs-website.example.com
http://bucketname.obs-website.example.com

In OBS, objects are horizontally stored in buckets. It does not provide any hierarchical organization as a file system on a computer. You can create a logical hierarchy by using the object name representing the folder structure. For example, consider a bucket with three objects and the following object names.

- sample1.jpg
- photos/2006/Jan/sample2.jpg
- photos/2006/Feb/sample3.jpg

Although they are not stored as any physical hierarchical organization, you can infer the following logical folder structure from the object name.

- The sample1.jpg object is located at the root level of the bucket.
- The sample2.jpg object is located in the photos/2006/Jan subfolder.
- The sample3.jpg object is located in the photos/2006/Feb subfolder.

**Customized error document support**

The following table lists the subset of the HTTP response code returned by OBS when an error occurs.

**Table 10-3 Error codes**

<table>
<thead>
<tr>
<th>HTTP error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301 Moved Permanently</td>
<td>When a user sends a request to an endpoint of OBS, a 301 Moved Permanently response is returned.</td>
</tr>
<tr>
<td>302 Found</td>
<td>When OBS receives a request for key x that does not end with a slash, it searches for the object whose key name is x. If the object is not found, OBS determines that the request is sent for subfolder x. Then OBS redirects the request by adding a slash at the end of the request, and returns 302 Found.</td>
</tr>
<tr>
<td>304 Not Modified</td>
<td>OBS users request the If-Modified-Since and If-Unmodified-Since headers to determine whether the requested object is the same as the cached copy stored on the client. If the objects are the same, the website terminal node returns a 304 Not Modified response.</td>
</tr>
<tr>
<td>HTTP error code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>403 Forbidden</td>
<td>When the user request is converted into an object that cannot be publicly read, the response of the website terminal node includes <strong>403 Forbidden</strong>. The object owner must use bucket policies or ACLs to make the object public and accessible.</td>
</tr>
</tbody>
</table>
| 404 Not Found    | The response of the website terminal node contains **404 Not Found** due to the following reasons:  
  - OBS determines that an invalid object key is referenced in the website URL.  
  - OBS determines that the request is for an index document that does not exist.  
  - The bucket specified in the URL does not exist.  
  - The bucket specified in the URL exists but is not configured as a website.  
You can create customized documents returned for **404 Not Found**. Ensure that the document has been uploaded to the bucket configured as a website and the website hosting configuration has been set to use the document. |
| 500 Service Error| When an internal server error occurs, the response of the website terminal node contains **500 Service Error**. |
| 503 Service Unavailable | When the OBS determines that you need to reduce the request frequency, the response of the website terminal node contains **503 Service Unavailable**. |

### Permissions required for website access

When you configure a bucket as a website, you must make the object to be provided public and accessible. To perform this operation, you need to write a bucket policy that grants GetObject permissions to everyone. If the object requested by the user does not exist on the terminal node of the website, OBS returns the HTTP response code **404 Not Found**. If the object exists but you do not grant read permission for the object, the website terminal node returns HTTP response code **403 Access Denied**. Users can use this response code to infer...
whether a particular object exists. If you do not need this function, do not enable the bucket website support.

The following example bucket policy grants each person the permission to access objects in a specified folder. For details about bucket policies, see **5.4 Bucket Policy**.

```json
{
    "Statement": [ {
        "Sid": "PublicReadGetObject",
        "Effect": "Allow",
        "Principal": { "ID": "*" },
        "Action": [ "GetObject" ],
        "Resource": [ "example-bucket/*" ]
    } ]
}
```

**NOTICE**

A bucket policy applies only to objects owned by a bucket owner. If the bucket contains objects that are not owned by the bucket owner, use object ACLs to grant the public READ permission to the objects.

You can use bucket policies or object ACLs to grant public read permissions to your objects. To use ACLs to make objects public and accessible, you can grant READ permission to everyone, as shown in the following authorization elements. You can add the authorization element to object ACLs. For details about managing ACLs, see **5.3 ACL**.

```xml
<Grant>
  <Grantee>
    <Canned>Everyone</Canned>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

### 10.3 Configuring Flash for Cross-domain Access

By default, the OBS system is configured to support cross-domain access using the root domain name. This allows access from all domains, exposing clients to attacks.

The client can create a `crossdomain.xml` with its own rules in its own bucket and add `Security.loadPolicyFile("http://bucket.obs.region.myhuaweicloud.com/crossdomain.xml")` to its flash code to avoid the risk of being attacked.

`crossdomain.xml` needs to comply with the XML syntax rules, and there is only one root node `cross-domain-policy` without any property. The root node can contain only the following sub-nodes: `site-control`, `allow-access-from`, `allow-access-from-identity`, and `allow-http-request-headers-from`. The following table lists description about the sub-nodes.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>site-control</td>
<td>Checks the attribute value and determines whether other policy files can be loaded.</td>
</tr>
<tr>
<td></td>
<td>The attribute values can be:</td>
</tr>
<tr>
<td></td>
<td><strong>none</strong>: loadPolicyFile cannot be used to load any policy file.</td>
</tr>
<tr>
<td></td>
<td><strong>master-only</strong>: Only the master policy file [default] can be used.</td>
</tr>
<tr>
<td></td>
<td><strong>by-content-type</strong>: Only loadPolicyFile can be used to load the file whose Content-Type is text/x-cross-domain-policy over HTTP/HTTPS as the cross-domain policy file.</td>
</tr>
<tr>
<td></td>
<td><strong>by-ftp-filename</strong>: Only loadPolicyFile can be used to load file crossdomain.xml over FTP as the cross-domain policy file.</td>
</tr>
<tr>
<td></td>
<td><strong>all</strong>: loadPolicyFile can be used to load any file of the target domain as the cross-domain policy file.</td>
</tr>
<tr>
<td>allow-access-from</td>
<td>Checks the attribute value and determines the source domain of the flash file that can access content of the domain.</td>
</tr>
<tr>
<td></td>
<td>The attribute values can be:</td>
</tr>
<tr>
<td></td>
<td><strong>domain</strong>: This property specifies an IP address, a domain, or a wildcard domain (any domain). Only domains specified in domain have the permission to access the content of the domain using the flash file.</td>
</tr>
<tr>
<td></td>
<td><strong>to-ports</strong>: Socket connection ports that can access content of the domain.</td>
</tr>
<tr>
<td></td>
<td><strong>secure</strong>: Indicates whether information is transmitted through encryption.</td>
</tr>
<tr>
<td>allow-access-from-identity</td>
<td>Allows a source domain that has a specific certificate to access resources in this domain.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| allow-http-request-headers-from | Grants permission to a third-party domain to sent data to the domain in HTTP header format.  
The attribute values can be:  
**domain**: This property specifies an IP address, a domain, or a wildcard domain (any domain). Only domains specified in domain have the permission to access the content of the domain using the flash file.  
**headers**: A list separated by commas (,), indicating HTTP headers to be sent. Wildcard (*) can be used to indicate the HTTP header.  
**secure**: Indicates whether information is transmitted through encryption. |
11 Cross-Region Replication

Introduction

You may need to back up objects between regions to enhance data reliability or achieve proximate access. Cross-region replication is able to meet such a need. Based on user-defined settings, OBS can automatically replicate specified objects to a designated region. A cross-region replication process occurs between a source bucket and a destination bucket. (For example, configure cross-region replication for bucket A in region 1, and then replicate objects from bucket A to bucket B that resides in region 2. In this circumstance, bucket A is called the source bucket and bucket B the destination bucket.) The two buckets can belong to the same OBS account or different OBS tenants.

Cross-region replication refers to the automatic and asynchronous replication of objects across buckets in different regions. The object replica in the destination bucket is the exact replica of the object replicated from the source bucket. The two objects have the same object name as well as the same metadata including the creation time, owner, user-defined metadata, version, storage class, and ACL.

To implement cross-region replication, set a cross-region replication policy for the source bucket, and OBS replicates objects according to the policy. The policy involves the following settings:

- The destination bucket to which you want OBS to replicate objects.
- The object that you want to replicate.

You can configure OBS to replicate all objects, or some objects (filtered by a key name prefix). For example, you can configure OBS to replicate only those objects whose names start with Doc/. Then OBS will replicate those objects whose names start with Doc/doc1 and Doc/doc2 but not those objects whose names start with Host/doc3.

- By default, OBS uses the storage class of the source object to create an object replica. You can also select a desired storage class to create an object replica.

Some other settings are also customizable.

Unless with user-defined settings, the object replica in the destination bucket is identical to the source object in the source bucket. For example:

- The object replica has the same key name and metadata (creation time, user-defined metadata, and version ID).
Unless you select a different storage class, OBS will create an object replica with the same storage class with the source object.

If the object replica will be still owned by the source object owner, OBS also replicates the object ACL when replicating an object.

If you select a different storage class, the object replica will have different metadata from the source object. For example, if object a in source bucket A is in the Standard storage class but the Archive storage class is selected for its replica, object replica a1 in the destination bucket B will be in the Archive storage class.

OBS uses SSL to encrypt all the data in flight during a cross-region replication.

Each object can be replicated to only one bucket, and cannot be repetitively replicated. If you change the destination bucket in a cross-region replication policy after a replication, OBS does not replicate it again.

Scenario

You may initiate a cross-region replication for the following considerations:

- **Regulatory compliance**
  OBS stores data across AZs that are relatively far apart from each other. However, regulatory compliance may require further distances. Cross-region replication enables OBS to replicate data across regions for regulator compliance.

- **Minimized latencies**
  You are near to two regions. For a shorter latency, you need to access objects from the nearer region.

- **Easy maintenance**
  You have a computing cluster across regions to analyze the same collection of objects. You need to maintain object replicas in the two regions.

- **Data replication**
  You need to replicate data from one data center to another.

- **Data backup and disaster recovery**
  You have stringent requirements on data security and availability. All data written to one data center must have backups in another data center. In case of natural disasters such as earthquakes and tsunamis in the source data center, object replicas in the backup data center can be used to recover data.

Application Limitations

Note the following when using cross-region replication:

- The source bucket and destination bucket must be in the same versioning status.
  - Versioning is disabled for both the source bucket and destination bucket.
  - Versioning is enabled for both the source bucket and destination bucket.
  - Versioning is suspended for both the source bucket and destination bucket.
The source and destination buckets must be in different regions.

OBS must be authorized by you to replicate objects from one bucket to another. That is to say, you need to create an OBS agency and assign OBS the Tenant Administrator permission.

If the source bucket and the object have different owners, the object owner needs to use the object ACL to assign the source bucket owner the READ and READ_ACP permissions. If the source bucket and destination bucket have different owners, the destination bucket owner needs to set a bucket policy to assign the source bucket owner the ReplicateObject and ReplicateDelete permissions.

Each object in a bucket can be replicated to only one bucket, and cannot be replicated to any other bucket. The newly generated objects in the destination bucket are called object replicas. For example, replicate object a from bucket A to bucket B. The newly generated object a1 is called an object replica. If you change the destination bucket of A to C, object a cannot be replicated to C. If you set the destination bucket of B to D, object a1 cannot be replicated to D.

Source objects cannot be at the OBS Archive storage class but destination objects can.

Objects in the destination bucket can be at the OBS Standard, OBS Infrequent Access, or OBS Archive storage class. If the region of the destination bucket does not support OBS Infrequent Access or OBS Archive, objects are at the OBS Standard storage class.

Operations on object ACLs in the source and destination regions are sorted in time order, and the latter operation overrides the former one. For example, if the ACLs of the source object and that of the object replica are modified concurrently, the source ACL may override the destination ACL.

Only when versioning is enabled for the source bucket and destination bucket and no version ID is specified when deleting objects from the source bucket, replicas of the objects will be deleted synchronously from the destination bucket. In other situations, deleting objects from the source bucket will not synchronously delete their replicas from the destination bucket.

If you want to replicate KMS encrypted objects, you need to authorize the IAM agency the KMS Administrator permissions in the regions where the source and destination buckets reside. After an object (encrypted with any key hosted in KMS and uploaded to the source bucket) is replicated to the destination bucket, the key that encrypts the object changes to the default master key obs/default of the region where the destination bucket resides.

Objects cannot be replicated under the following conditions:

- If you change the versioning status of the destination bucket, object replication will fail. If you want to change the versioning status of the source bucket, you need to cancel the cross-region replication configuration first.
- If you delete the OBS agency configuration in a cross-region replication, the replication status becomes FAILED.
- If KMS is not enabled in the destination region or the agency does not have the KMS Admin permissions in the region where the source bucket and destination bucket reside, the KMS encrypted objects in the source bucket cannot be replicated, and the replication status becomes FAILED.
- If an error occurs in configuring the read/write permission for the source or destination bucket, and thus OBS has no permission to read the source object.
or write the destination object, the replication status becomes **FAILED**. Even after the permission configuration is corrected, the objects that have failed to be replicated are not replicated again and the new permission configuration applies only to newly uploaded objects.

You also need to pay attention to the following:

- If you use the lifecycle management function to configure the expiration or archiving time for the source bucket, the replication of some objects may be incomplete and these objects may be deleted or archived. Under this condition, those objects fail to be replicated to the destination bucket.

- When versioning is not enabled for the source bucket and destination bucket, if an object is deleted from the source bucket while it is being replicated to the destination bucket, it may still be successfully replicated to the destination bucket though it is deleted from the source bucket. When versioning is enabled for the source bucket and destination bucket, if an object version is deleted from the source bucket meanwhile it is replicated to the destination bucket, it may still be successfully replicated though it is deleted from the source bucket.

- Objects in the destination bucket may have a different version sequence from namesake objects generated from a cross-region replication, it is not recommended to upload objects whose names are identical to cross-region replication objects to the destination bucket.

### Contents Replicated

OBS replicates the following contents:

- Server-side encrypted objects (encrypted in the SSE-KMS mode) if replicating encrypted objects is configured in the rule. Replicas are also encrypted. But objects encrypted in the SSE-C mode are not replicated.

- Object metadata

- OBS replicates only the objects that the bucket owner has permission to read and the objects in the source bucket of the ACL.

- If the ACL of a source object changes, the changes are synchronized to the replica of the source object.

- OBS also replicates the logs generated after the bucket logging is enabled.

- In the following situations, objects deletion is synchronized to the destination bucket:
  
  If versioning is enabled and a DELETE request is initiated for an object without specified version ID in the source bucket, instead of deleting the object permanently, OBS adds the Delete Marker to the object. (The object with the Delete Marker cannot be obtained if you do not specify a version ID. For details, see **7.1 Versioning**.) The destination bucket will be synchronized with this operation. The object cannot be listed or downloaded from the source bucket, but can be recovered from the deleted objects list on OBS Console.

### Contents Not Replicated

OBS does not replicate the following contents:
- OBS does not replicate the objects that have been uploaded before cross-region replication is enabled.
- Modifications on the ACL, policy, or lifecycle of the source bucket are not used in the destination bucket. For example, if you modify the lifecycle configuration of the source bucket or add a notification configuration or a bucket policy to the source bucket, these modifications are not used in the destination bucket.
- If versioning is disabled and a DELETE request is initiated for an object in the source bucket, the object is deleted from the source bucket but the deletion is not synchronized with the destination bucket. If versioning is enabled and a DELETE request is initiated for an object with a specified version ID in the source bucket, this object version is deleted from the source bucket but the deletion is not synchronized with the destination bucket. (This object version is not deleted from the destination bucket.) Such a design prevents malicious data deletions.
- Only user operations are replicated. Operations triggered by lifecycle rules are not replicated. For example, if a lifecycle rule is configured for the source bucket, OBS will label expired objects with Delete Markers, but those labels will not be replicated. If lifecycle rules are configured for objects in the source bucket, those lifecycle rules will not be carried over to object replica in the destination bucket.
- Do not delete or overwrite object replicas in the destination bucket, or modify their ACLs. Otherwise the latest objects versions and their ACL settings in the destination bucket may be inconsistent with those in the source bucket.

**Setting Cross-Region Replication**

Cross-region replication requires a source bucket and a destination bucket. The source bucket and destination bucket must be in different regions, and their versioning statuses must be the same. Each object can be replicated to only one bucket.

**Setting Cross-Region Replication Between Buckets Owned by the Same OBS Account**

Configure cross-region replication as follows:
- In the OBS account, create an agency to assign OBS the permission to replicate objects.
- Configure cross-region replication for the source bucket.

**Creating an OBS agency**

Assign the needed permission to OBS for OBS to replicate objects from the source bucket to the destination bucket.

---

**NOTICE**

By default, all OBS resources including buckets, objects, and subresources are private, and only the resource owner can access the resources. Therefore, OBS needs the permission to read objects from the source bucket and replicate them to the destination bucket.
**NOTE**

For details about how to create an OBS agency, see “Creating an Agency” in the *Identity and Access Management User Guide*. You can also go to the OBS console and open the *Identity and Access Management* page to create an OBS agency. Select OBS and assign the Tenant Administrator permission. If the object to be replicated is KMS encrypted, you need to authorize OBS the KMS Administrator permission in both the source and destination regions.

### Adding the cross-region replication configuration

Set the following rule:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
  <Agency>AgencyName</Agency>
  <Rule>
    <Status>Enabled</Status>
    <Prefix/></Prefix>
    <Destination>
      <Bucket>destinationbucket</Bucket>
      <StorageClass>STANDARD</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

- **AgencyName**: the OBS agency name.
- **Status**: indicates whether the rule is valid or not.
- **Prefix**: If it is left blank, the rule applies to all objects in the bucket.
- **Bucket**: the destination bucket of cross-region replication.
- **StorageClass**: the storage class of the object replica. If it is left blank, OBS uses the storage class of the source object for the object replica.
- You can set more than one rule, and specify a different key name prefix for each rule. These prefixes indicate which objects in the source bucket use a rule. In each rule, OBS replicates only the objects with the specified key name prefix. Do not specify repetitive prefixes.
- All rules must have the same destination bucket.
- A newly created bucket is available for being selected as a destination bucket a few minutes after its creation.

### Setting Cross-Region Replication Between Buckets Owned by Different OBS Tenants

If the source bucket and destination bucket are owned by different OBS tenants, the cross-region replication configuration is basically the same as that for cross-region replication between buckets owned by the same OBS account. In addition, the destination bucket owner needs to add a bucket policy, assigning the replication permission to the source bucket owner.

Set a bucket policy for the destination bucket, assigning the source bucket owner the ReplicateObject and ReplicateDelete permissions on the destination bucket:

```json
{
  "Version": "2008-10-17",
  "Statement":
  {
    "Effect": "Allow",
    "Principal": {
      "ID": "/domain/sourceBucketOwnerID"
    }
  }
}
```
Configuring Cross-Region Replication Between Buckets Owned by the Same OBS Account

In this part, you need to create the source bucket and destination bucket in separate regions in OBS, keep their versioning statuses the same, and configure the cross-region replication rule for the source bucket.

1. In the OBS console, configure OBS as an agency.
2. Create a source bucket and a destination bucket for the source region and the destination region respectively. For details, see Creating a Bucket.
   a. Create a source bucket in one OBS region.
   b. Create a destination bucket in the other OBS region.
3. Ensure that the versioning statuses of the source bucket and the destination bucket are the same. For details, see Versioning Overview.
4. Configure cross-region replication:

   ```xml
   <ReplicationConfiguration>
     <Agency>crr_agency</Agency>
     <Rule>
       <ID>Rule-1</ID>
       <Status>Enabled</Status>
       <Prefix>key</Prefix>
       <Destination>
         <Bucket>example_target_bucket</Bucket>
         <StorageClass>STANDARD</StorageClass>
       </Destination>
     </Rule>
   </ReplicationConfiguration>
   ```

   - **crr_agency**: the OBS agency name.
   - **Rule**: the replication rule.
   - **Status**: the cross-region replication status.
   - **Prefix**: the name prefix of the objects to be replicated.
   - **Destination**: destination bucket information. **StorageClass**: the storage class of the objects to be replicated.

Configuring Cross-Region Replication Between Buckets Owned by Different OBS Tenants

Enable versioning for the two buckets and configure cross-region replication for the source bucket. Configure a bucket policy for the destination bucket, assigning the source bucket owner the permission to replicate objects to the destination bucket.

Before configuring cross-region replication, create an OBS agency.
NOTE

For details about how to create an OBS agency, see "Creating an Agency" in the Identity and Access Management User Guide. You can also go to the OBS console and open the Identity and Access Management page to create an OBS agency. Select OBS and assign the Tenant Administrator permission. If the object to be replicated is KMS encrypted, you need to authorize OBS the KMS Administrator permission in both the source and destination regions.

Proceed to the following steps:

1. Use different accounts to create a source bucket and a destination bucket for the source region and the destination region respectively. For details, see Creating a Bucket.
   a. Create a source bucket in one OBS region.
   b. Create a destination bucket in the other OBS region.
2. Ensure that the versioning statuses of the source bucket and the destination bucket are the same. For details, see Versioning Overview.
3. Configure a bucket policy for the destination bucket.
   ```
   { 
   "Version": "2008-10-17",
   "Statement": 
   { 
   "Effect": "Allow",
   "Principal": { 
   "ID": "[domain/sourceBucketOwnerID]"
   },
   "Action": [ 
   "ReplicateObject",
   "ReplicateDelete"
   ],
   "Resource": "destination-bucket/*"
   }
   }
   ```
4. Configure cross-region replication for the source bucket. For details, see Configuring Cross-Region Replication for a Bucket in the Object Storage Service API Reference.

Cross-region replication refers to the automatic and asynchronous replication of objects across buckets in different regions. By enabling cross-region replication, OBS can replicate objects upon the upload from a source bucket to a destination bucket in another region, and synchronize updates with the destination bucket. The cross-region replication has specific requirements. Based on the object creation mode and encryption mode, you can define the objects that can or cannot be replicated across regions.

The following two requirements must be met when configuring the cross-region replication for a bucket:

1. Cross-region replication can be configured when only the versioning statuses of the source bucket and destination buckets are the same.
2. The agent and owner of the source bucket must have the write permission for the destination bucket (the bucket policy must be configured for the destination bucket), and the agent must also have the read permission for the source bucket. This permission delegation needs to be implemented through the bucket policy.
Agent Permission

The agent permission can be configured by configuring bucket policy.

- **If the source bucket and destination bucket have different owners, configure the bucket policy for the destination bucket as follows:**

  ```json
  {  
    "Version": "2008-10-17",  
    "Statement": {  
      "Effect":"Allow",  
      "Principal": {  
        "ID": ["domain/sourceBucketOwnerID"]  
      },  
      "Action": [  
        "ReplicateObject",  
        "ReplicateDelete"  
      ],  
      "Resource": "destination-bucket/*"  
    }  
  }
  ```

After the bucket policy is set, the agent (OBS) has the rights to read objects in the source bucket and copy objects to the destination bucket.

Cross-Region Replication

The configuration of bucket cross-region replication is described in XML format, and may contain multiple rules. The following table describes the configuration elements.

<table>
<thead>
<tr>
<th>Table 11-1 Bucket cross-region replication configuration elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>ReplicationConfiguration</td>
</tr>
<tr>
<td>Agency</td>
</tr>
<tr>
<td>Rule</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Prefix</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Destination</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Bucket</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>StorageClass</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

For example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
  <Agency>testAcy</Agency>
  <Rule>
    <ID>rule1</ID>
    <Status>Enabled</Status>
    <Prefix/>
    <Destination/>
  </Rule>
</ReplicationConfiguration>
```
After bucket cross-region replication is configured, qualified objects in the source bucket are automatically copied to the destination bucket in the destination region.

**Fee Description**

The charge of cross-region replication consists of request fee and traffic transmission fee from the source region to the destination region.

For the charging standard, refer to the product pricing details of the source region. KMS encryption fee is charged according to the KMS pricing details. The owner of the source bucket is responsible for the cost.

**Example:**

Copy a 1 GB object from the source region to the destination region. The charging is as follows:

1 GB * Cross-region replication traffic fee in the source region + 1 * Single request fee in the source region

Upload 100 parts with each part 50 MB to the source region, and copy them after merging to the destination region. The charging is as follows:

100 * 50 MB/1024 MB * Copy traffic fee in the source region + 102 * Single request fee in the source region
12 Event notification

The notification function provided by OBS enables you to receive notifications when certain events occur in your bucket. To enable notifications, you must first add a notification configuration to identify the events that you want OBS to publish and the destinations to which you want OBS to send the event notifications.

Event Types Supported by OBS

OBS can publish events of the following types. You need to specify these event types in the notification configuration.

Table 12-1 Event types supported by OBS

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectCreated:* (all uploading operations)</td>
<td>OBS can use APIs such as PUT, POST, and COPY to create objects and configure corresponding event types. You will receive a notification when an object is created using a specific API. In addition, you can use the s3:ObjectCreated:* event type to request all object creation notifications.</td>
</tr>
<tr>
<td>ObjectCreated:Put (uploading objects)</td>
<td></td>
</tr>
<tr>
<td>ObjectCreated:Post (uploading objects using a browser)</td>
<td></td>
</tr>
<tr>
<td>ObjectCreated:Copy (copying objects)</td>
<td></td>
</tr>
<tr>
<td>ObjectCreated:CompleteMultipartUpload (complete multipart upload)</td>
<td></td>
</tr>
</tbody>
</table>

NOTE

You will not receive event notifications from failed operations.
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectRemoved:* (all deletion operations)</td>
<td>By using the <strong>ObjectRemoved</strong> event types, you can enable notification when an object or a batch of objects are removed from a bucket. You can request notification when an object is deleted or a versioned object is permanently deleted by using the <strong>ObjectRemoved:Delete</strong> event type. Alternatively, you can request notification when a delete marker is created for a versioned object by using <strong>ObjectRemoved:DeleteMarkerCreated</strong>. You can also use <strong>ObjectRemoved:</strong>* to request notification each time an object is deleted. <strong>NOTE</strong> You will not receive event notifications from automatic deletions of lifecycle policies or from failed operations.</td>
</tr>
<tr>
<td>ObjectRemoved:Delete (deleting objects)</td>
<td></td>
</tr>
<tr>
<td>ObjectRemoved:DeleteMarkerCreated (A deletemarker object is created.)</td>
<td></td>
</tr>
</tbody>
</table>

### Destinations Supported by OBS

OBS can send event notification messages to the following destinations. You must grant the OBS permissions to send messages to these destinations. In addition, you need to specify the URN values of these destinations in the notification configuration.

- **Simple-Message-Notification (SMN) Topic**

  **NOTE**

  SMN is a type of web service that a user can easily construct and maintain. SMN sends notifications from a cloud. SMN employs the publish-subscribe (pub-sub) message sending mode. Publishers send messages to the topics where they are authorized to publish messages. SMN takes the initiative to push messages to topic subscribers (such as email and SMS addresses).

### How to Enable Event Notifications

Enabling notifications is a bucket-level operation. OBS stores your event notification configuration to bucket sub-resources in the format of XML. By default, notification is not enabled for any event type. That is, the initial event notification configuration of each bucket is empty.

To enable notifications for events of specific types, you must add the corresponding XML configuration that identifies the event types you want OBS to publish and the destination where you want the notifications to be published, for example,
publishing event messages to an SMN topic — To set an SMN topic as the notification destination for specific event types, add the **TopicConfiguration**.

```xml
<NotificationConfiguration>
    <TopicConfiguration>
        <Id>optional-id-string</Id>
        <Topic>topic-urn</Topic>
        <Event>event-type</Event>
        ... 
    </TopicConfiguration>
    ... 
</NotificationConfiguration>
```

To remove all notification configurations from a bucket, set the `<NotificationConfiguration>` element to null.

**Using Object Key Names to Filter Event Notifications**

You can set event notification by the prefix and suffix of an object key name. For example, you can set up a configuration so that a notification is published only when objects with a .jpg extension are added to a bucket.

OBS stores the notification configuration as XML. You can use the Filter element in the XML structure to define the rules for notifications to be filtered by the prefix and/or suffix of an object key name. Notification configurations that use the filter cannot define filtering rules with overlapping prefixes, overlapping suffixes, or prefix and suffix overlapping. The following are examples of notification configurations with object key name filtering:

- Example of valid notification configuration with object key name filtering
  The following information contains a configuration identifying an SMN topic for OBS to publish events of the **ObjectCreated:Put** type to. The events will be published each time an object that has an **image** prefix and a **jpg** suffix is PUT to a bucket.

```xml
<NotificationConfiguration>
    <TopicConfiguration>
        <Filter>
            <Object>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>image</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>jpg</Value>
                </FilterRule>
            </Object>
        </Filter>
        <Topic>urn:smn:southchina:11aa22bb:notification</Topic>
        <Event>ObjectCreated:Put</Event>
    </TopicConfiguration>
</NotificationConfiguration>
```

The following notification configuration has multiple non-overlapping prefixes. The configuration defines the following: When objects that have a prefix of **images** are uploaded to buckets, event notifications will be published to topic-A; when objects that have a prefix of **videos** are uploaded to buckets, event notifications will be published to topic-B.

```xml
<NotificationConfiguration>
    <TopicConfiguration>
        <Filter>
            <Object>
                ... 
            </Object>
        </Filter>
        ... 
</NotificationConfiguration>
```
The following notification configuration has multiple non-overlapping suffixes. The configuration defines the following: Notifications will be published to topic-A for all .jpg objects PUT to buckets and notifications will be published to topic-B for all .png objects. The suffixes .png and .jpg are not overlapping even though they have the same last letter.

- Examples of notification configuration with invalid prefix/suffix overlapping

In most cases, notification configurations that use the filter cannot define filtering rules with overlapping prefixes, overlapping suffixes, or overlapping combinations of prefixes and suffixes for the same event types. (You can have overlapping prefixes if the suffixes do not overlap.) You can use overlapping object key name filtering rules with different event types. For example, you
can create a notification configuration that uses the `image` prefix for the `ObjectCreated:Put` event type and the `image` prefix for the `ObjectDeleted:*` event type.

Configuration information that does not contain the filter attribute matches all prefixes and suffixes by default. The following notification configuration is invalid because it contains overlapping prefixes. (The same thing would be true if `suffix` instead of `prefix` is used in this example.)

```xml
<ApplicationConfiguration>
    <TopicConfiguration>
        <Topic>urn:smn:southchina:11aa22bb:topic-A</Topic>
        <Event>ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
        <Filter>
            <Object>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>abc</Value>
                </FilterRule>
            </Object>
        </Filter>
        <Topic>urn:smn:southchina:11aa22bb:topic-B</Topic>
        <Event>ObjectCreated:*</Event>
    </TopicConfiguration>
</ApplicationConfiguration>
```

The following notification configuration is invalid because it contains overlapping suffixes. Two suffixes are considered overlapping if a given string can end with both suffixes. A string can end with `jpg` and `pg`. Therefore, the suffixes are overlapping. (The same is true for prefixes.)

```xml
<ApplicationConfiguration>
    <TopicConfiguration>
        <Filter>
            <Object>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>jpg</Value>
                </FilterRule>
            </Object>
        </Filter>
        <Topic>urn:smn:southchina:11aa22bb:topic-A</Topic>
        <Event>ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
        <Filter>
            <Object>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>pg</Value>
                </FilterRule>
            </Object>
        </Filter>
        <Topic>urn:smn:southchina:11aa22bb:topic-B</Topic>
        <Event>ObjectCreated:Put</Event>
    </TopicConfiguration>
</ApplicationConfiguration>
```

**Event Message Structure**

A notification message sent by OBS to publish an event is a JSON message with the following structure. Note the following:

- If you want to trace the request by following up with the OBS support, the `responseElements` key value is useful. Both `x-obs-request-id` and `x-obs-id-2`
help OBS to trace the individual request. These values are the same as those returned by the OBS in response to your original request.

- The key values provide information about the bucket and object involved in the event. The object key name value is URL-encoded, for example, the key name `TEST/Chinese .jpg` is encoded to `TEST%2F%E4%B8%AD+%E6%96%87%2F.jpg`.

---

**NOTICE**

During secondary development, if you want to download this object using SDK, you need to decode the object key name from the URL-encoded format, and then call the API for downloading. If you use a browser to access the object, decoding is not required.

- The sequencer key provides a way to determine the sequence of events. Event notifications are not guaranteed to arrive in the sequence that the events occurred. However, notifications from events that create objects (PUTs) and delete objects contain a sequencer, which can be used to determine the sequence of events for a given object key. If you compare the hexadecimal sequencer strings from two event notifications on the same object key, you will find that the event notification with the greater sequencer value is the later event.

---

**NOTICE**

1. The sequencer cannot be used to determine the sequence for events on different object keys.
2. The value of sequencer is for reference only, and cannot be used in highly reliable systems.

---

**Example message:**

- **Test message** — When you configure an event notification on a bucket, OBS sends the following test message:

  ```
  
  
  
  ```

- **Example message when an object is created using the PUT request** — The following is an example of a message sent by OBS to publish an `ObjectCreated:Put` event:

  ```
  
  ```

---

Copyright © Huawei Technologies Co., Ltd.
Due to Internet issues or constraints on e-mail sending, notifications may fail to send to HTTP or e-mail terminals.
Bucket access logs are for bucket owners to obtain access data of buckets. After the logging function is enabled, a log is generated for each operation on a bucket and multiple logs are packed into a log file. Requester information and request parameters are saved in bucket access logs and are saved into destination buckets as objects to provide statistics to the bucket owner.

When enabling the logging function, you need to specify a location where log files are stored. They can be stored in the bucket for which you are enabling the logging, or in other buckets that you have the required permissions. However, the bucket where log files are stored and the bucket for which the logging is enabled must be in the same region.

The log files are generated by OBS and uploaded to the bucket where the logs are stored. Therefore, OBS needs to be authorized to upload the generated log files. Before configuring bucket log management, you need to generate a delegate for OBS service on IAM and configure the delegate name as a parameter on the bucket.

Buckets in the Infrequent Access and Archive storage classes cannot be used to store log file. Stored log files occupy storage space in a bucket. Therefore, users are charged for the logging service based on the pricing for data storage.

Format of Generated Logs

After the bucket log function is enabled, a log is generated upon each access to OBS. The format of the bucket access log files generated in the destination bucket is as follows:

```
3ebc47240f7640e4b6c4b33937dd9b58 hw203 [15/Jun/2015:06:38:27 +0000] 1192.168.0.1
3ebc47240f7640e4b6c4b33937dd9b58 A196564DC12F13AAA19FD56D23B04C58 REST.GET.BUCKET - "GET /hw203 HTTP/1.1" 200 - 247 - 58 - 58 "curl/7.19.0 (x86_64-suse-linux-gnu) libcurl/7.19.0 OpenSSL/0.9.8 zlib/1.2.3 libidn/1.10" – –
3ebc47240f7640e4b6c4b33937dd9b58 hw203 [15/Jun/2015:06:38:28 +0000] 1192.168.0.1
3ebc47240f7640e4b6c4b33937dd9b58 2CFE324941EE96DDDD41F5FE57B369DD REST.GET.BUCKET - "GET /hw203 HTTP/1.1" 200 - 247 - 39 - 39 "curl/7.19.0 (x86_64-suse-linux-gnu) libcurl/7.19.0 OpenSSL/0.9.8 zlib/1.2.3 libidn/1.10" – –
3ebc47240f7640e4b6c4b33937dd9b58 hw203 [15/Jun/2015:06:38:29 +0000] 1192.168.0.1
3ebc47240f7640e4b6c4b33937dd9b58 175189E1217733A1F41F4060DF2E7268 REST.GET.BUCKET - "GET /hw203 HTTP/1.1" 200 - 247 - 22 - 22 "curl/7.19.0 (x86_64-suse-linux-gnu) libcurl/7.19.0 OpenSSL/0.9.8 zlib/1.2.3 libidn/1.10" – –
3ebc47240f7640e4b6c4b33937dd9b58 hw203 [15/Jun/2015:06:38:30 +0000] 1192.168.0.1
3ebc47240f7640e4b6c4b33937dd9b58 0F1A4E95C42D7A491D8868CD040D2EB REST.GET.BUCKET - "GET /
```
The access log of each bucket contains the information listed in the following table:

### Table 13-1 Bucket logging format

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BucketOwner</td>
<td>3ebc47240f7640e4b6c4b33937dd9b58</td>
<td>ID of the bucket owner</td>
</tr>
<tr>
<td>Bucket</td>
<td>hw203</td>
<td>Bucket name</td>
</tr>
<tr>
<td>Time</td>
<td>[15/Jun/2015:06:38:27 +0000]</td>
<td>Request timestamp</td>
</tr>
<tr>
<td>RemoteIP</td>
<td>192.168.0.1</td>
<td>Request IP address</td>
</tr>
<tr>
<td>Requester</td>
<td>3ebc47240f7640e4b6c4b33937dd9b58</td>
<td>Requester account ID</td>
</tr>
<tr>
<td>RequestID</td>
<td>175189E1217733A1F41F4060DF2E7268</td>
<td>Request ID</td>
</tr>
<tr>
<td>Operation</td>
<td>REST.GET.BUCKET</td>
<td>Operation</td>
</tr>
<tr>
<td>Key</td>
<td>phptoto.gif</td>
<td>Object name</td>
</tr>
<tr>
<td>Request-URI</td>
<td>&quot;GET /hw203 HTTP/1.1&quot;</td>
<td>Request URI</td>
</tr>
<tr>
<td>HTTPStatus</td>
<td>200</td>
<td>Response code</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>NoSuchBucket</td>
<td>Error code</td>
</tr>
<tr>
<td>BytesSent</td>
<td>555</td>
<td>Size of the HTTP response, expressed in bytes</td>
</tr>
<tr>
<td>ObjectSize</td>
<td>400</td>
<td>Object size</td>
</tr>
<tr>
<td>TotalTime</td>
<td>58</td>
<td>Server processing time (millisecond)</td>
</tr>
<tr>
<td>Turn-AroundTime</td>
<td>58</td>
<td>Total request time (millisecond)</td>
</tr>
<tr>
<td>Referrer</td>
<td>&quot;<a href="http://example.com">http://example.com</a>&quot;</td>
<td>Request referrer header</td>
</tr>
<tr>
<td>User-Agent</td>
<td>&quot;curl/7.19.0&quot;</td>
<td>Request user-agent header</td>
</tr>
<tr>
<td>VersionID</td>
<td>25HXDUX78WEUU367</td>
<td>Version ID contained in a request</td>
</tr>
<tr>
<td>Parameter</td>
<td>Example</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>STSLogUrn</td>
<td>-</td>
<td>Federated authentication and agency information</td>
</tr>
</tbody>
</table>

**Setting Logging for a Bucket**

The setting is as follows in XML format:

```xml
<BucketLoggingStatus>
  <Agency>agency-name</Agency>
  <LoggingEnabled>
    <TargetBucket>mybucketlogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee>
          <ID>domainID</ID>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```
14 Monitoring

14.1 Related Concepts

14.2 Viewing OBS Metrics

14.1 Related Concepts

This section describes how to view OBS metrics, and only involves a basic introduction of the monitoring metrics and rollup of the Cloud Eye service. For details, see the Cloud Eye service.

Monitoring Metrics

As the core concept of cloud monitoring, a metric is used to measure resource performance of a specific dimension. For example, upload traffic of OBS, number of GET requests, and number of 4xx exceptions are all metrics. A metric is a time-dependent variable that generates a series of monitoring data over time. It helps you understand the metric changes over a specified period of time. Table 14-1 details the OBS metrics as follows:

Table 14-1 OBS metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Value</th>
<th>Monitored Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes downloaded</td>
<td>Specifies the response bytes of all download requests made to all buckets in a region, including bytes in HTTP bodies. Unit: byte</td>
<td>≥ 0 Byte</td>
<td>OBS</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Value</td>
<td>Monitored Object</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>Bytes uploaded</td>
<td>Specifies the bytes in the message bodies of all upload requests made to all buckets in a region, including bytes in HTTP bodies. Unit: byte</td>
<td>≥ 0 Byte</td>
<td>OBS</td>
</tr>
<tr>
<td>GET requests</td>
<td>Specifies the number of GET, HEAD, or OPTIONS requests made to all buckets and objects in the buckets of a region. Unit: count</td>
<td>≥ 0 Counts</td>
<td>OBS</td>
</tr>
<tr>
<td>PUT requests</td>
<td>Specifies the number of PUT, POST, or DELETE requests made to all buckets and objects in the buckets of a region. Unit: count</td>
<td>≥ 0 Counts</td>
<td>OBS</td>
</tr>
<tr>
<td>First byte download delay</td>
<td>Specifies the average time that a system receives a GET, HEAD, or OPTIONS request to the time that the system starts to respond to the request in a measurement period.</td>
<td>≥ 0 ms</td>
<td>OBS</td>
</tr>
<tr>
<td>4xx errors</td>
<td>Specifies the number of requests whose error code is 4xx returned by the server. Unit: count</td>
<td>≥ 0 Counts</td>
<td>OBS</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td>Value</td>
<td>Monitored Object</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td>5xx errors</td>
<td>Specifies the number of requests whose error code is 5xx returned by the server. Unit: count</td>
<td>≥ 0 Counts</td>
<td>OBS</td>
</tr>
</tbody>
</table>

**Rollup**

Rollup is the process in which Cloud Eye calculates the average, maximum, minimum, sum, or variance value based on sample raw data reported by each cloud service in specific periods. The calculation period is called rollup period. Currently, OBS supports five rollup periods: 5 minutes, 20 minutes, 1 hour, 4 hours, and 24 hours. OBS rollup allows you to collect and analyze statistics about the number of operations on buckets and objects in buckets in a specified period.

### 14.2 Viewing OBS Metrics

#### 14.2.1 Operation Method and Procedure

You can view the OBS metrics on the **Cloud Service Monitoring** page of Cloud Eye, which allows you to query the running status of OBS and all metrics in real time.

**Procedure**

1. **Step 1** Log in to the management console. Log in to the system as prompted.
Step 2  Select a region and project (Optional). After logging in, you can click in the upper left of the management console to select a region and project. Alternatively, you can perform the operation or modify your selection after accessing the OBS metrics page to view the monitoring metrics of OBS service in different regions.

Step 3  Go to the Cloud Eye page.

In the upper left of the management console, click Service List, and choose Management & Deployment > Cloud Eye.
Step 4  On the Cloud Eye page that is displayed, choose Cloud Service Monitoring > Object Storage Service.

Step 5  Go to the monitoring metrics page. Click View Graph to view the monitoring data.
14.2.2 Viewing Monitoring Graphs

On the monitoring graphs page that is displayed, you can perform the following operations to view monitoring data:

Modifying the Displayed Monitoring Metrics

You can specify a metric in the search box.

**Figure 14-6** Searching for a metric graph

Or you can click **Configure Metric** to select the metric graph to be displayed.

**Figure 14-7** Configure Metric
Selecting the Monitoring Period

On the monitoring graphs page that is displayed, the metric graphs of the last one hour is displayed by default (Figure 14-5). You can configure the period to the last three hours or last 12 hours.
Refreshing the Monitoring Data

You can enable the automatic refresh function or manually click the refresh button to refresh the monitoring data so that you can obtain the latest monitoring graphs.

Displaying the Detailed Data of a Metric at a Specific Time

When you move the cursor to a time point on curve of a monitoring graph, the detailed data of the time point is displayed.
14.2.3 Viewing the Detailed Data of Each Metric

*Table 14-1* details the OBS metrics. On the monitoring graphs page that is displayed, you can view the corresponding metrics. Click next to a metric to view more detailed data of the metric.

**Bytes Downloaded**

The metric specifies the response bytes of all download requests made to all buckets in a region, including bytes in HTTP bodies. Bytes downloaded is measured in the unit of byte. The following figure shows the monitoring graph of the metric.

**Figure 14-13** Metric graph of traffic downloaded

You can configure a period of the metric, its rollup data and unit to be displayed in a monitoring graph.

- **Configuring a period of the metric to be displayed on the graph**
  You can view the metric graphs of a site in last 1 hour, last 3 hours, last 12 hours, last 24 hours, last 7 days, and last 30 days.
You can also customize a period to view the monitoring graph.

**Figure 14-15** Metric graph of bytes downloaded (2018/07/02 16:43:27 – 2018/07/12 16:43:27)

- **Selecting a rollup group to be displayed**
  Click *Settings*, choose *Rollup*, and select a period and method to view the specific monitoring graph of the rollup group.
**Figure 14-16 Selecting a rollup group**

![Selecting a rollup group](image)

- **Selecting a unit for a metric graph to be displayed**
  You can click the unit drop-down list box to set the unit of the monitoring curve to Byte, KB, MB, or GB.

**Figure 14-17 Selecting a unit for a metric graph to be displayed**

![Selecting a unit for a metric graph](image)

**Bytes Uploaded**

The metric specifies the bytes in the message bodies of all upload requests made to all buckets in a region, including bytes in HTTP bodies. Bytes uploaded is measured in the unit of byte. The following figure shows the metric graph of bytes uploaded for a longer time. In the figure, the period is set to June 1, 2018 to July 12, 2018; for the rollup group, the period is 4 hours and the method is Sum; the unit is MB.
You can configure a period of the metric, its rollup data and unit to be displayed in a monitoring graph. The operation is the same as that of **Bytes Downloaded**.

**GET Requests**

This metric specifies the number of GET, HEAD, or OPTIONS requests made to all buckets and objects in the buckets of a region. It is measured in count. The following figure shows the metric graph of GET requests for a longer time. In the figure, the period is set to last 3 hours, and for the rollup group, the period is 20 minutes and the method is Variance.

![GET Requests](image)

You can configure a period of the metric and its rollup data to be displayed. The operation is the same as that of **Bytes Downloaded**.
First Byte Download Delay

This metric specifies the average time that a system receives a GET, HEAD, or OPTIONS request to the time that the system starts to respond to the request in a measurement period. The following figure shows the metric graph of the first byte download delay. In the figure last seven days, the period is set to last 7 days, and for the rollup group, the period is 1 hour and the method is Average.

Figure 14-20 Metric graph of the time to first byte (TTFB)

You can configure a period of the metric and its rollup data to be displayed. The operation is the same as that of Bytes Downloaded.

PUT requests

This metric specifies the number of PUT, POST, or DELETE requests made to all buckets and objects in the buckets of a region. It is measured in count. The following figure shows the metric graph of PUT requests for a longer time. In the figure, the period is set to last 30 days, and for the rollup group, the period is 4 hours and the method is Sum.
You can configure a period of the metric and its rollup data to be displayed. The operation is the same as that of **Bytes Downloaded**.

### 4xx Errors

This metric specifies the number of requests whose error code is 4xx returned by the server. It is measured in count. The following figure shows the metric graph of 4xx errors for a longer time. In the figure, the period is set to last 7 days, and for the rollup group, the period is 1 hour and the method is Sum.

![Metric graph of 4xx errors](image-url)

You can configure a period of the metric and its rollup data to be displayed. The operation is the same as that of **Bytes Downloaded**.
5xx Errors

This metric specifies the number of requests whose error code is 5xx returned by the server. It is measured in count. The following figure shows the metric graph of 5xx errors for a longer time. In the figure, the period is set to Last 30 days, and for the rollup group, the period is 24 hours and the method is Average.

Figure 14-23 Metric graph of 5xx errors

You can configure a period of the metric and its rollup data to be displayed. The operation is the same as that of Bytes Downloaded.
Overview

OBS provides bucket inventories to facilitate your management of objects in a bucket. You can configure bucket inventories to periodically list objects in a bucket. During the listing of objects, object metadata is saved in a CSV file, which is uploaded to the specified bucket.

When configuring a bucket inventory, you can specify the filter criteria (by object name prefix) of objects, set the inventory generation interval (daily or weekly), and choose whether to list all versions of objects. In addition, you can also specify the content of object metadata to be listed by the inventory, including the object size, modification time, and storage class.

You can configure multiple inventories for a bucket, but the object name prefixes specified in these inventories cannot overlap. For example, if an inventory has been configured to filter objects by the name prefix of 'a', the object name prefix 'ab' cannot be used as the filter criteria of another inventory for the bucket.

Use Restrictions

- Only OBS 3.0 buckets support this function. However, this restriction does not apply to the destination bucket where inventory files are saved.
- The source bucket for which an inventory is configured and the destination bucket where inventory files are saved must belong to the same tenant and reside in the same region.
- Inventory files are in CSV format.
- If a bucket has enabled the KMS encryption function, it cannot be used as the destination bucket for saving inventory files.
- Inventory files are delivered to the destination bucket by an OBS system user. Therefore, you need to authorize the system user the permission to write the destination bucket.

How to Configure a Bucket Inventory

Before the configuration, you need to briefly understand what is a source bucket and what is a destination bucket.

- Source bucket: A source bucket is the bucket for which an inventory is configured. The inventory lists objects stored in the source bucket.
• Destination bucket: A destination bucket is where generated inventory files are stored. A source bucket can also be the destination bucket. You can specify a name prefix for an inventory. Then generated inventory files will be named with the prefix and saved in the directory with the prefix. If you do not specify any name prefix for the inventory, the generated inventory files are stored in the root directory of the bucket.

  - Restrictions on the destination bucket
    ▪ The destination bucket and source bucket must belong to the same tenant.
    ▪ The destination bucket and source bucket must be in the same region.
    ▪ The policy of the destination bucket must grant the OBS system users the permission to write objects to the bucket. For details about how to authorize such permission, see 1.
  - The destination bucket contains the following files:
    ▪ A list of inventory files
    ▪ The Manifest file, which contains the list of all inventory files under a certain inventory configuration. For details about the Manifest file, see Manifest File.

Configuring a Bucket Inventory
You use OBS Console or call the API to configure a bucket inventory. If you configure a bucket inventory on OBS Console, a bucket policy with the required permission configuration is automatically generated for the destination bucket. If you call the API to configure the bucket inventory, you need to manually configure the bucket policy for the destination bucket.

1. Add a bucket policy for the destination bucket.
   A bucket policy must be configured for the destination bucket, to grant the OBS system users the permission to write objects to the destination bucket. The format of the bucket policy is as follows. Replace destbucket with the actual name of the destination bucket.

   ```
   ```

2. Configure a bucket inventory.
   For details about how to configure a bucket list by calling the API, see descriptions about configuring bucket inventories in the OBS API Reference.

Content in an Inventory File
The content in an inventory file can be configured when creating the inventory. For details about all possible fields, see Table 15-1.
<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket</td>
<td>Name of the source bucket</td>
</tr>
<tr>
<td>Key</td>
<td>The name of an object. Each object in a bucket has a unique key. (Object names in the inventory file are URL-encoded using UTF-8 character set and can be used only after being decoded.)</td>
</tr>
<tr>
<td>VersionId</td>
<td>Version ID of an object. If the value of IncludedObjectVersions in the inventory configuration is Current, this field is not included in the inventory file.</td>
</tr>
<tr>
<td>IsLatest</td>
<td>If the object version is the latest, this parameter is True. (If the value of IncludedObjectVersions in the inventory configuration is Current, this field is not included in the inventory file.)</td>
</tr>
<tr>
<td>IsDeleteMarker</td>
<td>When versioning is enabled for the source bucket, if an object is deleted, a new object metadata is generated for the object, and the IsDeleteMarker of the metadata is set to true. (If the value of IncludedObjectVersions in the inventory configuration is Current, this field is not included in the inventory file.)</td>
</tr>
<tr>
<td>Size</td>
<td>Object size, in bytes.</td>
</tr>
<tr>
<td>LastModifiedDate</td>
<td>Object creation date or last modification date.</td>
</tr>
<tr>
<td>ETag</td>
<td>128-bit MD5 digest of the Base64 code of an object. ETag is the unique identifier of the object content. It can be used to identify whether the object content is changed. For example, if ETag value is A when an object is uploaded and the ETag value has changed to B when the object is downloaded, it indicates that the object content is changed.</td>
</tr>
<tr>
<td>StorageClass</td>
<td>Storage class of an object.</td>
</tr>
<tr>
<td>IsMultipartUploa-</td>
<td>Indicates whether an object is uploaded in the multipart mode.</td>
</tr>
<tr>
<td>ReplicationStatus</td>
<td>Cross-region replication status of an object</td>
</tr>
<tr>
<td>EncryptionStatus</td>
<td>Encryption status of an object</td>
</tr>
</tbody>
</table>

### Inventory File Name

The name of an inventory file is in the following format:

```plaintext
destinationPrefix/sourceBucketName/inventoryId/yyyy-MM-dd'T'HH-mm'Z'/files/UUID_index.csv
```

- **destinationPrefix**: The inventory file name prefix configured when creating the inventory rule. Inventory files generated under the rule are named after...
the prefix, which can facilitate the classification of inventory files. If no prefix is specified, the default prefix is **BucketInventory**.

- **sourceBucketName**: Name of the source bucket for which an inventory is configured. This field can be used to differentiate inventory files of different source buckets, if those inventory files are saved in the same destination bucket.

- **inventoryId**: If a source bucket has multiple inventory rules whose inventory files are saved in the same destination bucket, this field can be used to identify different inventory rules.

- **yyyy-MM-dd'T'HH-mm'Z'**: Start time and date for scanning the destination bucket when an inventory file is generated. Objects uploaded to the source bucket after this time may not be listed in the inventory file.

- **UUID_index.csv**: One of the inventory files.

### Manifest File

If a source bucket contains a large number of objects, multiple inventory files may be generated under one inventory rule. After complete inventory files are generated, a **manifest.json** file is generated, summarizing information about all the generated inventory files. See information details as follows:

- **sourceBucket**: name of the source bucket
- **destinationBucket**: name of the destination bucket
- **version**: version of the inventory
- **fileFormat**: format of inventory files
- **fileSchema**: object metadata fields contained in the inventory files
- **files**: list of all inventory files
- **key**: inventory file name
- **size**: size of an inventory file, in bytes
- **inventoriedRecord**: number of records contained in an inventory file

The following is an example of a simple **manifest.json** file.

```json
{
  "sourceBucket":"user001",
  "destinationBucket":"bucket001",
  "version":"2019-01-03",
  "fileFormat":"CSV",
  "fileSchema": ",Bucket,Key,Size,LastModifiedDate,ETag,StorageClass,IsMultipartUploaded,ReplicationStatus,EncryptionStatus",
  "files": [
    {
      "key": "inventory/user001/test_id/2019-01-03T12-28Z/files/0000016B13AF58E66B06C1E2D7F15155_1.csv",
      "size": 6705647390,
      "inventoriedRecord": 70585762
    }
  ]
}
```

The name of a **manifest** file is as follows. The meanings of the fields are as follows:

```
destinationPrefix/sourceBucketName/inventoryId/inventoryId/yyyy-MM-dd'T'HH-mm'Z'/manifest.json
```
How Is a User Notified When the Generation of Inventory Files Complete?

You can enable SMN (the message notification service) for the destination bucket. By doing so, you can receive SMS messages or emails every time when inventory files and the manifest file are generated. For more information about SMN, see 12 Event notification.

The following is a simple example of SMN configuration. destinationPrefix/sourceBucketName indicates the prefix of the manifest file. destinationPrefix is the configured name prefix for inventory files, and sourceBucketName is the source bucket for which the inventory file is configured. The manifest.json is the suffix of the manifest file.

```
<NotificationConfiguration>
  <TopicConfiguration>
    <Id>01</Id>
    <Filter>
      <Object>
        <FilterRule>
          <Name>prefix</Name>
          <Value>destination-prefix/source-bucket</Value>
        </FilterRule>
        <FilterRule>
          <Name>suffix</Name>
          <Value>manifest.json</Value>
        </FilterRule>
      </Object>
    </Filter>
    <Topic>urn:smn:southchina:11aa22bb:notification</Topic>
    <Event>ObjectCreated:Put</Event>
  </TopicConfiguration>
</NotificationConfiguration>
```
16 Common Operations

16.1 Common Fault Locating

During the OBS running, a connection exception "Connection Timeout" occurs.

1. The network connection timed out due to a three-way handshake failure occurred during the TCP connection establishment. In this scenario, rectify the fault of the network between the client and OBS.
   a. On the client for accessing OBS, check whether the service address and port of OBS are correctly configured.
   b. Check whether the OBS network is well connected by using a curl or wget command.
2. OBS detects that the signatures do not match and returns error SignatureDoesNotMatch.
   a. Check whether the OBS domain name configured on the client is consistent with the OBS domain name in actual use.
   b. Check whether the AK/SK configuration on the client is correct.

OBS Returns a 403 Signature Error

After receiving an HTTP/HTTPS request, OBS obtains the specified headers or the query parameters using the signature calculation method and constructs StringToSign, and then calculates the signature and compares it with the client signature to check whether they are identical.

The server returns a 403 error if:

1. The headers used by the client for calculating signatures differ from those in the HTTP message. For example, the value of Date used by the client is the time for signature calculation and that used by OBS is the request time.
2. The signature calculation rules of the client and those of OBS are different. When OBS prompts a signature error, the StringToSign used by OBS for
signature calculation will be contained in the message body and returned to the client.

The following is an example error. After receiving a request and detecting the signatures unmatched, OBS returns the `StringToSign` used by the client for signature calculation as the `<StringToSign>` element. The client uses the OBS-returned information and reviews the calculation process to locate the fault.

```xml
<Error>
  <Code>SignatureDoesNotMatch</Code>
  <Message>The request signature we calculated does not match the signature you provided. Check your key and signing method.</Message>
  <RequestId>BF26000001644584C8706D0A72B15B3B</RequestId>
  <HostId>tOumRKZcscaF+jDHRvDTAHHehSWiCapYoIkJ8NuibeelINQsu6qtPps42CCIpmJSG</HostId>
  <AccessKeyId>UDSIAMSTUBTEST000003</AccessKeyId>
  <SignatureProvided>amgvrLdL0N1XKBUAiWodhGkbpov==</SignatureProvided>
  <StringToSign>GET
Thu, 28 Jun 2018 08:31:27 GMT
/bucket02/?cors</StringToSign>
  <StringToSignBytes>47 45 54 0a 0a 0a 54 68 75 2c 20 32 38 20 4a 75 6e 20 32 30 31 38 20 30 38 3a 33
31 3a 32 37 20 2b 30 30 30 30 0a 2f 62 75 63 6b 65 74 30 32 2f 3f 63 6f 72 73</StringToSignBytes>
</Error>

Incorrect MD5 Value of the Object to Be Uploaded

In APIs for uploading an object, Content-MD5 header can carry the MD5 value of the object to be uploaded. After receiving the request, the server checks whether the MD5 value of the received object is consistent with that of the client. If no, a data error occurs, and the server rejects the upload and the error message BadDigest is displayed.

MD5 is the Base64-encoded 128-bit MD5 digest of the HTTP message according to RFC 1864.

For example, if the preceding content is `1234567890`, the calculation method for the Content-Md5 of the character string is as follows:

Calculate the binary array (128 bits) encrypted by MD5, and perform Base64 coding on this binary (rather than encoding the 32-bit character string).

```python
import base64, hashlib
hash = hashlib.md5()
hash.update("1234567890")
base64.b64encode(hash.digest())
```

The result is `6Afx/PgtEy+bsBjkKZzihnw==`

Server Returns a 5xx Error

To facilitate fault locating and handling, OBS creates a unique requestId for each request and returns the requestId to the client as the value of header x-obs-request-id. If an error occurs, print this header and the message body which may help you for troubleshooting.
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>What's New</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-07-16</td>
<td>This issue is the fourth official release. Updated the following content: • Added the bucket inventory related content.</td>
</tr>
<tr>
<td>2019-06-12</td>
<td>This issue is the third official release. Updated the following content: • Added the description of accessing OBS through temporary authentication.</td>
</tr>
<tr>
<td>2019-02-03</td>
<td>This issue is the second official release. Updated the following content: • Resolved some ease-of-use issues.</td>
</tr>
<tr>
<td>2018-12-14</td>
<td>This issue is the first official release.</td>
</tr>
</tbody>
</table>