Object Storage Service

User Guide

Issue 06

Date 2023-05-05

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Before You Start

Object Storage Service (OBS) provides a series of documents to help you better understand and use the service.

Before using OBS, read this document and other related help documents (see **Table 1-1**) to understand the basic concepts, application scenarios, and operations of OBS, so you can quickly get started with OBS.

Table 1-1 OBS documents

Document	Description
Progressive Knowledge	Provides links to *Quick Start*, *User Guide*, *Best Practices*, *FAQs*, and Solutions, so you can easily and accurately locate the content you want to view.
What's New	Describes the latest updates of OBS, including new functions and documents.
Function Overview	An overview introduction to OBS functions and regions where these functions are available.
Service Overview	Basic concepts, advantages, application scenarios, billing, and permission management for OBS.
Getting Started	A quick guide to help you start with basic operations, such as creating buckets, uploading objects, and downloading objects.
Console Operation Guide	Guides you through OBS console operations with elaborated use examples.
Tools Guide	Introduces OBS tools, including OBS Browser+, obsutil, and obsfs, with use examples provided.
Image Processing Feature Guide	An operation guide to the image processing of OBS. It covers basic concepts, functions, and FAQs, and demonstrates how to use this feature through OBS Console or by calling OBS APIs.

Document	Description
Parallel File System Feature Guide	Introduces the basic concepts, application scenarios, limitations, user operations, and billing for the OBS Parallel File System (PFS).
Best Practices	A collection of OBS best practices for typical scenarios, helping you achieve your business goals with better performance, lower costs, and more convenient operations.
API Reference	Lists the REST APIs provided by OBS, including request samples, response samples, and parameter descriptions. It helps you use APIs to conduct secondary development.
SDK Reference	Describes the OBS Software Development Kits (SDKs) in mainstream programming languages such as Java, Python, C, GO, Android, and iOS. SDKs provide parameter details, examples, and FAQs to help you smoothly conduct installation and development.
FAQs	Offers you the answers to frequently asked questions about OBS.
Videos	Videos tutorials about OBS basic functions are provided for your reference.

2 Using OBS

You can manage OBS resources in the following ways:

Tool	Description	How to Use	Reference
OBS Console	OBS Console is a web-based GUI. You can manage all your OBS resources through this console.	You need an account (including the password) or an IAM user to log in to OBS Console. For details, see **Logging In to OBS Console**.	**Console Operation Guide**
SDKs	OBS SDKs encapsulate APIs provided by OBS to simplify development. Users can use API functions provided by the OBS SDKs to access OBS service capabilities.	Configure the endpoint during environment preparation and set the access keys (AK/SK) during initialization. For details, see **SDK Reference**.	**SDK Reference**
API	OBS provides REST APIs that support HTTP/HTTPS requests. You can call these APIs to create, modify, and delete buckets, as well as to upload, download, and delete objects.	Configure the endpoint during environment preparation and make the access keys (AK and SK) part of your signature in any requests you send. If you use REST APIs to develop a program, you need to calculate the signature based on the signature algorithm defined by OBS and add the signature to the requests. For details, see **User Signature Authentication**.	**API Reference**

Tool	Description	How to Use	Reference
OBS Browser+	OBS Browser+, a GUI tool, is a Windows client for managing OBS resources. With OBS Browser+, you can easily manage OBS resources from a local end.	Download OBS Browser+ and use access keys (AK and SK) for identity authentication. You can use OBS Browser+ to directly access OBS resources or you can configure a server address to access OBS resources. For details about how to configure the AK, SK, and endpoint of OBS Browser+, see **Where Can I Obtain Access Keys (AK and SK)**.	**Introduction to OBS Browser+**
obsutil	obsutil is a command line tool for accessing OBS. You can use it to perform common configuration and management operations on OBS. If you are familiar with command line interface (CLI), obsutil is the recommended tool for batch processing and automated tasks.	Download obsutil, configure the server address, and use access keys (AK and SK) for identity authentication. For details about how to configure the AK, SK, and endpoint of obsutil, see **Initializing Configurations**.	**Introduction to obsutil**
obsfs	obsfs, built on the basis of Filesystem in Userspace (FUSE), is a file system tool provided by OBS for mounting parallel file systems to Linux operating systems. It enables you to easily access the virtually unlimited storage space in OBS in the same way as you would use a local file system.	Download the obsfs tool and use access keys (AK and SK) for identity authentication. For details about how to configure the AK and SK of obsfs, see **Initializing obsfs**.	**Introduction to obsfs**

3 Accessing OBS

OBS Domain Name

The following two concepts are related to OBS domain names:

1. **Endpoint:** OBS provides an endpoint for each region. An endpoint is a domain name to access OBS in a certain region and is used to receive access requests sent from that region. For the mapping between regions and OBS endpoints, see **Regions and Endpoints**.
2. **Bucket domain name**: Each bucket in OBS has a domain name. A domain name is the Internet address of a bucket and can be used to access the bucket over the Internet. It is typically used in cloud application development and data sharing scenarios.

   An OBS bucket domain name is in the *BucketName*.*Endpoint* format.

   BucketName indicates the name of a bucket, and *Endpoint* indicates the OBS domain name of the region where the bucket is located.

Table 3-1 lists the bucket domain name, and other domain names of OBS, including their formats and protocols.

Table 3-1 OBS domain names

Type	Structure	Description	Protocol Type
Region domain name	[Structure] *Endpoint* [Example] obs.ap-southeast-1.myhuaweicloud.com	Each region has an OBS endpoint, which is the OBS service domain name of the region. For a complete mapping between regions and OBS endpoints, see **Regions and Endpoints**.	HTTPS HTTP

Type	Structure	Description	Protocol Type
Bucket domain name	[Structure] *BucketName*.*Endpoint* [Example] bucketname.obs.ap-southeast-1.myhuaweicloud.com	After a bucket is created, you can use the domain name to access the bucket. You can assemble the domain name by putting the bucket name and endpoint together, or you can obtain it by **viewing the basic bucket information** on OBS Console or OBS Browser+.	HTTPS HTTP
Object domain name	[Structure] *BucketName*.*Endpoint*/*ObjectName* [Example] bucketname.obs.ap-southeast-1.myhuaweicloud.com/object.txt	After an object is uploaded to a bucket, you can use the object domain name to access the object. You can assemble the object domain name by putting the bucket name, OBS service endpoint, and object name together, or you can obtain it by **Viewing Object Information** on OBS Console or OBS Browser+. Alternatively, you can call the GetObjectUrl API through an SDK to obtain the object domain name.	HTTPS HTTP
Static website domain name	[Structure] *BucketName*.**obs-website.***Endpoint* [Example] bucketname.obs-website.ap-southeast-1.myhuaweicloud.com	A static website domain name is a bucket domain name when the bucket is configured to host a static website.	HTTPS HTTP
User-defined domain name	Domain names that have been licensed by the Ministry of Industry and Information Technology (MIIT) of China.	You can bind a user domain name to a bucket so that you can access the bucket through the user domain name.	HTTP

Endpoints

OBS has an endpoint in each region.

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.

1. **Scenarios where the endpoint in a request must be 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 **ap-southeast-1**, you can list objects in the bucket by sending a request shown in the following example:

   A correct example of request and response for listing objects:

   [Request]

   ```
   GET / HTTP/1.1
   Host: mybucket.obs.ap-southeast-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: Rdj0zZvRkihRcjcQUqjkDGt8JuAgi2CGuLiP7Pv/cYYplsS0xTFJQHP5vSg5yOYC
   Content-Type: application/xml
   Date: Thu, 10 Mar 2016 16:58:12 GMT
   x-obs-bucket-location: ap-southeast-1
   Content-Length: 259

   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <ListBucketResult xmlns="http://obs.myhuaweicloud.com/doc/2015-06-30/">
     <Name>mybucket</Name>
     <Prefix/>
     <Marker/>
     <MaxKeys>1000</MaxKeys>
     <IsTruncated>false</IsTruncated>
     <Contents>
       <Key>object001</Key>
       <LastModified>2015-07-01T00:32:16.482Z</LastModified>
       <ETag>"2fa3bcaaec668adc5da177e67a122d7c"</ETag>
       <Size>12041</Size>
       <Owner>
         <ID>b4bf1b36d9ca43d984fbcb9491b6fce9</ID>
       </Owner>
       <StorageClass>STANDARD</StorageClass>
     </Contents>
   </ListBucketResult>
   ```

   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 example above, bucket **mybucket** is in region **ap-southeast-1**. 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 for **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: 0001EF710C000001536176DA465E4E6H
x-obs-id-2: Rdj0zZvRkihRcjcQUqjkDGt8JuAgi2CGuLiP7Pv/cYYplsS0xTFJQHP5vSg5yOYL
Date: Thu, 10 Mar 2016 08:51:30 GMT
Content-Length: 0
```

2. **Scenarios where any endpoint can be used in a request**

   To obtain the bucket list or a bucket's region information, you can use any region endpoint in the API requests you send, because these APIs search for the requested buckets in all regions.

   An example of request and response for obtaining a bucket's region information:

   For example, if bucket **mybucket** is in region **ap-southeast-1** 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: Rdj0zZvRkihRcjcQUqjkDGt8JuAgi2CGuLiP7Pv/cYYplsS0xTFJQHP5vSg5yOYC
   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/">ap-southeast-1</Location>
   ```

Accessing OBS over the Internet

Accessing OBS over a public network may generate inbound traffic for write operations (for example, uploading data to OBS), as well as outbound traffic for read operations (for example, downloading data from OBS). Inbound traffic does not incur fees, but outbound traffic does.

For details about the pricing of outbound traffic over the Internet, see **Product Pricing Details**.

If you access OBS over a public network, you can use a URL to specify resources in OBS. An OBS URL is structured as follows:

Protocol://[*BukcetName*.]*Endpoint*[:*Port*][/*Object*][?*Param*]

Table 3-2 Parameters in an OBS URL

Parameter	Description	Mandatory or Optional
Protocol	The 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.	Mandatory
BucketName	Name of the requested bucket, which uniquely identifies a bucket in OBS.	Optional
Endpoint	Domain name (endpoint) of the region where the OBS bucket is located. For details about the OBS domain name of each region, see **Regions and Endpoints**.	Mandatory
Port	The port enabled for protocols used for sending requests. The value varies depending on software server deployment. If no port number is specified, the protocol uses the default value. Each transmission protocol has its default port number. In OBS, the default HTTP port number is **80** and that of HTTPS is **443**.	Optional
Object	Path of the requested object resource.	Optional
Param	Specific resource contained by a bucket or object. The default value of this parameter indicates that the bucket or object itself is obtained.	Optional

Example: You have a bucket named **mybucket** in the CN-Hong Kong (**ap-southeast-1**) region. The bucket contains an object named **myfolder/myfile.txt**. The URL for accessing the object over the public network is as follows:

https://mybucket.obs.ap-southeast-1.myhuaweicloud.com/myfolder/myfile.txt

NOTE

All API requests except the one for listing objects must contain the **BucketName**. In consideration of the DNS resolution performance and reliability, OBS requires that the bucket name must precede 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.

Accessing OBS over the Intranet

Accessing OBS over a private network refers to accessing OBS through the internal communication network between different Huawei Cloud services. Inbound traffic generated by accessing OBS over an intranet (write operations like uploading data

to OBS) and outbound traffic (read operations like downloading data from OBS) are free of charge.

For example, you can access OBS from an Elastic Cloud Server (ECS) over a private network. Such access is not susceptible to public network quality issues, and it also reduces costs.

OBS provides you with a best practice for configuring such access. For details, see **Accessing OBS from an ECS over the Intranet**.

Checking OBS Version (OBS 2.0 or OBS 3.0)

OBS architecture has undergone two generations: OBS 2.0 and OBS 3.0. A newly created bucket is stored in OBS 3.0 by default, and the bucket version is OBS 3.0. However, previously created buckets are still in OBS 2.0.

Basic OBS features and functions are supported by both OBS 3.0 and OBS 2.0. Some new features are supported only by OBS 3.0, such as image processing and cross-region replication.

You can check the bucket version on OBS Console or use the **Head Bucket** API to check whether your bucket is in OBS 2.0 or OBS 3.0. The details are as follows:

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

If **Bucket Version** is **3.0**, the bucket is stored in OBS 3.0. If not, the bucket is stored in OBS 2.0.

Method 2: Use the Head Bucket API to check the bucket version.

Sample Request:

```
HEAD / HTTP/1.1
Host: bucketname.obs.ap-southeast-1.myhuaweicloud.com
Accept: */*
Date: WED, 01 Jul 2015 02:23:25 GMT
Authorization: auth string
```

Sample 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: 32AAAQAAEAABAAAQAAEAABAAAQAAEAABCS8wS9l00ll4oMWmdniV7XmdAvfewrQq
Date: WED, 01 Jul 2015 02:23:25 GMT
Content-Length: 0
```

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

For details about the **Head Bucket** API, see **Obtaining Bucket Metadata**.

4 Function Overview

For details, see **OBS Function Overview**.

5 Permissions Management

Scenarios

By default, OBS resources (buckets and objects) are private. Only resource owners can access their OBS resources. Without authorization, other users cannot access your OBS resources. OBS permission control means to grant permissions to other accounts 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 the public, so that anyone can access your buckets over the Internet. OBS offers multiple methods to help you to assign resource permissions to others. Resource owners can formulate different permissions control policies based on service requirements to ensure data security.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to configure permissions.

Tool	Reference
OBS Console	**Permission Control**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Permissions and Supported Actions** **Configuring a Bucket Policy** **Configuring a Bucket ACL** **Configuring an Object ACL**
OBS Browser+	-
obsutil	**Setting Bucket Properties** **Setting Object Properties**

6 Storage Classes

Scenarios

OBS provides three storage classes: Standard, Infrequent Access, and Archive. For more information about billing for different storage classes, see **Storage Space**.

Different storage classes meet different requirements for storage performance and costs.

- The Standard storage class features low access latency and high throughput. It is therefore ideal for storing massive quantities of hot files (frequently accessed every month) or small files (less than 1 MB). Its application scenarios include big data analytics, mobile apps, hot videos, and social apps.
- The Infrequent Access storage class is ideal for storing data that is accessed infrequently (less than 12 times a year) yet has quick response requirements. Its application scenarios include file synchronization, file sharing, and enterprise backup.
- The Archive storage class is ideal for archiving data that is rarely accessed (once a year on average). Its application scenarios include data archiving and long-term data backups. The Archive storage class is secure, durable, and inexpensive, and can be used to replace tape libraries. To keep cost low, it may take hours to restore data from the Archive storage class.

Comparison of Storage Classes

Compared Item	Standard	Infrequent Access	Archive
Feature	Top-notch performance, high reliability and availability	Reliable, inexpensive storage with real-time access	Long-term retention of archived data at a low cost
Application scenarios	Cloud applications, data sharing, content sharing, and hot data storage	Web disk applications, enterprise backup, active archiving, and data monitoring	Storage of archives, medical imaging data, and videos, as well as replacement of tape libraries

Compared Item	Standard	Infrequent Access	Archive
Durability	99.999999999%	99.999999999%	99.999999999%
Durability (multi-AZ)	99.9999999999%	99.9999999999%	Multi-AZ not supported
Availability	99.99%	99%	99%
Availability (multi-AZ)	99.995%	99.5%	Multi-AZ not supported
Minimum measurement unit[a]	64 KB	64 KB	64 KB
Minimum storage duration[b]	N/A	30 days	90 days
Data retrieval	N/A	Billed for each GB retrieved.	Data can be restored at a standard or an expedited speed. Billed for each GB retrieved.
Image processing	Supported	Supported	Not supported

NOTE

a: Minimum measurement unit refers to the lower size limit for object billing. For example, if you upload a Standard object of 32 KB (smaller than the minimum measurement unit of 64 KB), you will be billed for a 64 KB object.

b: Minimum storage duration refers to the lower storage limit for object billing. For example, if an Infrequent Access object has been stored in OBS for 20 days (shorter than the minimum storage duration of 30 days) and then deleted, you will be billed for 30 days.

Bucket Storage Classes and Object Storage Classes

You can specify the storage class for a bucket when creating the bucket. You can also change the storage class of a bucket after the bucket is created.

An object inherits the storage class of the bucket where it is uploaded. You can specify a storage class for an object when uploading it, or you can change the object storage class after the object is uploaded.

Changing the storage class of a bucket does not change the storage classes of existing objects in the bucket, but newly uploaded objects inherit the new storage class by default.

NOTE

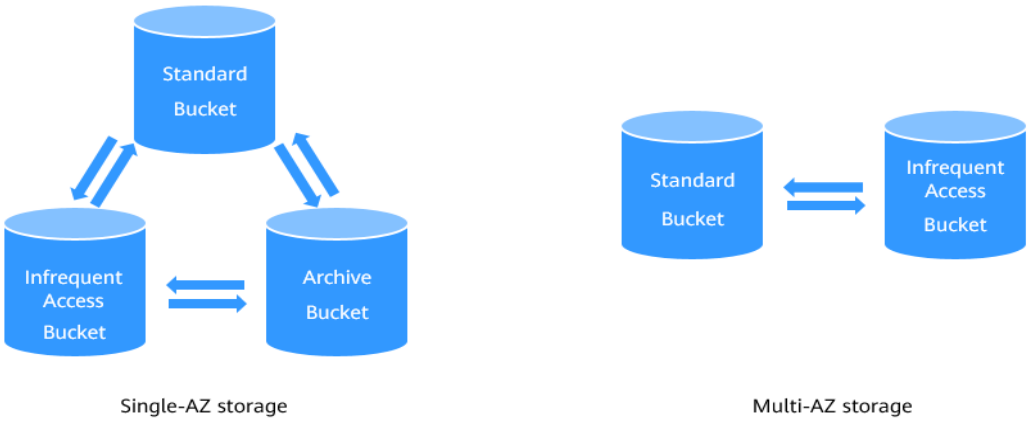
By default, objects in Archive storage class must be restored to Standard storage class before they can be downloaded.

Changing Bucket Storage Classes

The storage class of a bucket can only be manually changed.

Figure 6-1 shows the bucket storage class transition rules. OBS Standard and Infrequent Access storage classes support both single-AZ and multi-AZ redundancy policies, while the Archive storage class supports only the single-AZ redundancy. The data redundancy policy of a bucket cannot be modified, even when its storage class is changed.

Figure 6-1 Bucket storage class transition rules



Notes:

- Changing the storage class of a bucket does not change the storage classes of existing objects in the bucket. The storage class of an object uploaded later will inherit the new storage class of the bucket by default. You can also **configure lifecycle rules** to change storage classes of objects in a batch.

  For example, if **bucket1** is in the Standard storage class and contains **object1**, when **bucket1**'s storage class is transitioned to Infrequent Access, **object1** is still in the Standard storage class. If you upload **object2** to **bucket1** after the transition, **object2** will be in the Infrequent Access storage class.
- After a bucket is changed from Archive to Standard or Infrequent Access, Archive objects in the bucket are not automatically restored.

Changing Object Storage Classes

The storage class of an object can be changed manually or automatically. **Figure 6-2** shows the object storage class transition rules.

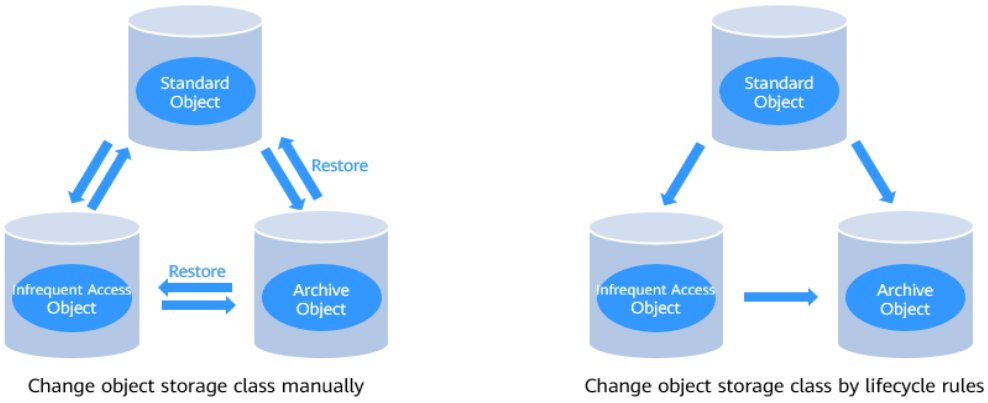
- Manual transition: After an object is uploaded, you can manually change its storage class.

  Objects can be manually changed from Standard to the Infrequent Access or Archive storage class, or from Infrequent Access to Standard or Archive. Objects in the Archive storage class must be restored before they can be manually changed to the Standard or Infrequent Access storage class. Transitioning objects in the Infrequent Access or Archive storage class involves retrieval fees, so you are advised to perform transitions based on the object access frequency and scenario.

- Automatic transition: OBS provides lifecycle rules for you to automatically transition objects from one storage class to another. For details, see **Lifecycle Management**.

  Objects can be automatically transitioned from Standard to the Infrequent Access or Archive storage class, or from Infrequent Access to Archive.

Figure 6-2 Object storage class transition rules



Notes:

- After objects in the Standard storage class are transitioned to Infrequent Access or Archive storage class, their restoration status is Unrestored.
- The minimum storage duration is 30 days for Infrequent Access storage, and 90 days for Archive storage. If an object is transitioned to another storage class before it has been stored for the required minimum storage duration, you need to pay for a full 30 or 90 days.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to configure storage classes for buckets and objects.

Tool	Reference
OBS Console	**Specifying a Storage Class During Bucket Creation** **Specifying a Storage Class During Object Uploads**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.

Tool	Reference
API	**Setting the Default Storage Class for a Bucket** **Specifying a Storage Class During Bucket Creation** (adding the **x-obs-storage-class** header) **Specifying a Storage Class During Object Uploads with PUT** (adding the **x-obs-storage-class** header) **Specifying a Storage Class During Object Uploads with POST** (adding the **x-obs-storage-class** header)
OBS Browser+	-
obsutil	**Specifying a Storage Class During Bucket Creation** **Specifying a Storage Class During Bucket Property Configuration** **Specifying a Storage Class During Object Uploads** **Specifying a Storage Class During Object Property Configuration**

7 Bucket Management

7.1 Overview

A bucket is a virtual container used to store **objects** in OBS. OBS offers a flat structure based on buckets and objects. This structure enables all objects to be stored at the same logical layer, rather than being stored hierarchically.

Buckets have their own attributes, such as storage classes (for more information, see **Storage Classes**), access permissions, and **regions**. You can specify access permissions, a storage class, and a region when creating a bucket. You can also configure advanced attributes to fit different storage requirements.

OBS provides the following storage classes: Standard, Infrequent Access, and Archive. With support for different storage classes, OBS caters to diverse storage performance and cost requirements. When creating a bucket, you can specify a storage class for the bucket, which can be changed later.

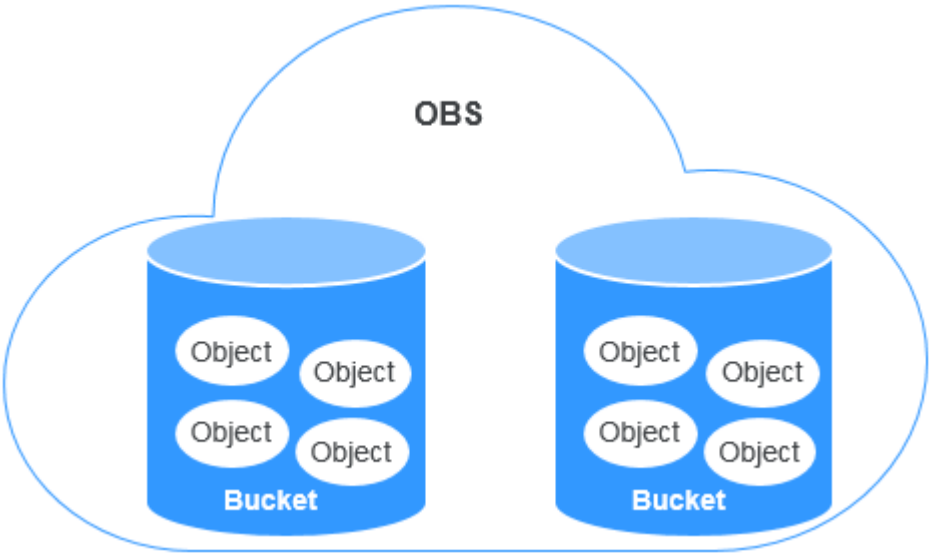
On OBS, each bucket name must be unique and cannot be changed. The region where a bucket resides cannot be changed once the bucket is created. When you create a bucket, OBS creates a default access control list (ACL) that grants the authorized user permissions on the bucket. Only authorized users can perform operations such as creating, deleting, viewing, and configuring buckets.

An account (including all IAM users under the account) can create a maximum of 100 buckets. You can leverage the fine-grained permission control capability of OBS to properly plan and use buckets. For example, you can create folders in a bucket based on object prefixes and use **fine-grained permission control** to isolate data between different departments. There is no limit on the number and total size of objects in a bucket.

As OBS is based on a RESTful architecture over HTTP and HTTPS, you can use uniform resource locators (URLs) to locate resources.

Figure 7-1 illustrates the relationship between buckets and objects in OBS.

Figure 7-1 Buckets and objects



You can **use different methods** to **create buckets** based on your use habits and storage needs. After a bucket is created, you can use different ways to **upload files (data) to the bucket**, where these files are stored as objects. In OBS, buckets and objects are located in **different regions**. You can use different methods to access the same bucket and resources in the same region.

7.2 Creating a Bucket

Scenarios

Buckets are containers that store objects in OBS. You need to create a bucket before you can start storing data in OBS.

Prerequisites

To create a bucket, make sure you have a registered account, sufficient account balance, access keys (AK and SK), and the endpoint information. For details, see **Getting Started**.

Constraints

- After a bucket is created, its name and region cannot be changed. Specify a proper region and bucket name when creating the bucket.
- An account (including all IAM users under the account) can create a maximum of 100 buckets. You can leverage the fine-grained permission control capability of OBS to properly plan and use buckets. For example, you can create folders in a bucket based on object prefixes and use **fine-grained permission control** to implement permission isolation between departments. There is no limit on the number and total size of objects in a bucket.
- A bucket name is part of the access domain name and needs to be resolved. Therefore, the bucket name must conform to the **DNS domain naming rules**.

When receiving a bucket creation request, the OBS strictly checks the bucket name. A bucket name:

- Must be unique across all accounts and regions. The name of a deleted bucket can be reused for another bucket or a parallel file system at least 30 minutes later after the deletion.
- Must be 3 to 63 characters long. Only lowercase letters, digits, hyphens (-), and periods (.) are allowed.
- Cannot start or end with a period (.) or hyphen (-), and cannot contain two consecutive periods (..) or contain a period (.) and a hyphen (-) adjacent to each other.
- Cannot be formatted as an IP address.

NOTICE

If you use a bucket with periods (.) in its name to access OBS, the client will display a message indicating that the bucket is risky, for example, a red alarm may be displayed in the browser security prompt. This is because the SSL wildcard certificate matches only buckets without periods (.) in their names when HTTPS is used for OBS access. We recommend that you avoid using periods (.) in bucket names.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to create a bucket.

Tool	Reference
OBS Console	**Creating a Bucket**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Creating a Bucket**
OBS Browser+	-
obsutil	**Creating a Bucket**

How to Access a Bucket

After a bucket is created, you can use the domain name to access the bucket. You can assemble the domain name by putting the bucket name and endpoint together, or you can obtain it by **viewing the basic bucket information** on OBS Console or OBS Browser+.

An access domain name is structured as follows:

[Structure] *BucketName.Endpoint*

[Example] *bucketname*.**obs.ap-southeast-1.myhuaweicloud.com**

Causes of Bucket Creation Failures and Solutions

For details, see **Why Am I Unable to Create a Bucket?**

7.3 Listing Buckets

Scenarios

You can list all created buckets to view their information.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to list buckets.

Tool	Reference
OBS Console	Log in to OBS Console and select **Object Storage**. Then all buckets under your account will be displayed.
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Listing Buckets**
OBS Browser+	Log in to OBS Browser+. All buckets under your account are displayed in the bucket list.
obsutil	**Listing Buckets**

7.4 Viewing Bucket Information

Scenarios

After creating a bucket on OBS Console, you can view the bucket information, including its basic statistics and details.

How to Use

You can use OBS console, APIs, SDKs, OBS Browser+, or obsutil to view basic bucket information.

Tool	Reference
OBS Console	**Viewing a Bucket's Basic Information**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Obtaining Bucket Metadata**

Tool	Reference
OBS Browser+	**Viewing a Bucket's Basic Information**
obsutil	**Querying Bucket Properties**

7.5 Managing Bucket Quotas

Scenarios

By default, neither the entire OBS system nor any single bucket has limitations on the total size or quantity of objects that can be stored. You can set a quota for a bucket to limit the total size of objects that can be uploaded to the bucket. If the total object size reaches the upper limit, object uploads will fail.

A bucket quota can control object uploading only after the quota is set. If the bucket quota is less than the capacity of the uploaded objects, the 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 storage capacity is less than the quota limit.

NOTE

- A bucket quota must be a non-negative integer, 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.

How to Use

You can use APIs and SDKs to manage bucket quotas.

Tool	Reference
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Bucket Storage Quota**

7.6 Deleting Buckets

Scenarios

You can delete unwanted buckets to free up the quota of buckets. Make sure that a bucket is emptied before you delete it.

An empty bucket must meet the following requirements:

- There is no object or any historical version of an object in the bucket.

- There is not any incomplete multipart upload in the bucket. In other words, there are no fragments in the bucket.

  NOTE

  - If versioning is enabled for the bucket, ensure that all historical versions and versions with the **Delete Marker** (which are also considered as historical versions) have been deleted.
  - The name of a deleted bucket can be reused at least 30 minutes after the deletion.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to delete buckets.

Tool	Reference
OBS Console	**Deleting a Bucket**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Deleting Buckets**
OBS Browser+	**Deleting a Bucket**
obsutil	**Deleting a Bucket**

Causes of Bucket Deletion Failures and Solutions

For details, see **Why Can't I Delete a Bucket?**

8 Object Management

8.1 Object Overview

An object is the basic unit of data storage on OBS. It consists of object data and object metadata that describes object attributes. Data uploaded to OBS is stored as objects in **buckets**.

An object consists of data, metadata, and a key.

- A key specifies the name of an object. An object key is a UTF-8 string up to 1024 characters long. Each object is uniquely identified by a key within a bucket.
- Metadata describes an object, and can be system-defined or user-defined. The metadata is a set of key-value pairs that are assigned to the object stored in OBS.
  - System-defined metadata is automatically assigned by OBS for processing objects. Such metadata includes Date, Content-Length, Last-Modified, Content-MD5, and more.
  - User-defined metadata is specified when you upload objects and is used to describe objects.
- Data refers to the content of the object.

Generally, objects are managed as files. However, OBS is an object-based storage service and there is no concept of files and folders. For easy data management, OBS provides a method to simulate folders. By adding a slash (/) in an object name, for example, **test/123.jpg**, you can simulate **test** as a folder and **123.jpg** as the name of a file under the **test** folder. However, the object key remains **test/123.jpg**.

When uploading an object, you can set a **storage class** for the object. If no storage class is specified, the object storage class keeps the same as that of the bucket. You can also change the storage class of an existing object in a bucket.

You can **upload files (data) to a bucket** in **the ways** you like based on your habits and service scenarios. OBS then stores the files as objects in the bucket. In OBS, buckets are located in **different regions**. No matter what method you use, you can access the same bucket and its resources in the same region.

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.

Alphanumeric characters (also known as unreserved characters)	[0-9a-zA-Z]
Special characters (also known as reserved characters)	Exclamation mark (!) Hyphen (-) Underscore (_) Period (.) Asterisk (*) Single quotation mark (') Left bracket "(" Right bracket ")"

The following are examples of valid object key names:

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

Percent-Encoding of Reserved Characters

If a reserved character has a special meaning (known as reserved purpose) in a URI and the character must be used for other purposes in the URI, this character must be percent-encoded. Use UTF-8 to encode non-ASCII characters. Otherwise, the names of the objects that are uploaded to OBS may be different from what is expected. For example, if reserved character "/" is used as the delimiter of path components in a URI, the character has a special meaning (separating a bucket name from an object name). If "/" is used in a component of the path in a URI, use three characters "%2F" or "%2f" to replace "/".

Characters That May Require Special Processing

Characters that require encoding in a key name

- Ampersand (&)
- Dollar sign ($)
- Semicolon (;)
- Colon (:)
- Plus sign (+): OBS decodes plus signs (+) in a request URI into spaces. If an original object key name contains plus signs (+), it must be encoded into %2B before being put into the request URI.

- Space: A large number of consecutive spaces may be lost in some cases.
- Equality sign (=)
- At sign (@)
- Comma (,)
- Question mark (?)
- ASCII characters: 00–1F in hexadecimal form (0–31 in decimal form) and 7F (127 in decimal form)

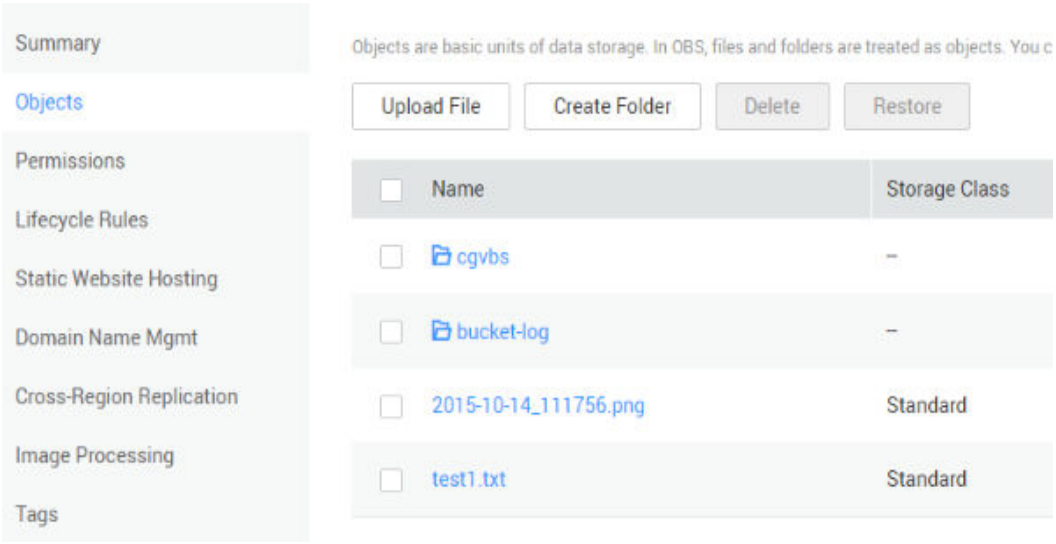
Avoid using the following characters in key names, because these characters require a lot of special processing to keep consistency across all applications.

- Backslash (\)
- Left brace ({)
- Non-printable ASCII characters (128–255 decimal characters)
- Insert symbol (^)
- Right brace (})
- Percentage character (%)
- Accent/Untick (`)
- Right square bracket (])
- Quotation mark
- Greater than sign (>)
- Left square bracket ([)
- Tilde (~)
- Less than sign (<)
- Hashtag (#)
- Vertical bar (|)

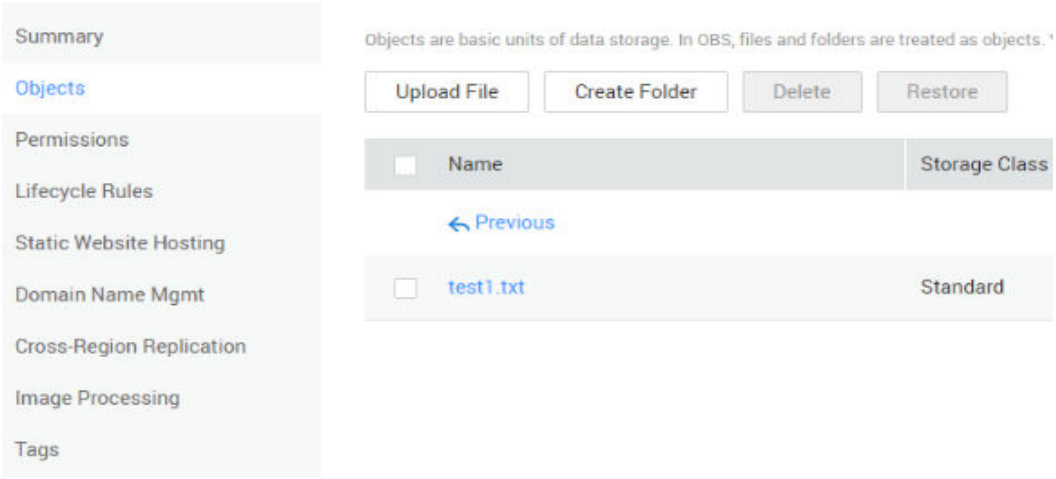
Note that OBS adopts a flat structure, where you create buckets and store objects in buckets. There are no sub-buckets or sub-folders in the structure. However, you can use key name prefixes and delimiters to deduce the logical structure like OBS Console does. The folder concept is available on OBS Console. 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.



Assume that your bucket (**companybucket**) contains two objects with the following object keys:

- obj
- 1/../obj

If you call an API (for example, using the SDK) to obtain the objects, you can successfully get them. If you access the objects from the console, object **obj** will be returned based on the relative relationship. Because **../** has special semantics in a URI, avoid using it in object keys.

8.2 Creating a Folder

Scenarios

You can create a folder in a bucket to facilitate classification and management of data stored in OBS.

Unlike a file system, OBS does not involve the concepts of file and folder. For easy data management, OBS provides a method to simulate folders. In OBS, an object is simulated as a folder by adding a slash (/) to the end of the object name on OBS Console. When objects are listed by calling the API, the obtained object name is the path of the object, and the content following the last slash (/) is the actual object name. If the path ends with a slash (/), it indicates that the object is a folder. The hierarchical depth of the object does not affect the performance of accessing the object.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to create a folder.

Tool	Reference
OBS Console	**Creating a Folder**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page. **NOTE** To create a folder in OBS is to create an object whose size is 0 and whose name ends with a slash (/), in essential.
API	- **NOTE** To create a folder in OBS is to create an object whose size is 0 and whose name ends with a slash (/), in essential.
OBS Browser+	**Creating a Folder**
obsutil	**Creating a Folder**

8.3 Uploading an Object

Scenarios

You can upload files or folders to an existing OBS bucket. If you want to classify files to be uploaded, create a folder and upload related files to the folder.

For details about how to create a folder, see **Creating a Folder**.

These files can be texts, images, videos, or any other type of files.

NOTICE

OBS allows you to upload objects to buckets in a specified region, but Huawei Cloud does not detect the object content you uploaded. If your object uploads involve cross-border transfer, ensure that your use complies with relevant laws and regulations.

Constraints

- OBS Console has restrictions on the size and number of files uploaded.
  - In regions where batch upload is supported, a maximum of 100 files can be uploaded at a time, with a total size of no more than 5 GB. If you upload only one file in batch upload mode, this file cannot exceed 5 GB in size.
  - In regions where batch upload is not supported, only one file can be uploaded at a time, with a size of no more than 50 MB.
- You can use OBS Browser+ and obsutil to upload files with a total size smaller than 48.8 TB. OBS Browser+ allows you to upload a maximum of 500 files at a time. There is no limit on the number of files you can upload using obsutil at a time.
- Using SDK/API PUT, POST, or appendable operations, you can upload files with a total size smaller than 5 GB.
- Using SDK/API multipart upload or SDK resumable upload, you can upload files with a total size smaller than 48.8 TB.
- The batch upload function is available only when the following conditions are met:
  a. The region where the bucket resides supports batch upload. Currently, the following regions support batch upload: CN-Hong Kong, AP-Bangkok, and AP-Singapore.
  b. The bucket version must be **3.0**. For details about how to view the bucket version, see **Checking OBS Version (OBS 2.0 or OBS 3.0)**.
- You can directly upload folders through OBS Console, OBS Browser+, or obsutil.
- You can set a bucket quota to limit the capacity of objects to be uploaded to a bucket. For details, see **Managing Bucket Quotas**.
- If versioning is disabled and the name of a newly uploaded file is the same as that of a file in the bucket, the newly uploaded file automatically overwrites the existing file and does not retain the ACL information of the existing file. If the name of the newly uploaded folder is the same as that of a folder in the bucket, the two folders will be merged, and files in the new folder will overwrite namesake files in the old folder.
- If versioning is enabled and the name of a newly uploaded file is the same as that of a file in the bucket, a new version is added to the existing file. For details about versioning, see **Versioning**.

Upload Methods

Upload Method	Description
PUT (Streaming upload)	Use the PUT or POST method when the size of the file to be uploaded is less than 5 GB. For details about the differences between the two upload methods, see **What Are the Differences Between PUT and POST Upload Methods?**

Upload Method	Description
POST (browser-based upload)	
Multipart upload	Use this method when the size of the file to be uploaded is greater than 5 GB and less than 48.8 TB. For details, see **Multipart Upload**. **NOTE** If you have more data (larger than 48.8 TB) to OBS, refer to **Migrating Local Data to OBS**.
Resumable transfer	Uploading large files often fails due to poor network conditions or program breakdowns. It is a waste of resources to restart the upload process upon an upload failure, and the restarted upload process may still suffer from the unstable network. To resolve such issues, you can use the API for resumable upload, whose working principle is to divide the to-be-uploaded file into multiple parts and upload them separately. This method saves resources and improves efficiency upon the re-upload, and speeds up the upload process by concurrently uploading parts. For details, see **Does OBS Support Resumable Data Transfer?**
Synchronous upload of incremental objects	This method synchronizes all content in the local source path to the specified target bucket in OBS, ensuring that the content is consistent between the local path and the target bucket. Incremental synchronization has the following meanings: 1) Increment: Compare the source file with the target object and upload only the source file that has changes. 2) Synchronization: After the command is executed, ensure that the local source path is a subset of the target bucket specified by OBS. That is, any file in the local source path has its corresponding object in the target bucket on OBS. For details, see **Synchronously Uploading Incremental Objects**.
Appendable upload	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. For details, see **Appending an Object**.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to upload an object.

Tool	Reference
OBS Console	**Uploading a File**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Operations on Objects**
OBS Browser+	**Upload by Dragging**
obsutil	**Uploading an Object**

Related Operations

You can specify its **storage class** when uploading an object or change its storage class after the upload.

- Objects can be manually changed from Standard to Infrequent Access or Archive storage class, or from Infrequent Access to Standard or Archive storage class, but objects in the Archive storage class must be restored before they can be changed to Standard or Infrequent Access. Transitioning objects in the Infrequent Access or Archive storage class involves retrieval fees, so you are advised to perform transitions based on the object access frequency and scenario.
- The minimum storage duration is 30 days for Infrequent Access storage, and 90 days for Archive storage. If an object is transitioned to another storage class before it has been stored for the required minimum storage duration, you need to pay for a full 30 or 90 days.
- You can also configure a lifecycle rule to change the storage class of an object. For details, see **Lifecycle Management**.

Causes of Object Upload Failures and Solutions

For details, see **Why Am I Unable to Upload an Object?**

8.4 Downloading an Object

Scenarios

You can download objects from OBS to a local computer.

OBS supports batch download of objects. For details, see **Does OBS Support Batch Download?**

NOTE

Objects in the Archive storage class can be downloaded only when they are in the **Restored** state.

How to Use

You can use OBS Console, OBS Browser+, APIs, SDKs, or obsutil to download objects and use OBS Browser+ and obsutil to download folders.

Tool	Reference
OBS Console	**Downloading a File**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Downloading Objects**
OBS Browser+	**Downloading a File or Folder**
obsutil	**Downloading an Object**

8.5 Listing Objects

Scenarios

You can view the list of created objects by listing objects.

How to Use

You can use OBS Console, OBS Browser+, APIs, SDKs, or obsutil to list objects.

Tool	Reference
OBS Console	Log in to OBS Console. Select **Object Storage**, click the name of a bucket, and choose **Objects** from the left navigation pane to view all objects in the bucket.
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Listing Objects in a Bucket**
OBS Browser+	-
obsutil	**Listing Objects**

8.6 Copying Objects

Scenarios

You can copy objects in OBS. An object of up to 5 GB can be copied by each operation. To copy an object larger than 5 GB, use the multipart upload API. Specifically, this copy operation allows you to:

- Create a copy for an object.
- Rename an object by creating a copy for it and deleting the source object.
- Edit object metadata. Each object has metadata, which is a set of name-value pairs. You can set object metadata when uploading the object. After you upload the object, you can modify its metadata using an API. For details, see **Modifying Object Metadata**. You can also create an object copy and set the metadata. In the copy operation, set the target object to be the same as the source object.

  Each object contains metadata, including system metadata and user-defined metadata. You can control some system metadata. When you copy 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, if the source object is configured to be in Standard storage, you may want to change the storage class of the object copy to Infrequent Access. You may also want to modify some user-defined metadata of the source object. If you want to modify metadata, even only one piece of 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.

If you want objects to be automatically copied to another other region, create a cross-region replication rule so that off-site disaster recovery can be ensured. For details, see **Cross-Region Replication**.

NOTE

- OBS allows you to replicate the service data stored in OBS to a specified region, but Huawei Cloud does not detect the stored data and is not responsible for the legal compliance of your use of OBS. If your replication involves cross-border transfer, ensure that your use complies with relevant laws and regulations.
- When versioning is disabled for a bucket, if you make a copy of **objecta** and save it as **objectb**, and an object named **objectb** already exists before this copy, the new **objectb** will overwrite the existing one. After the copy succeeds, only new **objectb** can be downloaded because old **objectb** has been deleted. Therefore, before copying an object, make sure that there is no object with the same name as the object copy to prevent data from being deleted mistakenly. Source object **objecta** does not change during the copying.
- 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.

How to Use

You can use APIs, SDKs, OBS Browser+, or obsutil to copy objects.

Tool	Reference
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Copying Objects**
OBS Browser+	**Copying a File or Folder**
obsutil	**Copying Objects**

8.7 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, if you want to upload a 500 MB object to an OBS bucket, you can use OBS Browser+ to upload the object using a multipart upload. OBS Browser+ divides the object into small parts and then uploads the parts. Alternatively, you can call the multipart upload API, 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 object in multiple parts. You can upload a new large 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 a new 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

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 a Part

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 a Part

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 successful only based on the **status_code** in the returned HTTP header. If **200** is returned for **status_code**, the server has received the request and started to process the request. The copy is successful only when the body in the response contains ETag.

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 copying a part. Otherwise, data may be deleted by mistake. The source object does not change during the copying.

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 selected for merge, the parts that are not merged will be automatically deleted by the system. The parts that are not merged cannot be restored after being deleted. Before merging the parts, adopt the API 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 the parts of 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. 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 Uploads

You can obtain the list of initialized multipart upload tasks by listing the multipart upload tasks in the bucket. Initialized multipart upload tasks refer to the multipart upload tasks that are not merged or canceled after initialization. A maximum of 1000 multipart upload tasks can be returned for each request. If the number of

ongoing multipart upload tasks exceeds 1000, you need to send more requests to query the remaining tasks.

Table 8-1 lists the restrictions on listing multipart uploads.

Table 8-1 Restrictions on listing multipart uploads

Item	Restriction
Object size	Up to 48.8 TB
Maximum number of parts for each upload task	10,000
Part number	1–10,000 (included)
Part size	The part size is between 5 MB to 5 GB. The size of the last part is between 0 bytes to 5 GB.
Maximum number of uploaded parts returned in response to the request for listing uploaded parts.	1000
Maximum number of initialized multipart upload tasks returned in response to the request for listing initialized multipart tasks.	1000

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.

Operation	Required Permission
Initiating a multipart upload	To perform this operation, you need to have the PutObject permission. 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.

Operation	Required Permission
Copying parts	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.
Assembling parts	To perform this operation, you need to have the PutObject permission. Only the initiator of a multipart upload can assemble parts. The bucket owner must grant the multipart upload initiator the PutObject permission so that the initiator can complete the multipart upload.
Canceling a multipart upload	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.
Listing uploaded parts	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.

Operation	Required Permission
Listing multipart uploads	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.

REST APIs Applicable to Multipart Upload

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

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

8.8 Viewing Object Information

Scenarios

After uploading an object, you can view the information or properties of the object.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to view the information of an object.

Tool	Reference
OBS Console	-
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Querying Object Metadata**
OBS Browser+	-
obsutil	**Querying Object Properties**

8.9 Sharing a File

Scenarios

You can share files with other users temporarily or permanently.

- Temporary sharing:

  Share a file to other users through the temporary URL of the file. All shared URLs are temporary with a validity period.

  A temporary URL consists of the access domain name and the temporary authentication information of a file. Example:

  ```
  https://bucketname.obs.ap-southeast-1.myhuaweicloud.com:443/image.png?
  AccessKeyId=xxx&Expires=xxx&x-obs-security-token=xxx&Signature=xxx
  ```

  The temporary authentication information contains the **AccessKeyId**, **Expires**, **x-obs-security-token**, and **Signature** parameters. The **AccessKeyId**, **x-obs-security-token**, and **Signature** parameters are used for authentication. The **Expires** parameter specifies the validity period of the authentication.

  For details about the temporary authentication method and parameters, see **Authentication of Signature in a URL** in the *Object Storage Service API Reference*.

- Permanent sharing:

  If you set the permission for an object to allow anonymous users to read it, anonymous users can access the object through the URL that you shared. For details about how to configure permissions, see **Granting Public Read Permissions on Objects to Anonymous Users**.

  The method of using a browser to access objects varies depending on the object type. You can directly open **.txt** and **.html** files using a browser. However, when you open **.exe** and **.dat** files using a browser, the files are automatically downloaded to your local computer.

  For details about how to obtain the object access URL, see **How Do I Obtain the Access Path to an Object?**

  If access to an object using the URL fails, fix the problem by referring to **Why Can't I Access an Object Through Its URL?**

Constraints

- Encrypted objects cannot be shared.
- Objects in the Archive storage class can be shared only after they have been restored.
- An authorization code is not required for temporarily sharing a file. Temporarily sharing a folder requires an authorization code when the method of sharing by access code is used. For details, see **Sharing a Folder**.
- Only buckets of version 3.0 or later support temporary file sharing. For details about how to query the bucket version, see **Checking OBS Version (OBS 2.0 or OBS 3.0)**.
- The following table describes the URL validity period of the files that are temporarily shared using different tools.

Tool	URL Validity Period
OBS Console	The URL is valid for 1 minute to 18 hours. After an object is shared, the system will generate a URL that contains the temporary authentication information, valid for five minutes since its generation by default. Each time you change the validity period of a URL, OBS obtains the authentication information again to generate a new URL for sharing. The new URL becomes valid at the moment the validity period is changed.
SDKs	Configure parameter **Expires** to specify when a temporary authorization expires. The temporary authorization expires in 24 hours.
API	Configure parameter **Expires** to specify when a temporary authorization expires. The temporary authorization expires in 24 hours.
OBS Browser+	When you log in to OBS Browser+ using an account and password, a shared URL will be valid for 24 hours at most. The default validity period is 10 hours. If a longer validity period is required, use the permanent AK/SK for login.
obsutil	Use the additional parameter **e** to specify when an object download URL expires. The minimum value is 60s and the default one is 300s. There is no upper limit on the expiration time.

How to Use

You can use OBS Console, SDKs, APIs, OBS Browser+, or obsutil to share files.

Tool	Reference
OBS Console	**Sharing a File**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Authentication of Signature in a URL**
OBS Browser+	**Sharing a File or Folder**
obsutil	**Generating the Download Link of an Object**

Related Operations

To download a generated temporary URL with wget, use **""** and **-O** to specify the file name, or an error will occur.

Example:

```
[root@ecs-test ~]# wget "Temporary URL" -O abc
```

In the preceding command, *abc* indicates a new file name.

8.10 Sharing a Folder

Scenarios

You can share folders with other users temporarily or permanently.

- Temporary sharing:

  Share a folder to other users through the temporary URL of the file. All shared URLs are temporary with a validity period.

  Folders can be temporarily shared by access code or URL:

  Sharing by access code

  You need to prepare a six-digit extraction code before sharing a folder. After the sharing link of the folder is created, OBS automatically aggregates the download links of all objects in the folder to a static website that is hosted by a public OBS bucket. Then users who have the temporary URL and extraction code can access the static website and download files.

  A temporary URL consists of the access domain name and the temporary authentication information of a folder. Example:

  ```
  https://e-share.obs-website.ap-southeast-1.myhuaweicloud.com:443/image.png?token=xxx
  ```

  For details about the temporary authentication method and parameters, see **Authentication of Signature in a URL** in the *Object Storage Service API Reference*.

  Sharing by URL

  You can specify a validity period and then share the generated link with others. Anyone can use a signature to access all objects in the shared folder.

  A temporary URL consists of the bucket domain name, object access path, and signature information.

- Permanent sharing:

  If you set the permission for an object to allow anonymous users to read it, anonymous users can access the object through the URL that you shared. For details about how to configure permissions, see **Granting Public Read Permissions on Objects to Anonymous Users**.

  For details about how to obtain the object access URL, see **How Do I Obtain the Access Path to an Object?**

  If access to an object using the URL fails, fix the problem by referring to **Why Can't I Access an Object Through Its URL?**

Constraints

- Folder sharing is available only in some regions. For details about the applicable regions, see **Function Overview**.
- Objects in the Archive storage class can be shared only after they have been restored.

- Only buckets of version 3.0 or later support temporary folder sharing. For details about how to query the bucket version, see **Checking OBS Version (OBS 2.0 or OBS 3.0)**.
- The following table describes the URL validity period of the folders that are temporarily shared using different tools.

Tool	URL Validity Period
OBS Console	The URL is valid for 1 minute to 18 hours. After an object is shared, the system will generate a URL that contains the temporary authentication information, valid for five minutes since its generation by default. Each time you change the validity period of a URL, OBS obtains the authentication information again to generate a new URL for sharing. The new URL becomes valid at the moment the validity period is changed.
API	Configure parameter **Expires** to specify when a temporary authorization expires. The temporary authorization expires in 24 hours.
OBS Browser+	When you log in to OBS Browser+ using an account and password, a shared URL will be valid for 24 hours at most. The default validity period is 10 hours. If a longer validity period is required, use the permanent AK/SK for login.
obsutil	Use the additional parameter **vp** to specify how long an authorization code is valid. The default validity period is one day. The allowed units include m (months), w (weeks), d (days), h (hours), min (minutes), and s (seconds). If no time unit is specified, the value is calculated in seconds.

How to Use

You can use OBS Console, APIs, OBS Browser+, or obsutil to share folders.

Tool	Reference
OBS Console	**Sharing a Folder**
API	**Authentication of Signature in a URL**
OBS Browser+	**Sharing a File or Folder**
obsutil	**Creating an Authorization Code for Directory Sharing**

8.11 Managing Object Metadata

Scenarios

Object metadata is a set of name-value pairs that describe the object, and are used for object management.

Currently, only the metadata defined by the system is supported.

The metadata defined by the system is classified into the following types: system-controlled and user-controlled. For example, metadata such as **Last-Modified** is controlled by the system and cannot be modified. You can modify the metadata such as **ContentLanguage** through the API. The metadata that can be modified is described as follows:

Table 8-2 OBS metadata

Name	Description
ContentDisposition	Provides a default file name for the object that is being requested. When an object is being downloaded or accessed, the file with the default file name is directly displayed in the browser or a download dialog box is displayed if the file is being accessed. For example, select **ContentDisposition** as the metadata name and enter **attachment;filename="testfile.xls"** as the metadata value for an object. If you access the object through a link, a dialog box is directly displayed for downloading objects, and the object name is changed to **testfile.xls**. For details, see the definition about ContentDisposition in HTTP.
ContentLanguage	Indicates the language or languages intended for the audience. Therefore, a user can differentiate according to the user's preferred language. For details, see the definition about ContentLanguage in HTTP.

Name	Description
WebsiteRedirectLocation	Redirects an object to another object or an external URL. The redirection function is implemented using static website hosting. For example, you can perform the following operations to implement object redirection: 1. Set metadata of object **testobject.html** in the root directory of bucket **testbucket**. Select **WebsiteRedirectLocation** for **Name** and enter **http://www.example.com** for **Value**. **NOTE** OBS only supports redirection for objects in the root directory of a bucket. 2. Configure static website hosting for bucket **testbucket**, and set the object **testobject.html** in the bucket as the default home page of the hosted static website. 3. If you access object **testobject.html** through the URL link provided on the **Configure Static Website Hosting** page, the access request is redirected to **http://www.example.com**.
ContentEncoding	Content encoding format when an object is downloaded. The options are as follows: • Standard: **compress**, **deflate**, **exi**, **identity**, **gzip**, and **pack200-gzip** • Others: **br**, **bzip2**, **lzma**, **peerdist**, **sdch**, **xpress**, **xz**
CacheControl	Cache behavior of the web page when the specified object is downloaded. • Cacheability: **public**, **private**, **no-cache**, and **only-if-cached** • Expiration time: **max-age=<*seconds*>**, **s-maxage=<*seconds*>**, **max-stale[=<*seconds*>]**, **min-fresh=<*seconds*>**, **stale-while-revalidate=<*seconds*>**, **stale-if-error=<*seconds*>** • Re-verification and reloading: **must-revalidate**, **proxy-revalidate**, **immutable** • Others: **no-store**, **no-transform**
Expires	Cache expiration time (GMT)
ContentType	File type of an object. For details, see **Object Metadata Content-Type**.

NOTE

- When versioning is enabled for a bucket, you can set metadata for objects which are **Latest Version**, but cannot set metadata for objects which are **Historical Version**.
- You cannot configure object metadata for an Archive object.

Object Metadata Content-Type

When an object is uploaded to OBS through OBS Console or a tool, the system automatically matches the value of **Content-Type** based on the file name extension of the object. When you access an object through a web browser, the system specifies an application to open the object according to the value of **Content-Type**. You can modify the **Content-Type** of an object based on its file name extension.

NOTE

If you upload an object by calling an API, specify the value of **Content-Type** because the system does not automatically match the value of **Content-Type**.

Table 8-3 Common Content-Type values

File Name Extension	Content-Type	File Name Extension	Content-Type
* (binary stream, which does not know the type of the file to be downloaded)	application/octet-stream	.7z	application/x-7z-compressed
.001	application/x-001	.301	application/x-301
.323	text/h323	.906	application/x-906
.907	drawing/907	.a11	application/x-a11
.acp	audio/x-mei-aac	.ai	application/postscript
.aif	audio/aiff	.aifc	audio/aiff
.aiff	audio/aiff	.anv	application/x-anv
.asa	text/asa	.asf	video/x-ms-asf
.asp	text/asp	.asx	video/x-ms-asf
.au	audio/basic	.avi	video/avi
.awf	application/vnd.adobe.workflow	.biz	text/xml
.bmp	application/x-bmp	.bot	application/x-bot
.c4t	application/x-c4t	.c90	application/x-c90

File Name Extension	Content-Type	File Name Extension	Content-Type
.cal	application/x-cals	.cat	application/vnd.ms-pki.seccat
.cdf	application/x-netcdf	.cdr	application/x-cdr
.cel	application/x-cel	.cer	application/x-x509-ca-cert
.cg4	application/x-g4	.cgm	application/x-cgm
.cit	application/x-cit	.class	java/*
.cml	text/xml	.cmp	application/x-cmp
.cmx	application/x-cmx	.cot	application/x-cot
.crl	application/pkix-crl	.crt	application/x-x509-ca-cert
.csi	application/x-csi	.css	text/css
.cut	application/x-cut	.dbf	application/x-dbf
.dbm	application/x-dbm	.dbx	application/x-dbx
.dcd	text/xml	.dcx	application/x-dcx
.der	application/x-x509-ca-cert	.dgn	application/x-dgn
.dib	application/x-dib	.dll	application/x-msdownload
.doc	application/msword	.dot	application/msword
.drw	application/x-drw	.dtd	text/xml
.dwf	Model/vnd.dwf	.dwf	application/x-dwf
.dwg	application/x-dwg	.dxb	application/x-dxb
.dxf	application/x-dxf	.edn	application/vnd.adobe.edn
.emf	application/x-emf	.eml	message/rfc822
.ent	text/xml	.epi	application/x-epi
.eps	application/x-ps	.eps	application/postscript
.etd	application/x-ebx	.exe	application/x-msdownload

File Name Extension	Content-Type	File Name Extension	Content-Type
.fax	image/fax	.fdf	application/vnd.fdf
.fif	application/fractals	.fo	text/xml
.frm	application/x-frm	.g4	application/x-g4
.gbr	application/x-gbr	.	application/x-
.gif	image/gif	.gl2	application/x-gl2
.gp4	application/x-gp4	.hgl	application/x-hgl
.hmr	application/x-hmr	.hpg	application/x-hpgl
.hpl	application/x-hpl	.hqx	application/mac-binhex40
.hrf	application/x-hrf	.hta	application/hta
.htc	text/x-component	.htm	text/html
.html	text/html	.htt	text/webviewhtml
.htx	text/html	.icb	application/x-icb
.ico	image/x-icon	.ico	application/x-ico
.iff	application/x-iff	.ig4	application/x-g4
.igs	application/x-igs	.iii	application/x-iphone
.img	application/x-img	.ins	application/x-internet-signup
.isp	application/x-internet-signup	.IVF	video/x-ivf
.java	java/*	.jfif	image/jpeg
.jpe	image/jpeg	.jpe	application/x-jpe
.jpeg	image/jpeg	.jpg	image/jpeg
.jpg	application/x-jpg	.js	text/javascript
.jsp	text/html	.la1	audio/x-liquid-file
.lar	application/x-laplayer-reg	.latex	application/x-latex
.lavs	audio/x-liquid-secure	.lbm	application/x-lbm

File Name Extension	Content-Type	File Name Extension	Content-Type
.lmsff	audio/x-la-lms	.ls	application/x-javascript
.ltr	application/x-ltr	.m1v	video/x-mpeg
.m2v	video/x-mpeg	.m3u	audio/mpegurl
.m4e	video/mpeg4	.mac	application/x-mac
.man	application/x-troff-man	.math	text/xml
.mdb	application/msaccess	.mdb	application/x-mdb
.mfp	application/x-shockwave-flash	.mht	message/rfc822
.mhtml	message/rfc822	.mi	application/x-mi
.mid	audio/mid	.midi	audio/mid
.mil	application/x-mil	.mml	text/xml
.mnd	audio/x-musicnet-download	.mns	audio/x-musicnet-stream
.mocha	application/x-javascript	.movie	video/x-sgi-movie
.mp1	audio/mp1	.mp2	audio/mp2
.mp2v	video/mpeg	.mp3	audio/mp3
.mp4	video/mp4	.mpa	video/x-mpg
.mpd	application/vnd.ms-project	.mpe	video/x-mpeg
.mpeg	video/mpg	.mpg	video/mpg
.mpga	audio/rn-mpeg	.mpp	application/vnd.ms-project
.mps	video/x-mpeg	.mpt	application/vnd.ms-project
.mpv	video/mpg	.mpv2	video/mpeg
.mpw	application/vnd.ms-project	.mpx	application/vnd.ms-project
.mtx	text/xml	.mxp	application/x-mmxp
.net	image/pnetvue	.nrf	application/x-nrf

File Name Extension	Content-Type	File Name Extension	Content-Type
.nws	message/rfc822	.odc	text/x-ms-odc
.out	application/x-out	.p10	application/pkcs10
.p12	application/x-pkcs12	.p7b	application/x-pkcs7-certificates
.p7c	application/pkcs7-mime	.p7m	application/pkcs7-mime
.p7r	application/x-pkcs7-certreqresp	.p7s	application/pkcs7-signature
.pc5	application/x-pc5	.pci	application/x-pci
.pcl	application/x-pcl	.pcx	application/x-pcx
.pdf	application/pdf	.pfx	application/x-pkcs12
.pdx	application/vnd.adobe.pdx	.pic	application/x-pic
.pgl	application/x-pgl	.pl	application/x-perl
.pko	application/vnd.ms-pki.pko	.pls	audio/scpls
.plg	text/html	.png	image/png
.plt	application/x-plt	.pot	application/vnd.ms-powerpoint
.png	application/x-png	.ppm	application/x-ppm
.ppa	application/vnd.ms-powerpoint	.ppt	application/vnd.ms-powerpoint
.pps	application/vnd.ms-powerpoint	.pr	application/x-pr
.ppt	application/x-ppt	.prn	application/x-prn
.prf	application/pics-rules	.ps	application/x-ps
.prt	application/x-prt	.ptn	application/x-ptn
.ps	application/postscript	.r3t	text/vnd.rn-realtext3d

File Name Extension	Content-Type	File Name Extension	Content-Type
.pwz	application/vnd.ms-powerpoint	.ram	audio/x-pn-realaudio
.ra	audio/vnd.rn-realaudio	.rat	application/rat-file
.ras	application/x-ras	.rec	application/vnd.rn-recording
.rdf	text/xml	.rgb	application/x-rgb
.red	application/x-red	.rjt	application/vnd.rn-realsystem-rjt
.rjs	application/vnd.rn-realsystem-rjs	.rle	application/x-rle
.rlc	application/x-rlc	.rmf	application/vnd.adobe.rmf
.rm	application/vnd.rn-realmedia	.rmj	application/vnd.rn-realsystem-rmj
.rmi	audio/mid	.rmp	application/vnd.rn-rn_music_package
.rmm	audio/x-pn-realaudio	.rmvb	application/vnd.rn-realmedia-vbr
.rms	application/vnd.rn-realmedia-secure	.rnx	application/vnd.rn-realplayer
.rmx	application/vnd.rn-realsystem-rmx	.rpm	audio/x-pn-realaudio-plugin
.rp	image/vnd.rn-realpix	.rt	text/vnd.rn-realtext
.rsml	application/vnd.rn-rsml	.rtf	application/x-rtf
.rtf	application/msword	.sam	application/x-sam
.rv	video/vnd.rn-realvideo	.sdp	application/sdp

File Name Extension	Content-Type	File Name Extension	Content-Type
.sat	application/x-sat	.sit	application/x-stuffit
.sdw	application/x-sdw	.sld	application/x-sld
.slb	application/x-slb	.smi	application/smil
.slk	drawing/x-slk	.smk	application/x-smk
.smil	application/smil	.sol	text/plain
.snd	audio/basic	.spc	application/x-pkcs7-certificates
.sor	text/plain	.spp	text/xml
.spl	application/futuresplash	.sst	application/vnd.ms-pki.certstore
.ssm	application/streamingmedia	.stm	text/html
.stl	application/vnd.ms-pki.stl	.svg	text/svg+xml
.sty	application/x-sty	.tdf	application/x-tdf
.swf	application/x-shockwave-flash	.tga	application/x-tga
.tg4	application/x-tg4	.tif	application/x-tif
.tif	image/tiff	.tld	text/xml
.tiff	image/tiff	.torrent	application/x-bittorrent
.top	drawing/x-top	.txt	text/plain
.tsd	text/xml	.uls	text/iuls
.uin	application/x-icq	.vda	application/x-vda
.vcf	text/x-vcard	.vml	text/xml
.vdx	application/vnd.visio	.vsd	application/vnd.visio
.vpg	application/x-vpeg005	.vss	application/vnd.visio
.vsd	application/x-vsd	.vst	application/x-vst
.vst	application/vnd.visio	.vsx	application/vnd.visio

File Name Extension	Content-Type	File Name Extension	Content-Type
.vsw	application/vnd.visio	.vxml	text/xml
.vtx	application/vnd.visio	.wax	audio/x-ms-wax
.wav	audio/wav	.wb2	application/x-wb2
.wb1	application/x-wb1	.wbmp	image/vnd.wap.wbmp
.wb3	application/x-wb3	.wk3	application/x-wk3
.wiz	application/msword	.wkq	application/x-wkq
.wk4	application/x-wk4	.wm	video/x-ms-wm
.wks	application/x-wks	.wmd	application/x-ms-wmd
.wma	audio/x-ms-wma	.wml	text/vnd.wap.wml
.wmf	application/x-wmf	.wmx	video/x-ms-wmx
.wmv	video/x-ms-wmv	.wp6	application/x-wp6
.wmz	application/x-ms-wmz	.wpg	application/x-wpg
.wpd	application/x-wpd	.wq1	application/x-wq1
.wpl	application/vnd.ms-wpl	.wri	application/x-wri
.wr1	application/x-wr1	.ws	application/x-ws
.wrk	application/x-wrk	.wsc	text/scriptlet
.ws2	application/x-ws	.wvx	video/x-ms-wvx
.wsdl	text/xml	.xdr	text/xml
.xdp	application/vnd.adobe.xdp	.xfdf	application/vnd.adobe.xfdf
.xfd	application/vnd.adobe.xfd	.xls	application/vnd.ms-excel
.xhtml	text/html	.xlw	application/x-xlw
.xls	application/x-xls	.xpl	audio/scpls
.xml	text/xml	.xql	text/xml
.xq	text/xml	.xsd	text/xml

File Name Extension	Content-Type	File Name Extension	Content-Type
.xquery	text/xml	.xslt	text/xml
.xsl	text/xml	.x_b	application/x-x_b
.xwd	application/x-xwd	.sisx	application/vnd.symbian.install
.sis	application/vnd.symbian.install	.ipa	application/vnd.iphone
.x_t	application/x-x_t	.xap	application/x-silverlight-app
.apk	application/vnd.android.package-archive	.rar	application/x-rar-compressed
.zip	application/zip	-	-

User-Defined Object Metadata

You can add the user-defined metadata whose name starts with **x-obs-meta-** for easy object management. When you retrieve or query the metadata of the object, the added user-defined metadata will be returned in the response header. The user-defined metadata is limited to 8 KB in size. To measure the total size of user-defined metadata, sum the number of bytes in the UTF-8 encoding of each key and value.

The user-defined metadata keys are case insensitive, but are stored in lowercase by OBS. The key values are case sensitive.

The following is an example.

```
PUT /key HTTP/1.1
Host: bucket01.obs.myhuaweicloud.com
x-obs-meta-Test1: Test Meta1

HEAD /Key HTTP/1.1
Host: bucket01.obs.myhuaweicloud.com
x-obs-meta-test1: Test Meta1
```

Both user-defined metadata keys and their values must conform to US-ASCII characters. If non-ASCII or unrecognizable characters are necessary, they must be encoded or decoded in URL or Base64 on the client side, because the server side does not perform any decoding.

If there are non-US-ASCII or unrecognizable characters in a value and the client does not encode the value, the server side will encode the value in Base64 and encapsulate the value using **?UTF-8?B?<***base64(str)***>?=**. Take **x-obs-meta-nonascii: nonasciiÄÄ** as an example. **nonasciiÄÄ** is encoded as **bm9uYXNjaWnDhMOE** in Base64 and the returned response is **x-obs-meta-nonascii: =?UTF-8?B?bm9uYXNjaWnDhMOE?=**.

The following is an example.

```
PUT /key HTTP/1.1
Host: bucket01.obs.myhuaweicloud.com
x-obs-meta-nonascii: nonasciiÄÄ

HEAD /Key HTTP/1.1
Host: bucket01.obs.myhuaweicloud.com
x-obs-meta-nonascii: =?UTF-8?B?bm9uYXNjaWnDhMOE?=
```

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to customize object metadata.

Tool	Reference
OBS Console	**Configuring Object Metadata**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Modifying Object Metadata**
OBS Browser+	-
obsutil	**Uploading an Object**

8.12 Restoring Objects from Archive Storage

Scenarios

You need to restore an Archive object before downloading it, accessing it using its URL, or setting ACL permissions or object metadata for it.

For details about data restore duration and pricing, see **Product Pricing Details**.

Constraints

- If an Archive object is being restored, you cannot suspend or delete the restoration task.
- You cannot re-restore an object that is in the **Restoring** state.
- After an object is restored, an object copy in Standard storage class will be generated. In this way, there is an object in Archive storage class and also a copy in Standard storage class in the bucket. After an Archive object is restored, the object status displays **Restored**, and the generated object copy in Standard storage class is not displayed in the bucket.

  During the restoration validity period, you will be charged for the storage space occupied by both the object and its copy. The copy will be automatically deleted upon expiration of its validity period.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, or obsutil to restore Archive objects.

Tool	Reference
OBS Console	**Restoring Objects from the Archive Storage**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Restoring Archive Objects**
OBS Browser+	**Restoring a File**
obsutil	**Restoring Objects from OBS Archive**

8.13 Direct Reading

Scenarios

You can enable direct reading if you want to obtain Archive objects immediately or if there are interface compatibility issues when OBS is interconnected with other systems.

With direct reading enabled for a bucket, you can download objects in the Archive storage class without restoring them first.

Direct reading is a billable function. For details, see **Product Pricing Details**.

You can enable direct reading when creating a bucket or enable it for an existing bucket.

How to Use

You can use OBS Console, APIs, SDKs, or OBS Browser+ to set direct reading.

Tool	Reference
OBS Console	**Direct Reading**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Setting the Direct Reading Policy for Archive Objects in a Bucket**
OBS Browser+	-

8.14 Deleting an Object

Scenarios

You can delete unwanted files or folders to save storage space and reduce costs.

NOTICE

If versioning is not enabled for a bucket, deleted objects cannot be recovered. Exercise caution when performing this operation.

Manually or Automatically Deleting Objects

You can manually delete objects anytime. Alternatively, you can use **lifecycle rules** to periodically delete objects from a bucket or clear all objects in a bucket at a time.

Deleting an Object or a Batch of Objects

You can delete one or more objects at a time. For details about how to delete objects in batches, see **Does OBS Support Batch Deletion of Objects?**

NOTE

The batch deletion performance is negatively correlated with the number of objects in a single request. When it comes to QPS, deleting *N* objects is counted as *N* operations. If a large number of objects named with prefixes in lexicographic order are deleted, lots of requests may be concentrated in a specific partition, which results in hot access. This limits the request rate in the hot partition and increases access delay.

To address this problem, you can reduce the number of objects in a single batch deletion request, initiate more concurrent requests, and name objects with random prefixes.

Object Deletion Mechanism When Versioning Is Enabled

When versioning is enabled, OBS uses different deletion methods for different objects.

- Deleting a file or folder does not delete it permanently. The deleted file or folder will be retained in the **Deleted Objects** list and marked with the **Delete Marker**.
  - If you want to delete the file or folder permanently, you need to delete it from the **Deleted Objects** list.
  - To recover a deleted file, you can cancel the deletion by the **Undelete** operation. For details, see **Undeleting an Object**.
- Deleting a version of an object will permanently delete that version. If the deleted version is the latest one, the next latest version becomes the latest version.

When versioning is enabled, files in the **Deleted Objects** list also have multiple versions. Note the following points when deleting different versions of files:

- If you delete a version with the **Delete Marker**, it actually recovers that specific version instead of permanently deleting it. For details, see **Undeleting an Object**.
- If you delete a version without the **Delete Marker**, that specific version is deleted permanently. Even if the object is recovered later, this version will not be recovered.

For more information, see **Object Recovery Mechanism When Versioning Is Enabled**.

How to Use

You can use OBS Console, APIs, SDKs, OBS Browser+, obsutil, or obsfs to delete objects.

Tool	Reference
OBS Console	**Deleting a File or Folder**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Deleting an Object**
OBS Browser+	**Deleting a File or Folder**
obsutil	**Deleting an Object**

8.15 Undeleting an Object

Scenarios

If versioning is not enabled for a bucket, deleted objects cannot be recovered. If versioning is enabled, you can recover deleted objects through the **Undelete** operation. For details about versioning, see **Versioning**.

Object Recovery Mechanism When Versioning Is Enabled

When a bucket has the versioning function enabled, deleting a file from the **Objects** list does not permanently delete it. The deleted file will be retained with the **Delete Marker** in the **Deleted Objects** list. You can recover a deleted object by the **Undelete** operation.

When performing the **Undelete** operation, note the following points:

1. You can only undelete deleted files but not folders.

   After you undelete a deleted file, the file is recovered and will appear in the **Objects** list. Then you can perform basic operations on the file as you normally do on other objects. If the file was stored in a folder before the deletion, it will be recovered to its original path after you undelete it.

2. Deleted files in the **Deleted Objects** also have multiple versions. When deleting different versions of files, note the following points:
   - If you delete a version with the **Delete Marker**, it actually recovers that specific version instead of permanently deleting it.
   - If you delete a version without the **Delete Marker**, that specific version is deleted permanently. Even if the object is recovered later, this version will not be recovered.
3. At least one version without the **Delete Marker** exists in the **Deleted Objects** list. Otherwise, the deletion cannot be canceled.

How to Use

You can use OBS Console to undelete objects.

Tool	Reference
OBS Console	**Undeleting a File**

8.16 Managing Fragments

Scenarios

Fragments refer to incomplete data in a bucket. Data is uploaded to OBS in multipart mode. In the following common and other scenarios, an upload fails and fragments are generated. You need to clear these fragments to free up storage space.

- The network is in poor conditions, and the connection to the OBS server is interrupted frequently.
- The upload task is manually suspended.
- The device is faulty.
- The device is powered off suddenly.

Fragments in OBS consume storage space and are billed based on price rates of storage space. If fragments are generated due to interruptions or failures of multipart upload tasks, you can resume such tasks so that fragments will be cleared, or you can directly clear the fragments to save storage space.

For details about how to manage fragments, see **How Do I Manage Fragments?**

NOTE

- If a bucket fails to be deleted, check whether all fragments have been deleted. If not, delete all fragments from the bucket.
- If no object exists in the bucket but the fee is still being deducted, check whether there are fragments in the bucket. If yes, delete the fragments to reduce storage costs.

How to Use

Table 8-4 Methods for deleting fragments

Tool	Method
OBS Console	OBS Console allows you to batch delete all selected fragments with one click. For details, see **Managing Fragments**.
OBS Browser+	You can delete all of the selected fragments in a bucket in a batch. Alternatively, you can click **Delete All** above the list to delete all fragments.
obsutil	With obsutil, you can delete the failed or interrupted multipart upload task to delete fragments generated by the task. If a bucket has more than one multipart upload tasks, you need to delete all the multipart upload tasks to delete all fragments in the bucket. For details, see **Deleting a Multipart Upload Task**.
OBS API	You can delete fragments from a bucket through the following procedure: 1. Use the **Listing Initiated Multipart Uploads in a Bucket** operation to list all the multipart upload tasks and obtain their upload IDs. 2. Use the **Canceling a Multipart Upload Task** operation to cancel the multipart upload tasks. After these tasks are canceled, all fragments generated by them can be deleted.
OBS SDK	With OBS SDK, fragments are generated when parts of a multipart task are not merged. You can cancel the task to delete generated fragments. The procedure is as follows: 1. Use the **ObsClient.listMultipartUploads** interface to list all multipart upload tasks and obtain their UploadId. 2. Use the **ObsClient.abortMultipartUpload** interface to cancel the multipart upload task so that the generated fragments will be cleared.

9 Parallel File System

Parallel File System (PFS) is a high-performance file system provided by OBS, with access latency in milliseconds. PFS supports TB/s-level bandwidth and millions of IOPS, making it ideal for processing high-performance computing (HPC) workloads.

You can access data in a parallel file system using standard OBS APIs. It also supports data read and write through obsfs, a PFS client that supports POSIX. obsfs can be deployed on an ECS, and then you can use the obsfs to mount a parallel file system to the Linux OS running on the ECS. The mounted parallel file system functions as a local file system. You can manage the mounted parallel file system online, including creating, deleting, renaming files and folders, as well as modifying files.

For details, see **Parallel File System Feature Guide**.

10 Image Processing

Integrated in OBS, the image processing feature provides stable, secure, efficient, and low-cost image processing services. By using this feature, you can slim (downsize), crop, compress, and watermark images, as well as convert the formats of images.

This feature leverages OBS Console and REST APIs. You can process images stored in OBS in various ways anytime and anywhere, and obtain the processed images right away.

For details, see **Image Processing Feature Guide**.

11 Lifecycle Management

Lifecycle management for OBS includes periodically deleting objects from buckets and transitioning between object storage classes based on pre-defined rules.

Scenarios

You may configure lifecycle rules to:

- Periodically delete logs that are only meant to be retained for a specific period of time (a week or a month).
- Transition documents that are seldom accessed to the Infrequent Access or Archive storage class or delete them.
- Store some types of data in OBS for archive purposes, such as digital media, financial and medical records, raw genome sequence data, long-term database backups, and data that must be retained for regulatory compliance.
- Schedule the deletion of a large number of files from a bucket. Manually deleting objects is time-consuming, and only a limited number of objects can be deleted at a time.

You can define lifecycle rules to identify objects in the scenarios above and further manage their lifecycles.

Objects that are no longer frequently accessed can be transitioned to Infrequent Access or Archive, reducing your costs. In short, transition basically means that the object storage class is altered without copying the object. You can also manually change the storage class of an object on the **Objects** page. For details, see **Uploading a File**.

Lifecycle configuration can be added to a bucket with versioning enabled or disabled. By default, versioning is disabled. You can enable it. A version-enabled bucket keeps a current object version and zero or more non-current object versions. You can use versioning and lifecycle configurations together to help reduce storage costs. The predefined lifecycle configuration actions can facilitate management over the lifecycle of the current object versions and non-current object versions.

Precautions

- There is no limit on the number of lifecycle rules in a bucket, but the total size of XML descriptions about all lifecycle rules in a bucket cannot exceed 20 KB.
- A maximum of 20 lifecycle rules can be configured for a parallel file system.
- The minimum storage duration is 30 days for Infrequent Access storage, and 90 days for Archive storage. After an object is transitioned to the Archive storage class, if it stays in this storage class for less than 90 days, you still need to pay for a full 90 days.
- Restrictions on storage class transition:
  - Only transitions from the Standard storage class to the Infrequent Access storage class are supported. To transition objects from Infrequent Access to Standard, you must manually operate it.
  - Only transitions from the Standard or Infrequent Access storage class to the Archive storage class are supported. To transition objects from Archive to Standard or Infrequent Access, you must restore the archived objects first and then manually transition their storage class.
- Object deletion upon expiration:

  You can use the **Expiration** action a filecycle rule to expire an object when the object reaches the end of its lifetime. In a versioning-enabled bucket, you can also use the **NoncurrentVersionExpiration** action to expire non-current object versions.
  - In a bucket with versioning disabled, the **Expiration** action permanently deletes objects.
  - In a bucket with versioning enabled (or suspended), the **Expiration** action retains the current version as a non-current version by adding a delete marker, which then becomes the current version. The **NoncurrentVersionExpiration** action permanently deletes non-current object versions.
- 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 call the GET object API or HEAD object API. These APIs return response headers that provide expiration information about the object.
- After a lifecycle rule is modified, the modification does not apply to the objects that have already met the configured conditions. For example, the original lifecycle rule causes objects meeting the configured conditions expire and then be deleted after they are stored for one day. Under this rule, the objects uploaded on January 1, 2021 will be deleted on January 3, 2021. On January 3, 2021, if you change the lifecycle rule to delete objects after they are stored for seven days, the objects uploaded on January 1, 2021 will still be deleted on January 3, 2021, but this modification will apply to those uploaded on and after January 2, 2021.

Lifecycle Rules

Lifecycle rules have the following key elements:

- Policy: Configure a lifecycle rule that takes effect on specified objects.
  - By prefix: You can specify an object name prefix, so the lifecycle rule will take effect on objects share the same prefix.
  - Entire bucket: You can specify an entire bucket, so the lifecycle rule will take effect on all objects in the bucket.
- Time: a scheduled time when object storage class is transitioned

  You can specify the number of days after which objects that have been last updated and meet specified conditions are automatically transitioned to Infrequent Access or Archive, or are automatically deleted upon expiration.
  - Transition to Infrequent Access: You can specify the number of days after which objects that have been last updated and meet specified conditions are automatically transitioned to Infrequent Access.
  - Transition to Archive: You can specify the number of days after which objects that have been last updated and meet specified conditions are automatically transitioned to Archive.
  - Deleted upon expiration: You can specify the number of days after which objects that have been last updated and meet specified conditions are automatically deleted.

  The lifecycle rule time has the following restrictions:
  - If only one transition is involved, the time should be at least one day later.
  - If multiple transitions are involved, the latter transition should be at least one day longer than the former transition.

    For example, if you want a lifecycle rule to change object storage class and delete expired objects, you can set the transition time to 23 days later and the deletion time to at least 24 days later.

    NOTE

    The minimum storage duration is 30 days for Infrequent Access storage, and 90 days for Archive storage. If objects are deleted by a lifecycle rule before they have been stored for this minimum duration, you still need to pay for a full 30 or 90 days.

    Assume that you have an object in the Standard storage class stored in a bucket and you no longer plan to perform any operations on the object. The lifecycle rule applied to this object changes the object from Standard to Infrequent Access three days later, then to Archive on the fourth day, and finally deletes the object when it expires seven days later. In this case, you will be charged for three days of Standard storage, 30 days of Infrequent Access storage, and 90 days of Archive storage.

Lifecycle Configuration Elements

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 any actions defined in the rule.
- Filtering criteria, identifying objects on which lifecycle rules are applied. 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.

Configuration examples:

Example 1: Lifecycle configuration for a bucket with versioning disabled

By default, versioning is disabled for buckets. In such case, each object in the bucket has only one version.

Assume that versioning is disabled for your bucket. If you want Standard objects whose name starts with **documents/** to transition to the Infrequent Access storage class 30 days after they are uploaded, then transition from Infrequent Access to Archive 60 days after they are uploaded, and finally to be deleted one year after they are uploaded, you can add the following lifecycle configuration for your bucket. This configuration includes the **Transition** and **Expiration** actions and applies to objects whose key prefix is **documents** (specified in the **Prefix** element).

```
<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>
    <Transition>
        <Days>60</Days>
        <StorageClass>COLD</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

Example 2: Lifecycle configuration for a bucket with versioning enabled

You can enable versioning for buckets. If versioning is enabled for a bucket, the bucket will retain the current object version and its non-current object versions. For details, see **Versioning**. 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 enabled, lifecycle configuration provides multiple predefined actions that can be used to manage non-current object versions. In this example, the lifecycle configuration has a rule that specifies two operations (**NoncurrentVersionTransition** and **NoncurrentVersionExpiration**) for objects whose key prefix is **logs/**. The **NoncurrentVersionTransition** action transitions objects to Infrequent Access and then to Archive 30 days and 60 days respectively after they become non-current versions. The **NoncurrentVersionExpiration** action permanently deletes objects 180 days after they become non-current versions.

```
<LifecycleConfiguration>
  <Rule>
```



```
    <ID>sample-rule</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>180</NoncurrentDays>
    </NoncurrentVersionExpiration>
    <NoncurrentVersionTransition>
      <NoncurrentDays>30</NoncurrentDays>
      <StorageClass>WARM</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionTransition>
      <NoncurrentDays>60</NoncurrentDays>
      <StorageClass>COLD</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

You can use the predefined **NoncurrentVersionTransition** and **NoncurrentVersionExpiration** actions to manage non-current versions in your bucket.

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 any actions defined in the rule.
- Prefix (**<Prefix>** element), which identifies objects to which the rule applies.
- Actions that you want to perform on the specified objects (such as **NoncurrentVersionTransition** and **NoncurrentVersionExpiration** mentioned in the previous example). For each action, you must specify a date or time period.

Elements in the lifecycle configuration rule are described as follows:

- Element **ID**

  A lifecycle configuration can have up to 1,000 rules. The ID element uniquely identifies a rule.

- 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 objects whose name starting with **logs/**. If you leave **Prefix** blank, 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 effect of these operations depends on the versioning status of your bucket.

NOTE

By default, versioning is disabled for buckets. You can enable it for your bucket as needed, so that each object in this bucket can have a current version and zero or more non-current versions. You can also suspend versioning. For details, see **Versioning**.

- Action **Transition**

  This action archives objects by changing object storage class to Infrequent Access or Archive. When the date or time period specified in the lifecycle of an object is reached, OBS transitions the object as configured.

  – For buckets with versioning disabled, the **Transition** action changes the object storage class to Infrequent Access or Archive.

  – For buckets with versioning enabled and suspended, the **Transition** action changes the storage class of the current object version to Infrequent Access or Archive. This action does not affect the non-current versions of the object.

- Action **Expiration**

  This action expires objects identified in the rule. Objects become unavailable once they expire. Whether the expired objects will be permanently deleted depends on the versioning status of the bucket.

  NOTE

  The **Expiration** action does not delete incomplete multipart uploads.

  – For buckets with versioning disabled, the **Expiration** action deletes objects permanently and the deleted objects cannot be restored.

  – For buckets with versioning enabled, this action applies only to current object versions, instead of non-current object versions. This action does not permanently delete current object versions. It retains the current version as a non-current version by adding a delete marker to it. This action will not be performed on current object versions that already have the 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 has the 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 recover the deleted object.

For example, you can set a rule to make the object named **photo.gif** expire 5 days after it is uploaded. 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 how a delete marker works, see **Versioning**.

For buckets with versioning 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.

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 non-current versions. You can use these actions to request OBS to perform specific operations on non-current object versions. These actions do not affect current object versions.

NoncurrentVersionTransition: Specifies the time period after which non-current versions will be transited from Standard to Infrequent Access or Archive.

NoncurrentVersionExpiration: Specifies the time period after which non-current object versions will be permanently deleted. The deleted objects cannot be recovered.

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 non-current version.

At 10:30 (UTC time) on January 1, 2016, you created object **photo.gif** with version ID **111111**. At 10:30 (UTC time) on January 2, 2016, you accidentally deleted this object and OBS created a delete marker with version ID **4857693**. In the next five days, you were allowed to recover the original object **photo.gif** with version ID **111111**.

At 00:00 (UTC time) on January 8, 2016, 5 days after object **photo.gif** with version ID **111111** became non-current, the **NoncurrentVersionExpiration** action permanently deleted this object version.

How does OBS calculate how long an object has become non-current?

In a bucket with versioning enabled, an object can have multiple versions. There is always one current version and zero or more non-current versions. Each time an object is uploaded, the current version is saved as a non-current version, and the newly-uploaded version (the successor) becomes the current version. To determine the number of days an object version has become non-current, OBS checks when its successor was created. OBS uses the number of days since its successor was created as the number of days an object has been non-current.

Restoring a non-current version using the lifecycle configuration

You can use either of the following methods to restore an object to a non-current version:

1. Copy a non-current object version of the object to the same bucket. The copied version will become the current version, and all object versions are reserved.

2. Permanently delete the current version of the object. Deleting the current version of an object actually turns a non-current version into the current version.

You are advised to use the first method. 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). Meanwhile, the expiration action you configured for non-current versions may permanently delete non-current object versions, including those you want to restore. Therefore, copying a non-current version, as described in the first method, is more secure.

Table 11-1 lists the relationship between the lifecycle configuration rule and versioning status of a bucket.

Table 11-1 Lifecycle actions and bucket versioning status

Action	Bucket with Versioning Not Enabled	Bucket with Versioning Enabled	Bucket with Versioning Suspended
Transition (Performed when a date or time period specified in the lifecycle rule is reached.)	Objects can be transitioned to Infrequent Access or Archive.	If the version is the current version and is not a delete marker, it can be transitioned to Infrequent Access or Archive.	Same action as a versioning-enabled bucket.
Expiration (Performed when a date or time period specified in the lifecycle rule is reached.)	The **Expiration** action deletes the object, and the deleted object cannot be restored.	If the version is not a delete marker, this action inserts a delete marker that becomes the current version. The existing current version turns into a non-current version.	For buckets with versioning 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.

Action	Bucket with Versioning Not Enabled	Bucket with Versioning Enabled	Bucket with Versioning Suspended
NoncurrentVersionTransition (Performed when the object has become non-current for the specified number of days.)	The NoncurrentVersionTransition action has no effect.	If the version is not a delete marker or the current version, it can be transitioned to Infrequent Access or Archive.	Same action as a versioning-enabled bucket.
NoncurrentVersionExpiration (Performed when the object has become non-current for the specified number of days.)	The NoncurrentVersio-nExpiration action has no effect.	The NoncurrentVersio-nExpiration action deletes the non-current version of the object, and the deleted object cannot be recovered.	Same action as a versioning-enabled bucket.

Date-based lifecycle rules

You can specify the execution dates for **Transition** and **Expiration** actions. The dates must conform to the ISO 8601 standards and the exact time is always 00:00 (UTC time). If you specify the time to a past date, all applicable objects will be executed immediately.

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

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.

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 number of days is specified, OBS starts to calculate the time from 00:00 (UTC time) on the next day. For example, you created an object at 2016-01-15 10:30 (UTC time) and you specified that objects would be changed 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.

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

When you use the **NoncurrentVersionTransition** and **NoncurrentVersionExpiration** operations to specify the number of days, OBS adds the number of days specified in the rule to the time when the object version becomes a non-current version. Then the operation time is calculated starting from 00:00 (UTC time) of the next day. For example, the current version of an object in a bucket is created at 10:30 UTC on January 1, 2016. If the object version becomes a non-current version at 10:30 UTC on January 15, 2016 and you specify three days in the conversion rule, then the date for changing object storage class is calculated as 00:00 UTC on January 19, 2016.

NOTE

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 consistent.

How to Use

You can use OBS Console, APIs, or SDKs to configure lifecycle rules.

Tool	Reference
OBS Console	**Configuring a Lifecycle Rule**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding development guide on the **SDK Overview** page.
API	**Configuring Bucket Lifecycle Rules** **Obtaining Bucket Lifecycle Configuration** **Deleting Lifecycle Rules**

12 Cross-Region Replication

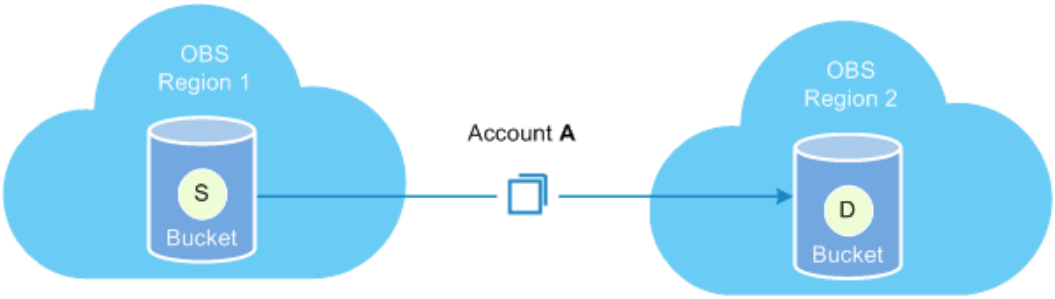
Scenarios

Cross-region replication provides the capability for disaster recovery across regions, allowing you to set up a remote backup solution.

Cross-region replication refers to the process of automatically and asynchronously replicating data from a bucket (source bucket) to another bucket (destination bucket) across regions by creating a cross-region replication rule. The source bucket and destination bucket must belong to the same account. Replication across accounts is not supported.

For a cross-region replication rule, you can configure it to match a pre-defined object prefix so that all objects with this prefix will be replicated. Alternatively, you can configure the rule to apply to the whole bucket so that all objects in the bucket will be replicated. Objects replicated to the destination bucket are precise copies of objects in the source bucket. They have the same object names, metadata, content, size, last modification time, creator, version ID, user-defined metadata, and ACLs. By default, the storage class of an object copy is the same as that of the source object. You can also specify a different storage class for an object copy.

Figure 12-1 Cross-region replication



- 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 regulatory compliance.

- Minimized latencies

  The same OBS resources may need to be accessed from different locations. To minimize the access latency, you can use cross-region replication to create object copies in the region nearest to end users.

- Data replication

  Cross-region replication allows you to easily migrate your data stored in OBS from one region to another.

- Data backup and disaster recovery

  To ensure data security and availability, you need to create explicit backups for all data written to OBS in the data center of another region, so that secure backup is available in case the source data is damaged irrevocably.

- 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.

NOTICE

OBS allows you to replicate the service data stored in OBS to a specified region, but Huawei Cloud does not detect the stored data and is not responsible for the legal compliance of your use of OBS. If your replication involves cross-border transfer, ensure that your use complies with relevant laws and regulations.

Content Replicated

With cross-region replication enabled, OBS will replicate the following objects to a destination bucket:

- Newly uploaded objects (excluding objects in the Archive storage class)
- Updated objects, for example, objects whose content or ACL is updated
- Historical objects in a bucket with **Synchronizing Existing Objects** enabled (excluding objects in the Archive storage class)

NOTE

Cross-region replication does not replicate objects encrypted using SSE-C.

Constraints

- A cross-region replication rule may do not take effect immediately upon its configuration. Accordingly, the objects that this rule is applied to may not be replicated immediately after the rule is configured.
- Currently, only buckets of version 3.0 support cross-region replication. To check the bucket version, go to the **Overview** page of the bucket on OBS Console. Then you can view the bucket version in the **Basic Information** area.
- By default, objects uploaded before cross-region replication is enabled are not copied to the destination bucket unless the function for synchronizing existing objects is enabled.

- The source bucket and the destination bucket must belong to different regions separately. Data cannot be copied between buckets in the same region.
- Objects cannot be copied from the source bucket to the destination bucket if they are in the Archive storage class.
- If the region where the destination bucket resides does not support different storage classes for data, object copies will be stored in the Standard storage class.
- The versioning status of the source bucket must be the same as that of the destination bucket.
- Objects in a source bucket can be copied to only one destination bucket, and cannot be copied again from the destination bucket to another bucket. For example, bucket A and bucket B are in two different regions. You can copy data from bucket A to bucket B or the other way round. However, data copies in either bucket A or bucket B cannot be replicated anymore.
- If versioning is enabled or suspended for both the source and destination buckets and cross-region replication is also enabled for both buckets, deleting an object without specifying its version in the source bucket will also delete the object in the destination bucket.
- If you change the versioning status of the destination bucket when cross-region replication is enabled, the replication of objects will fail. If you want to change the versioning status of the source bucket, disable the cross-region replication first, and then make the change.
- Ensure that owners of the source and destination buckets have the read and write permissions to the two buckets. Otherwise, data cannot be synchronized. If the system does not have the permissions to read the source bucket or write the destination bucket due to read/write permission errors, objects cannot be copied successfully, and such replication will not be resumed even if the permission error is rectified.
- For a source bucket, you can create only one cross-region replication rule that applies to the whole bucket for replication of all objects in the bucket. However, you can create a maximum of 100 cross-region replication rules based on object prefixes for the replication of objects that match the prefixes.
- OBS currently only supports the replication between one source bucket and one destination bucket. Replication from one source bucket to multiple destination buckets is not supported. The destination bucket can be modified. However, modifying the destination bucket will change the destination bucket of all existing rules.
- If you delete the OBS agency configuration in a cross-region replication, the replication status becomes **Failed**.
- Do not delete, overwrite object replicas in the destination bucket, or modify their ACLs, which may cause inconsistency of latest object versions or permission control settings between the destination bucket and the source bucket.
- If the function for synchronizing existing objects is enabled, modifying the cross-region replication configuration may cause failures in synchronizing existing objects. Therefore, do not modify the cross-region replication configuration before the synchronization finishes.
- If cross-region replication is enabled, data cannot be added to the end of objects in the source bucket.

- After a replication with **Synchronize Existing Objects** enabled is complete, if the replication policy keeps unchanged, any ACL changes of source objects will be synchronized to object copies. However, ACL changes of source historical objects will not be synchronized to the copies of historical objects.
- Before replicating data, ensure that source and destination regions can have their data replicated from each other. **Figure 12-2** lists the supported regions. √ indicates that data can be replicated between regions. x indicates that data cannot be replicated between regions.

Figure 12-2 Replication between regions

	CN North-Beijing4	CN East-Shanghai1	CN East-Shanghai2	CN North-Beijing1	CN North-Beijing2	CN North-Ulanqab1	CN Southwest-Guiyang1	CN South-Guangzhou	CN-Hong Kong	AP-Bangkok	LA-Sao Paulo1	AF-Johannesburg	LA-Mexico City1	AP-Singapore
CN North-Beijing4		√	√	√	√	√	√	√	x	x	x	x	x	x
CN East-Shanghai1	√		√	√	√	√	√	√	x	x	x	x	x	x
CN East-Shanghai2	√	√		√	√	√	√	√	√	x	x	√	x	x
CN North-Beijing1	√	√	√		√	√	√	√	x	x	x	√	x	x
CN North-Beijing2	√	√	√	√		√	√	√	x	x	x	x	x	x
CN North-Ulanqab1	√	√	√	√	√		√	√	x	x	x	x	x	x
CN Southwest-Guiyang1	√	√	√	√	√	√		√	x	x	x	x	x	x
CN South-Guangzhou	√	√	√	√	√	√	√		x	x	x	x	x	x
CN-Hong Kong	x	x	√	x	x	x	x	x		x	x	√	√	x
AP-Bangkok	x	x	x	x	x	x	x	x	x		x	x	x	x
LA-Sao Paulo1	x	x	x	x	x	x	x	x	x	x		x	x	x
AF-Johannesburg	x	x	√	√	x	x	x	x	√	x	x		x	x
LA-Mexico City1	x	x	x	x	x	x	x	x	√	x	x	x		x
AP-Singapore	x	x	x	x	x	x	x	x	x	x	x	x	x	

How to Use

You can use OBS Console, APIs, SDKs, or obsutil to configure cross-region replication.

Tool	Reference
OBS Console	**Configuring Cross-Region Replication**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Cross-Region Replication for a Bucket**
obsutil	**Copying Objects**

13 Server-Side Encryption

Scenarios

After server-side encryption is enabled, objects uploaded to OBS will be encrypted and then stored on the server. When you download the encrypted objects, the encrypted data will be decrypted on the server and displayed in plaintext to you.

OBS provides the following server-side encryption methods that adopt the 256-bit Advanced Encryption Standard (AES-256).

- Server-side encryption using keys hosted by KMS (SSE-KMS)

  You need to create a key using Key Management Service (KMS) or use the default key provided by KMS. The KMS key is then used to perform server-side encryption when you upload objects to OBS.

  In the SSE-KMS mode, you can enable the default encryption when creating a bucket. Then all objects uploaded to the bucket are encrypted. You can also enable the default encryption for a bucket after its creation, and then newly uploaded objects are encrypted.

  OBS encrypts only the objects uploaded after the default encryption function is enabled. The encryption status of existing objects in the bucket remains unchanged. Disabling default encryption does not change the encryption status of existing objects in a bucket. After this function is disabled, you can still manually encrypt objects upon upload.

  You can use OBS Console, APIs, SDKs, or OBS Browser+ to configure SSE-KMS.

- Server-side encryption with customer-provided keys (SSE-C)

  OBS uses the keys and MD5 values provided by customers for server-side encryption.

  You can use APIs or SDKs to configure SSE-C.

Background Information

In SSE-KMS mode, KMS uses a hardware security module (HSM) to protect key security, helping you easily create and control encryption keys. Keys are not displayed in plaintext outside HSMs, which prevents key disclosure. All operations performed on keys are controlled using access permissions and logged, meeting regulatory compliance requirements.

Precautions

When server-side encryption is disabled for a bucket, the encrypted objects must be accessed over HTTPS.

How to Use

You can use OBS Console, APIs, SDKs, or OBS Browser+ to configure server-side encryption.

Tool	Reference
OBS Console	**Uploading a File in Server-Side Encryption Mode** **Configuring Bucket Default Encryption**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Server-Side Encryption (SSE-KMS)** **Server-Side Encryption (SSE-C)** **Configuring Bucket Encryption**
OBS Browser+	-

14 WORM

OBS provides write-once-read-many (WORM) to protect objects from being deleted or tampered with within a specified period. WORM works at both the bucket and object levels in compliance mode.

Scenarios

In compliance mode, a WORM-protected object version cannot be overwritten or deleted by anyone, including the root user in your account.

When WORM is configured for a bucket, the protection applies to all objects in the bucket. When WORM is configured for an object version, the protection applies to the current object version only. No matter which type of WORM protection you want to use, you must enable WORM for the bucket first. A WORM retention policy takes effect only for objects uploaded after the policy takes effect. If an object is protected by a bucket-level WORM policy and an object-level WORM policy at the same time, the object-level WORM policy takes precedence.

Precautions

- When you enable WORM for a bucket, OBS automatically enables versioning and versioning cannot be suspended later for that bucket. WORM protects objects based on the object version IDs. Only object versions with any WORM retention policy configured can be protected. Assume that object **test.txt 001** is protected by WORM. If another file with the same name is uploaded, a new object version **test.txt 002** with no WORM policy configured will be generated. In such case, **test.txt 002** is not protected and can be deleted. When you download an object without specifying a version ID, the current object version (**test.txt 002**) will be downloaded.
- A lifecycle rule cannot delete WORM-protected objects, but can transition their storage class. After an object is no longer protected, it will be deleted when meeting the expiration rule in a lifecycle configuration.
- Once you enable WORM for a bucket, you cannot disable it or suspend versioning for the bucket, but you can disable the default WORM policy for the bucket.
- Buckets with WORM enabled do not support cross-region replication.
- If you have deregistered your account or your account has been frozen, the WORM-protected objects will be permanently deleted.

- WORM-based protection is not available for migration.
- The metadata of a WORM-protected object can still be modified.

How to Use

You can perform WORM-related operations using OBS Console and OBS API.

Tool	Reference
OBS Console	**Configuring WORM Retention**
API	**Configuring a Default WORM Policy for a Bucket** **Obtaining the Default WORM Policy of a Bucket** **Configuring WORM Retention for an Object** Obtain the object-level WORM retention configuration by referring to **Querying Object Metadata**.

15 Static Website Hosting

Precautions

For security and compliance purposes, using static website hosting through the default OBS domain name (**bucket domain name or static website domain name**) will be prohibited by OBS. When you use such domain name to access web pages through a browser, no content will be displayed, instead, the content is downloaded as an attachment.

This prohibition will take effect in different regions at the following two points in time:

January 1, 2022: CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, and CN South-Guangzhou

March 25, 2022: CN-Hong Kong, AP-Bangkok, AP-Singapore, AF-Johannesburg, LA-Mexico City1, LA-Mexico City2, LA-Sao Paulo1, and LA-Santiago

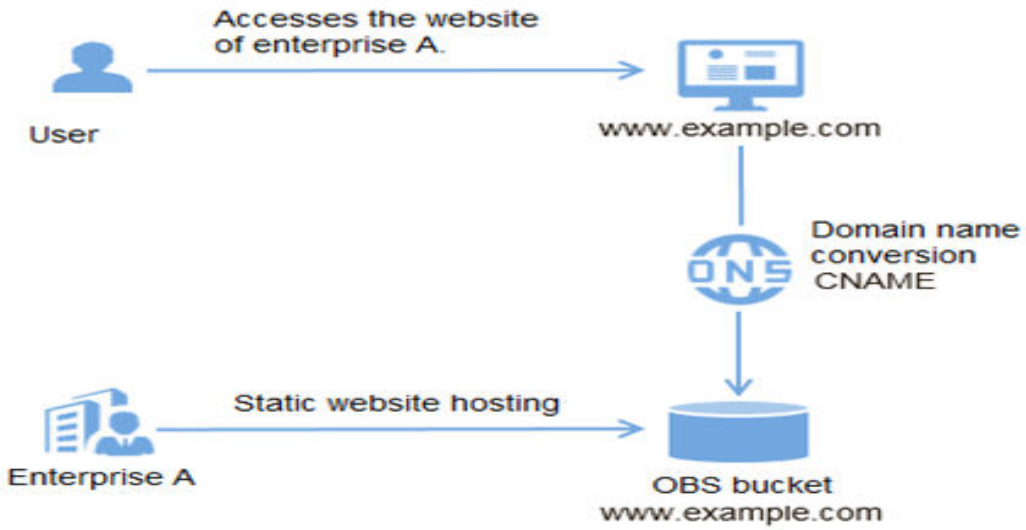
However, you can still use static website hosting through a user-defined domain name and can preview the web page content. For details, see **How Do I Preview Objects in OBS Through a Browser?**

Scenarios

You can use OBS to host your static website, for which you can also configure an index document, error document, and page direction. You can upload the content files of the static website to your bucket in OBS and configure a read permission to anonymous users for these files, and then configure the static website hosting mode for your bucket to host your static websites in OBS.

Static websites contain static web pages and some scripts that can run on clients, such as JavaScript and Flash. Different from static websites, dynamic websites rely on servers to process scripts, including PHP, JSP, and ASP.NET. OBS does not support scripts that run on servers.

Figure 15-1 Static website hosting



After static website hosting is enabled, you can access objects in a bucket using either of the following methods:

- Resource management: You can access objects in a bucket through a common domain name. You can use APIs or SDKs and default OBS domain names with endpoints contained to perform common operations on buckets and objects, such as upload, download, deletion, and listing.
- Static website: You 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 configuration of static website hosting takes effect within two minutes. After it takes effect, you can access static resources using the following access domain names:

https://*bucketname.OBS static website hosting domain name/Object name*
http://*bucketname.OBS 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 permissions.

The following table lists the differences between the resource management mode and static website mode.

Major Difference	Resource Management	Static Website
Access control	Both public content and private content are supported.	Only public content is supported.

Major Difference	Resource Management	Static Website
Error message processing	An error response in XML format is returned.	An HTML document is returned.
Redirection support	N/A	Both object-level and bucket-level redirection are supported.
Supported request	Operations on all buckets and objects are supported.	Only GET and HEAD requests on objects are supported.
Response to GET and HEAD requests of bucket root level	List of object keys in a bucket is returned.	Index file specified in the configuration is returned.

Redirection Overview

When using static website hosting, you can configure redirection to redirect specific or all requests.

If the structure, address, or file name extension of a website is changed, users will fail to access the website using the old address (such as the address saved in the folder of favorites), and the 404 error message will be returned. In this case, you can configure redirection for the website to redirect user access requests to the specified page instead of returning the 404 error page.

Typical configurations include:

- Redirecting all requests to another website.
- Redirecting specific requests based on redirection rules.

Configuring Static Website Hosting

Overview

If you want to use a bucket to host static websites, add the website 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 provide your own customized error document, or 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:

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

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

```
<RoutingRules> =
   <RoutingRules>
      <RoutingRule>...</RoutingRule>
      [<RoutingRule>...</RoutingRule> ...]
   </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith> ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ][
      <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>
```

For request elements required for redirecting all requests sent to a specified website and for setting redirection rules, see **Configuring Static Website Hosting for a Bucket – Request Elements**

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:

```
<RoutingRules>
  <RoutingRule>
  <Condition>
    <KeyPrefixEquals>docs/</KeyPrefixEquals>
  </Condition>
  <Redirect>
    <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

Example 2: Redirecting requests sent to 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**.

```
<RoutingRules>
  <RoutingRule>
  <Condition>
    <KeyPrefixEquals>images/</KeyPrefixEquals>
  </Condition>
  <Redirect>
    <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

Example 3: Redirecting when HTTP errors occur

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.

```
<RoutingRules>
  <RoutingRule>
  <Condition>
    <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
  </Condition>
  <Redirect>
    <HostName>www.example.com</HostName>
    <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
```



```
  </Redirect>
  </RoutingRule>
</RoutingRules>
```

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, **index.html** will be returned to any of the following URLs:

```
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

Table 15-1 lists the subset of the HTTP response code returned by OBS when an error occurs.

Table 15-1 List of OBS error codes

HTTP error code	Description
301 Moved Permanently	When a user sends a request to an endpoint of OBS, a **301 Moved Permanently** response is returned.

HTTP error code	Description
302 Found	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**.
304 Not Modified	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.
403 Forbidden	When the user request is converted into an object that cannot be publicly read, the response of the website terminal node includes **403 Forbidden**. The object owner must use bucket policies or ACLs to make the object public and accessible.
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. ● The 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.

HTTP error code	Description
500 Service Error	When an internal server error occurs, the response of the website terminal node contains **500 Service Error**.
503 Service Unavailable	When 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**. You 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.

NOTE

In the static website hosting scenario, anonymous users must be granted access to the hosted static website file. During their access to the hosted file, fees for outbound internet traffic and requests will be incurred.

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

```
{
  "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 the read permission to everyone, as shown in the following authorization elements.

You can add the authorization element to object ACLs. For details about ACL management, see **ACLs**.

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

Related Functions

Function	Relationship Description	Reference
CORS	By default, static websites hosted in an OBS bucket can only respond to access requests from websites in the same domain. You can configure cross-origin resource sharing (CORS) for the bucket so that the static websites can be accessed by cross-domain requests.	**CORS**
User-Defined Domain Name Binding	OBS allows you to use a user-defined domain name to access static websites hosted in an OBS bucket. You can use the original domain name to access the website without modifying the website code.	**User-Defined Domain Name Configuration** **Using a User-Defined Domain Name to Host a Static Website**
URL Validation	OBS uses URL validation to prevent your websites hosted in an OBS bucket being stolen. OBS verifies URLs based on the referer field in the HTTP header.	**URL Validation**

How to Use

You can use OBS Console, APIs, or SDKs to configure CORS.

Tool	Reference
OBS Console	**Configuring Static Website Hosting**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Static Website Hosting for a Bucket**

16 CORS

Cross-origin resource sharing (CORS) is a browser-standard mechanism provided by the World Wide Web Consortium (W3C). It defines the interaction methods between client-side web applications in one origin and resources in another. For general web page requests, website scripts and contents in one origin cannot interact with those in another because of Same Origin Policies (SOPs).

OBS supports CORS rules and allows resources in OBS to be accessed across origins. The configuration of CORS takes effect within two minutes.

OBS supports **Static Website Hosting**. Static websites stored in OBS can respond to website requests from another origin only when CORS is configured for the bucket.

NOTICE

By default, the OBS system is configured to support cross-domain access using the root domain name. This allows access from all domains, and clients are likely to be attacked.

To address this issue, you can create a **crossdomain.xml** file with specific rules in the bucket for each client, and add **Security.loadPolicyFile("https://bucket.obs.ap-southeast-1.myhuaweicloud.com/crossdomain.xml")** in the flash code of the file to prevent attacks. **bucket.obs.ap-southeast-1.myhuaweicloud.com** needs to be replaced with the actual access domain name of your bucket.

Background Information

Cross-domain refers to access between different domains.

Restricting cross-domain access is a browser policy for security purposes, that is, the same-origin policy. Due to this JavaScript same-origin policy, JavaScript under domain A cannot operate objects under domain B or C.

The same protocol, domain name (or IP address), and port are considered as the same domain. If the protocols, domain names, and ports (if specified) of the two web pages are the same, the two web pages have the same origin. To better

understand the same-origin policy, you can see the analysis on accessing the example address **https://support.huaweicloud.com/dir/test.html** in Table 16-1.

Table 16-1 Example analysis

URL	Access Result	Cause
https://support.huaweicloud.com/dir/other.html	Successful	Same protocol, domain name, and port
https://support.huaweicloud.com/dir/inner/other.html	Successful	Same protocol, domain name, and port
http://support.huaweicloud.com/dir/test.html	Failed	Same domain name and port, but different protocols
https://support.huaweicloud.com:81/dir/test.html	Failed	Same protocol and domain name, but different ports
https://help.huaweicloud.com/dir/test.html	Failed	Same protocol and port, but different domain names

Scenarios

Typical application scenarios of CORS are as follows:

- Enables JavaScript and HTML5 to be used for establishing web applications that can directly access resources in OBS. No proxy servers are required for transfer.
- Enables the dragging function of HTML5 to be used to upload files to OBS (with the upload progress displayed) or update OBS contents using web applications.
- External web pages, style sheets, and HTML5 applications hosted in different origins can access web fonts or pictures stored in OBS, implementing resource sharing.

How to Use

You can configure CORS using OBS Console, APIs, or SDKs.

Tool	Reference
OBS Console	**Configuring CORS**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Bucket CORS**

17 User-Defined Domain Name Configuration

Scenarios

If you want to migrate files from a website to OBS while keeping the website address unchanged, you can bind the website domain name to an OBS bucket, so that you can still use the website address to access the files stored in the bucket.

Assume the domain name of your website is **www.example.com** and the file you want to migrate to OBS is **abc.html**. After the file is migrated to OBS, you can use **http://www.example.com/abc.html** to access it. The steps below describe the configurations:

1. Create a bucket on OBS, and upload **abc.html** to the bucket.
2. On OBS Console, bind the domain name **www.example.com** to the created bucket.
3. On the DNS server, add **a CNAME rule** and map **www.example.com** to the domain name of the bucket.
4. Access the **abc.html** file. After the request for **http://www.example.com/abc.html** reaches OBS, OBS finds the mapping between the **www.example.com** and the bucket domain name, and redirects the request to the **abc.html** file stored in the bucket. The essence of this process is that OBS redirects the request to access **http://www.example.com/abc.html** to **http://***bucket domain name***/abc.html**.

Constraints

- Only buckets with version 3.0 or later support user-defined domain name configuration. To check the bucket version, go to the **Overview** page of the bucket on OBS Console. Then you can view the bucket version in the **Basic Information** area.
- By default, a bucket can have up to 20 user-defined domain names bound.
- User-defined domain names currently allow requests over only HTTP, but not HTTPS.

  If you want to use a bound domain name to access OBS over HTTPS, you need to enable CDN to manage HTTPS certificates.

For details about how to manage HTTPS certificates on the CDN management console, see **HTTPS Settings**.

- A user-defined domain name can be bound to only one bucket.
- Chinese domain names are not supported.
- The suffix of a user-defined domain name can contain 2 to 6 uppercase or lowercase letters.
- As required by the MIIT, you must complete the ICP filing, if the bucket which your domain name is bound to is in any of the following regions:

  CN North-Beijing1, CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, and CN South-Guangzhou

NOTE

If an acceleration domain name is also needed, to prevent objects in OBS buckets from being directly downloaded upon access, you need to perform other required operations after the custom domain name and the acceleration domain name have been configured. For details, see **With CDN Acceleration Enabled, Why Are the Objects in My OBS Bucket Directly Downloaded When I Access Them?**

How to Use

You can configure user-defined domain name binding using OBS Console, APIs, or SDKs.

Tool	Reference
OBS Console	**Configuring User-Defined Domain Names**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring a Custom Domain Name for a Bucket**

18 Back to Source

When a client does not access the requested data in OBS, the 404 error is returned. However, OBS provides the back-to-source function to help you obtain the requested data from its source site if it is not found in OBS.

Back-to-Source by Mirroring

If a mirroring back-to-source rule is configured for an OBS bucket and the requested data is not found in the bucket, the system will retrieve the data, when the back-to-source rule applies to the data, from the origin server, upload it to the bucket, and then return it to the requesting client. This process does not interrupt services. Therefore, you can use this function to seamlessly migrate data from the origin server to OBS, or migrate services to OBS without being sensed by users, at low costs. **Figure 18-1** illustrates the mirroring back-to-source process.

Figure 18-1 Back-to-source by mirroring

Constraints

- This feature is currently available only in the CN North-Beijing4, CN East-Shanghai1, CN East-Shanghai2, CN-Hong Kong, AP-Singapore, and TR-Istanbul regions.
- Anonymous users cannot configure mirroring back-to-source rules for a bucket.
- Parallel file systems do not support mirroring back-to-source rules.
- A mirroring back-to-source rule is not compatible with the static website hosting function. Specifically, if a 404 error occurs when objects are downloaded from an OBS hosted static website domain, it does not trigger the mirroring back-to-source process.

- The bucket, to which a back-to-source rule is configured, cannot be specified as the source site.
- Currently, mirroring back to source from private buckets is supported for only some cloud vendors.
- The origin server cannot transfer data in **Transfer-Encoding: chunked** mode. That is, the response to the request for downloading an object from the origin server must contain the **Content-Length** header to specify the size of the source object.
- An object cannot match two different mirroring back-to-source rules.
- Only buckets of version 3.0 or later support the mirroring back-to-source function.
- A mirroring back-to-source rule takes effect five minutes later after any change to the rule.
- A maximum of 10 mirroring back-to-source rules can be configured for a bucket.
- The mirroring back-to-source function is offered for free.

Creating a mirroring back-to-source rule

You can create mirroring back-to-source rules on OBS Console or by calling APIs.

Tool	Reference
OBS Console	**Creating a Back-to-Source Rule**
API	**Configuring Mirroring Back-to-Source Rules**

19 URL Validation

Scenarios

Some rogue sites may steal links from other sites to enrich their content without any costs. Link stealing hurts the interests of the original websites and it is also a strain on their servers. OBS provides URL validation to solve this problem.

In HTTP, the **Referer** field allows websites and web servers to identify where people are visiting them from. URL validation of OBS utilizes this **Referer** field. The idea is that once you find that a request to your resource is not originated from an authorized source (for example, a URL), you can have the request blocked or redirected to a specific web page. This way, OBS prevents unauthorized access to data stored in buckets.

Such authorization is controlled using a whitelist and a blacklist.

Referer setting rules:

- The length of a whitelist or blacklist cannot exceed 1,024 characters.
- Referer format:
  - You can enter multiple referers, each in a line.
  - The referer parameter supports asterisks (*) and question marks (?). An asterisk works as a wildcard that can replace zero or multiple characters, and a question mark (?) can replace a single character.
  - If the referer header field contains **http** or **https** during download, the referer must contain **http** or **https**.
- If there are websites configured in the blacklist referer, but no websites in the whitelist referer, all websites except those on the blacklist are allowed to access the target bucket.
- If there are websites configured in the whitelist referer, only the websites on the whitelist but not on the blacklist are allowed to access the target bucket, regardless of whether there are websites configured in the blacklist referer or not.

NOTE

If a website is configured in both the whitelist and blacklist referers, access from this website will be forbidden. For example, if **https://www.example.com** is configured in both **Whitelisted Referers** and **Blacklisted Referers**, access requests from **https://www.example.com** will be blocked.

- If **Whitelisted Referers** and **Blacklisted Referers** are both left blank, all websites are allowed to access data in the target bucket by default.
- Before determining whether a user has the four types of permissions (**Read**, **Write**, **ACL View**, and **ACL Edit**) for a bucket or objects in the bucket, check whether this user complies with the URL validation principles of the **Referer** field.

Whitelist and blacklist setting methods:

- Whitelist settings

  By setting a whitelist, you can allow requests from the websites in the whitelist, but deny those from the websites that are not in the whitelist.

  For the requests that are initialized from a browser's address box, you can add the **${null}** field to **Referer** of **Condition** to specify whether to allow the HTTP requests with a blank **referer**.

  To configure a whitelist, refer to the following policy:

```
"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, only the requests whose **referer** is set to **www.example01.com** or left blank can work on resources in buckets.

- Blacklist settings

  To configure a blacklist, refer to the following policy:

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

  If you set a blacklist in this way, requests whose **referer** is set to **www.example01.com** or **www.example02.com** cannot work on resources in buckets.

How to Use

You can use OBS Console or APIs to configure URL validation.

Tool	Reference
OBS Console	**Configuring URL Validation**
API	**Configuring a URL Validation Whitelist**

20 Tags

Scenarios

If your service system uses multiple cloud services from Huawei Cloud, you can set tags to better identify resource instances for different cloud services. For OBS, a resource instance is a bucket. These tags will be included in the service detail records (SDRs) generated for those services and their instances. If your service system is composed of multiple applications, setting the same tag for all resource instances used for each application helps you easily analyze resource usage and costs.

In OBS, tags are used to identify and classify buckets. If you add tags to a bucket, SDRs generated for the requests sent to this bucket will include these tags, so you can use the tags to classify SDRs for detailed cost analysis. For example, if you have an application that uploads runtime data to a bucket, you can tag the bucket with the application name. In this manner, the costs on the application can be analyzed using tags in SDRs.

A tag is described using a key-value pair. A bucket can have a maximum of 10 tags. Each tag has only one key and one value. The key and value can exist in either sequence in a tag. Each key is unique among all tags of a bucket, whereas values can be repetitive or blank. It takes approximately three minutes for the tag to take effect.

How to Use

You can configure tags using OBS Console, APIs, or SDKs.

Tool	Reference
OBS Console	**Configuring Tags for a Bucket**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Tags for a Bucket**

21 Bucket Inventory

Scenarios

A bucket inventory can list objects in a bucket, save the related object information in CSV files, and deliver the CSV files to the bucket specified for storing bucket inventory files. In this manner, you can easily manage objects in a bucket. A source bucket can also be the destination bucket.

- A bucket inventory file can contain the following object related information: versions, sizes, storage classes, tags, encryption statuses, and last modification time.
- You can encrypt bucket inventory files in the SSE-KMS mode.
- You can set the frequency (daily or weekly) for generating bucket inventory files.
- You can also specify a bucket to store the generated bucket inventory files.

Constraints

- Bucket inventories can be generated only for OBS 3.0 buckets, but they can be stored in either OBS 3.0 or OBS 2.0.
- A bucket can have a maximum of 10 inventory rules.
- The source bucket (for which a bucket inventory rule is configured) and the destination bucket (where the generated inventory files are stored) must belong to the same account.
- The source bucket and the destination bucket must be in the same region.
- Inventory files must be in the CSV format.
- OBS can generate inventory files for all objects in a bucket or a group of objects whose names begin with the same prefix.
- If a bucket has multiple inventory rules, these rules must not overlap.
  - If a bucket already has an inventory rule for the entire bucket, new inventory rules that filter objects by prefixes cannot be created. If you need an inventory rule that covers only a subset of objects in the bucket, first delete the inventory rule configured for the entire bucket.
  - If an inventory rule that filters objects by a specified prefix already exists, you cannot create an inventory rule for the entire bucket. To create an

inventory rule for the entire bucket, make sure that the bucket has no other inventory rules that filter objects by specified prefixes.

- If a bucket already has an inventory rule that filters objects by the object name prefix **ab**, the filter of a new inventory rule cannot start with **a** or **abc**. Or, you can delete the existing inventory rule and create a new one that filters objects according to your needs.

- Bucket inventory files can be encrypted only in the SSE-KMS mode.
- The bucket inventory function is offered for free, but inventory files are billed for the storage space they use.
- Default encryption cannot be enabled for the destination bucket configured for storing 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 Is a Bucket Inventory Configured?

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

- 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.

   ```
   {
      "Statement": [
         {
            "Effect": "Allow",
            "Sid": "1",
            "Principal": {"Service": "obs"},
                     "Resource": ["destbucket/*"],
            "Action": ["PutObject"]
         }
      ]
   }
   ```

2. Configure a bucket inventory.

   We provide multiple tools to configure a bucket inventory. For details, see **How to Use**.

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 21-1**.

Table 21-1 Object metadata listed in an inventory file

Metadata	Description
Bucket	Name of the source bucket
Key	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.)
VersionId	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.
IsLatest	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.)
IsDeleteMarker	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.)
Size	Object size, in bytes.
LastModifiedDate	Object creation date or last modification date

Metadata	Description
ETag	Hexadecimal digest of the object MD5. 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.
StorageClass	Storage class of an object
IsMultipartUploaded	Indicates whether an object is uploaded in the multipart mode.
ReplicationStatus	Cross-region replication status of an object
EncryptionStatus	Encryption status of an object

Inventory File Name

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

```
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 there are a large number of objects in a bucket, multiple inventory files may be generated for a single inventory configuration. It takes some time to generate these files. For example, if there are 200,000 objects in a bucket, it takes about 1.5 minutes to generate all inventory files. One or two hours after all inventory files are generated, a **manifest.json** file will be generated. The **manifest.json** file contains information about all inventory files generated this time, including:

- **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.

```
{
	"sourceBucket":"user001",
	"destinationBucket":"bucket001",
	"version":"2019-01-03",
	"fileFormat":"CSV",

"fileSchema":"Bucket,Key,Size,LastModifiedDate,ETag,StorageClass,IsMultipartUploaded,ReplicationStatus,En
cryptionStatus",
	"files":[
		{
			"key":"inventory/user001/test_id/2019-01-03T12-28Z/files/
0000016813AF58E66806C1E2D7F15155_1.csv",
			"size":6705647390,
			"inventoriedRecord":70585762,
		}
	]
}
```

The name of a **manifest** file is as follows (for details about each field, see **Inventory File Name**):

```
destinationPrefix/sourceBucketName/inventoryId/yyyy-MM-dd'T'HH-mm'Z'/manifest.json
```

How to Use

You can configure bucket inventories using OBS Console, APIs, or SDKs.

Tool	Reference
OBS Console	**Configuring a Bucket Inventory**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring a Bucket Inventory**

22 Event Notifications

NOTE

Event notifications have been taken offline. If you have any questions, **submit a service ticket**.

23 Logging

Scenarios

You can enable logging to facilitate analysis or audit as required. Access logs enable a bucket owner to analyze the property, type, or trend of requests to the bucket in depth. With logging enabled, OBS automatically logs access requests for the bucket, and writes the generated log files to the specified bucket.

You need to specify a bucket for storing log files when enabling logging for a bucket. Log files can be stored in any bucket in the region where the logged bucket resides, including the logged bucket itself. To better manage logs, you are advised to store log files in a bucket other than the logged bucket. If log files are stored in the logged bucket, OBS creates additional logs for writing log files to the bucket, which takes up extra storage space that will increase your costs and makes it more difficult for you to locate required logs.

NOTICE

Uploading bucket logs to the target bucket incurs billable PUT requests. For details about the pricing, see **Requests**.

OBS can record bucket access requests in logs for request analysis and log audit.

Logs occupy some OBS storage space rented by users, incurring extra fees. For this reason, the default policy is that OBS does not collect bucket access logs.

The log files are generated and uploaded by OBS to the bucket where the logs are stored. Therefore, OBS requires the authorization to upload the generated log files. Therefore, before configuring logging for a bucket, you need to create an IAM agency for OBS and add this IAM agency when configuring logging for the bucket. By default, when configuring permissions for an IAM agency, you only need to grant the IAM agency the permission to upload log files to the bucket where log files are stored. In the following example, **mybucketlogs** is the name of the bucket for storing log files. If the default encryption function is enabled for the log storing bucket, the IAM agency also requires the KMS Administrator permissions in the region where the log storing bucket resides.

```
{
    "Version": "1.1",
```



```
  "Statement": [
    {
      "Action": [
        "obs:object:PutObject"
      ],
      "Resource": [
        "OBS:*:*:object:mybucketlogs/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

After logging is configured, you can view operation logs in the bucket that stores the logs in approximately fifteen minutes.

The following shows a sample access log record:

```
787f2f92b20943998a4fe2ab75eb09b8 bucket [13/Aug/2015:01:43:42 +0000] xx.xx.xx.xx
787f2f92b20943998a4fe2ab75eb09b8 281599BACAD9376ECE141B842B94535B
REST.GET.BUCKET.LOCATION
- "GET /bucket?location HTTP/1.1" 200 - 211 - 6 6 "-"  "HttpClient" - -
```

The access log of each bucket contains the following information.

Table 23-1 Format of bucket access logs

Name	Example	Description
BucketOwner	787f2f92b20943998a4fe2ab75eb09b8	Account ID of the bucket owner
Bucket	bucket	Name of the bucket
Time	[13/Aug/2015:01:43:42 +0000]	Timestamp of the request (UTC)
Remote IP	xx.xx.xx.xx	Request IP address
Requester	787f2f92b20943998a4fe2ab75eb09b8	Requester ID • When an account initiates a request, this is the account ID. When an IAM user initiates a request, this is the ID of the account to which the IAM user belongs. • When a request is initiated by an anonymous user, the value of this parameter is **Anonymous**.
RequestID	281599BACAD9376ECE141B842B94535B	Request ID

Name	Example	Description
Operation	REST.GET.BUCKET.LOCATION	Name of the operation See **Table 23-2** for common operations and their description.
Key	-	Object name
Request-URI	GET /bucket?location HTTP/1.1	Request URI
HTTPStatus	200	Response code
ErrorCode	-	Error code
BytesSent	211	Size of the HTTP response, expressed in bytes
ObjectSize	-	Object size (bytes) **NOTE** When OBS deletes an object, it does not log the size of the deleted object. In the object deletion log, the value of **ObjectSize** is **0**.
TotalTime	6	Processing time on the server (ms)
Turn-AroundTime	6	Total time for processing the request (ms) **NOTE** This parameter can also be written as **TotalTime**.
Referer	-	Referer header of the request
User-Agent	HttpClient	User-Agent header of the request
VersionID	-	Version ID contained in a request
STSLogUrn	-	Federated authentication and agency information
StorageClass	STANDARD_IA	Current storage class of the object
TargetStorageClass	GLACIER	Storage class that the object will be transited to

Name	Example	Description
DentryName	12456/file.txt	• For a parallel file system, this field indicates an internal identifier of a file or directory. Its value consists of a parent directory inode number and a file or directory name. • For a bucket, the value of this field is **-**.

Table 23-2 Common operations

Operation	Description	Operation	Description
REST.GET.SERVICE	Lists buckets.	REST.GET.ENCRYPTION	Obtains the bucket encryption configuration.
REST.PUT.BUCKET	Creates a bucket.	REST.DELETE.ENCRYPTION	Deletes the bucket encryption configuration.
REST.HEAD.BUCKET	Views the bucket information.	REST.PUT.OTM_DIRECT_COLD_ACCESS	Configures direct reading for Archive objects in a bucket.
REST.GET.BUCKETVERSIONS	Lists objects in a bucket.	REST.GET.OTM_DIRECT_COLD_ACCESS	Obtains the direct reading configuration of a bucket.
REST.GET.BUCKET	Obtains the bucket metadata.	REST.DELETE.OTM_DIRECT_COLD_ACCESS	Deletes the direct reading configuration of a bucket.
REST.GET.BUCKET.LOCATION	Obtains the bucket location.	REST.PUT.BUCKET.WEBSITE	Configures static website hosting for a bucket.
REST.DELETE.BUCKET	Deletes a bucket.	REST.GET.BUCKET.WEBSITE	Obtains the static website hosting configuration of a bucket.

Operation	Description	Operation	Description
REST.PUT.POLICY	Configures a bucket policy.	REST.DEL.BUCKET.WEBSITE	Deletes the static website hosting configuration of a bucket.
REST.GET.POLICY	Obtains a bucket policy.	REST.PUT.BUCKET.CORS	Configures CORS for a bucket.
REST.DELETE.POLICY	Deletes a bucket policy.	REST.GET.BUCKET.CORS	Obtains the CORS configuration of a bucket.
REST.PUT.ACL	Configures an ACL for a bucket or an object.	REST.DEL.BUCKET.CORS	Deletes the CORS configuration of a bucket.
REST.GET.ACL	Obtains a bucket ACL or an object ACL.	REST.PUT.BUCKET.REQUEST.PAYMENT	Configures Requester Pays for a bucket.
REST.PUT.LOGGING_STATUS	Configures logging for a bucket.	REST.GET.BUCKET.REQUEST.PAYMENT	Queries the Requester Pays configuration of a bucket.
REST.GET.LOGGING_STATUS	Obtains the bucket logging configuration.	REST.OPTIONS.BUCKET	Checks bucket OPTIONS.
REST.PUT.BUCKET.LIFECYCLE	Configures a bucket lifecycle rule.	REST.OPTIONS.OBJECT	Checks object OPTIONS.
REST.GET.LIFECYCLE	Obtains the lifecycle configuration of a bucket.	REST.PUT.OBJECT	Uploads an object with PUT.
REST.DEL.LIFECYCLE	Deletes the lifecycle configuration of a bucket.	REST.POST.OBJECT	Uploads an object with POST.
REST.PUT.VERSIONING	Configures versioning for a bucket.	REST.COPY.OBJECT	Copies an object.
REST.GET.VERSIONING	Obtains the bucket versioning status.	REST.GET.OBJECT	Obtains the object content.
REST.GET.BUCKET.STORAGE.POLICY	Configures the default storage class for a bucket.	REST.HEAD.OBJECT	Obtains the object metadata.

Operation	Description	Operation	Description
REST.PUT.BUCKET.STORAGE.POLICY	Obtains the default storage class of a bucket.	REST.DELETE.OBJECT	Deletes an object.
REST.PUT.REPLICATION	Configures cross-region replication for a bucket.	REST.TRANSITION.STORAGECLASS.OBJECT	Changes the storage class of an object.
REST.DELETE.REPLICATION	Deletes the cross-region replication configuration of a bucket.	OP_MULTIPLE_DELETEOBJECT	Batch deletes objects.
REST.GET.REPLICATION	Obtains the cross-region replication configuration of a bucket.	REST.POST.RESTORE	Restores an Archive object.
REST.PUT.TAGGING	Configures tags for a bucket.	REST.APPEND.OBJECT	Appends data to an object.
REST.GET.TAGGING	Obtains bucket tags.	REST.MODIFY.OBJECT.META	Modifies object metadata.
REST.DEL.TAGGING	Deletes bucket tags.	REST.TRUNCATE.OBJECT	Truncates an object.
REST.PUT.BUCKET_QUOTA	Configures a storage quota for a bucket.	REST.RENAME.OBJECT	Renames an object.
REST.GET.BUCKET.QUOTA	Queries the bucket storage quota.	REST.GET.UPLOADS	Lists the initiated multipart uploads in a bucket.
REST.GET.BUCKET.STORAGEINFO	Obtains the information about the used space in a bucket.	REST.POST.UPLOADS	Initiates a multipart upload.
REST.PUT.BUCKET.INVENTORY	Configures inventories for a bucket.	REST.PUT.PART	Uploads a part.
REST.GET.BUCKET.INVENTORY	Obtains or lists bucket inventories.	REST.COPY.PART	Copies a part.
REST.DELETE.BUCKET.INVENTORY	Deletes bucket inventories.	REST.GET.UPLOAD	Lists uploaded parts.

Operation	Description	Operation	Description
REST.PUT.CUSTOMDOMAIN	Configures a custom domain name for a bucket.	REST.POST.UPLOAD	Assembles parts.
REST.GET.CUSTOMDOMAIN	Obtains the custom domain name of a bucket.	REST.DELETE.UPLOAD	Cancels a multipart upload.
REST.DELETE.CUSTOMDOMAIN	Deletes a custom domain name of a bucket.	REST.CLEAR.EXPIRE.UPLOAD	Deletes expired segments.
REST.PUT.ENCRYPTION	Configures encryption for a bucket.	-	-

How to Use

You can configure logging on OBS Console, using APIs, or using SDKs.

Tool	Reference
OBS Console	**Configuring Access Logging for a Bucket**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Logging for a Bucket**

24 Versioning

Scenarios

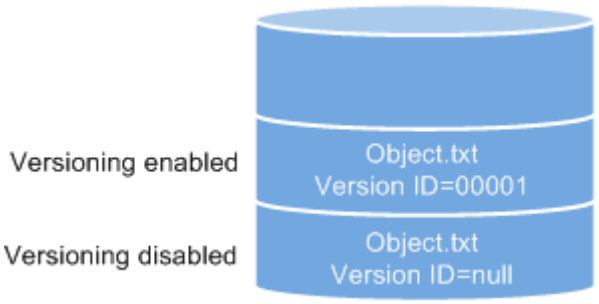
OBS can store multiple versions of an object. You can quickly search for and restore different versions or restore data in the event of accidental deletions or application faults.

By default, versioning is disabled for new OBS buckets. New objects will overwrite existing objects in case they have the same names.

Enabling Versioning

- Enabling versioning does not change the versions or contents of existing objects in the bucket. The version ID of an object is null before versioning is enabled. If a namesake object is uploaded after versioning is enabled, a version ID will be assigned to the object. For details, see **Figure 24-1**.

  Figure 24-1 Versioning (enabled vs. disabled)

- With versioning enabled, OBS automatically allocates a unique version ID to a newly uploaded object. When an object with the same name as an existing object is uploaded again, both objects are stored in OBS with the same name but different version IDs. For details, see **Figure 24-2**.

Figure 24-2 Versioning (different version IDs for namesake objects)

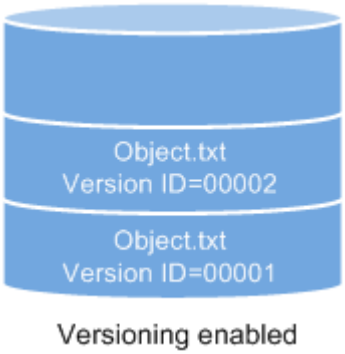
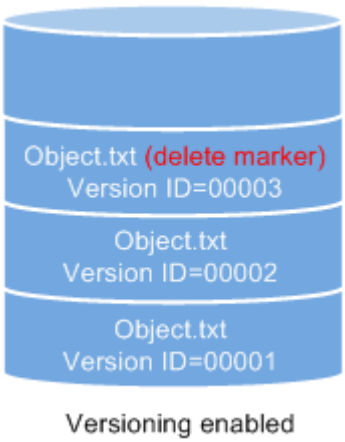


Table 24-1 Version description

Version	Description
Latest version	After versioning is enabled, each operation on an object will result in saving of the object with a new version ID. The version ID generated upon the latest operation is called the latest version.
Historical Version	After versioning is enabled, each operation on an object will result in saving of the object with a new version ID. Version IDs generated upon operations other than the latest operation are called historical versions.

- The latest objects in a bucket are returned by default after a GET Object request.
- Objects can be downloaded by version IDs. By default, the latest object is downloaded if the version ID is not specified.
- You can select an object and click **Delete** on the right to delete the object. After the object is deleted, OBS generates a **Delete Marker** with a unique version ID for the deleted object, and the deleted object is displayed in the **Deleted Objects** list. If you try to access the deleted object, a 404 error will be returned.

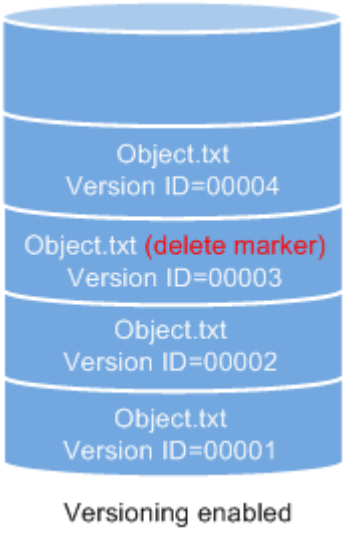
Figure 24-3 Object with a delete marker



- You can recover a deleted object by deleting the object version that has the **Delete Marker**.
- After an object is deleted, you can specify the version number in **Deleted Objects** to permanently delete the object of the specified version.
- An object is displayed either in the object list or the list of deleted objects. It will never be displayed in both the lists at the same time.

  For example, after object **A** is uploaded and deleted, it will be displayed in the **Deleted Objects** list. If you upload an object named **A** again, the object **A** will be displayed in the **Objects** list, and the previously deleted object **A** will no longer be displayed in the **Deleted Objects** list. For details, see **Figure 24-4**.

  Figure 24-4 Uploading a namesake object after the original one is deleted

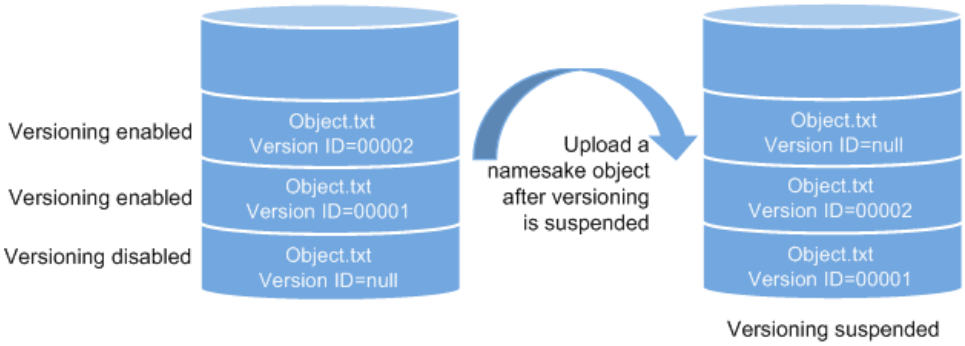
  

- All object versions except those with **Delete Marker** stored in OBS are charged.

Suspending Versioning

Once the versioning function is enabled, it can be suspended but cannot be disabled. Once versioning is suspended, version IDs will no longer be allocated to newly uploaded objects. If an object with the same name already exists and does not have a version ID, the object will be overwritten.

Figure 24-5 Object versions in the scenario when versioning is suspended



If versions of objects in a bucket do not need to be controlled, you can suspend the versioning function.

- Historical versions will be retained in OBS. If you do not need these historical versions, manually delete them.
- Objects can be downloaded by version IDs. By default, the latest object is downloaded if the version ID is not specified.
- All historical object versions except those with **Delete Marker** stored in OBS are charged.

Differences Between Scenarios When Versioning Is Suspended and Disabled

If you delete an object when versioning is suspended, a null version with the **Delete Marker** is generated regardless of whether the object has historical versions. But, if versioning is disabled, the same operation will not generate a version with the **Delete Marker**.

NOTE

After versioning is enabled, each historical version of an object is stored and occupies storage space. OBS charges storage fees for all versions. Exercise caution to avoid extra storage fees.

How to Use

You can configure versioning through the OBS Console, APIs, and SDKs.

Tool	Reference
OBS Console	**Configuring Versioning**
SDKs	OBS supports software development kits (SDKs) in multiple languages. For details, see the corresponding developer guide on the **SDK Overview** page.
API	**Configuring Versioning for a Bucket**

25 Monitoring

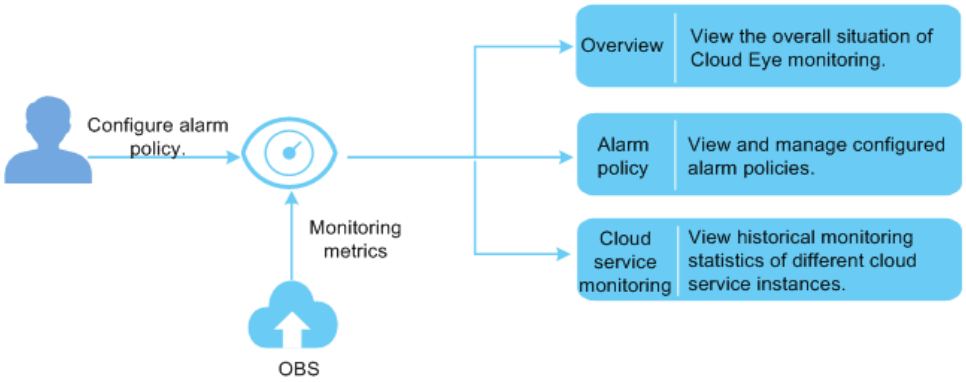
Scenarios

You may send PUT and GET requests continuously when using OBS, which generates upload and download traffic. You may also receive error responses from the server. Huawei Cloud provides Cloud Eye to help you monitor OBS and better understand your bucket statuses. Cloud Eye can perform automatic and real-time monitoring over your buckets. It triggers alarms and notifications upon user operations based on predefined policies, allowing you to keep a close eye on your bucket access requests, traffic, and error responses.

You do not need to separately subscribe to Cloud Eye. It starts automatically once you create a resource (a bucket, for example) in OBS.

For more information about Cloud Eye, see **What Is Cloud Eye?**

Figure 25-1 Cloud Eye monitoring



Setting Alarm Rules

In addition to automatic and real-time monitoring, you can configure alarm rules in Cloud Eye to have alarm notifications sent when specific situations occur.

For details about how to configure alarm rules for Cloud Eye monitoring over OBS, see **Creating an Alarm Rule**.

Viewing OBS Monitoring Metrics

Cloud Eye monitors OBS usage in real time. You can view detailed statistics of each metric on the console of Cloud Eye.

For details about how to view OBS monitoring metrics, see **Querying Metrics of a Cloud Service**.

Monitoring Metrics

For details, see **OBS Monitoring Metrics**.

26 Auditing

Scenarios

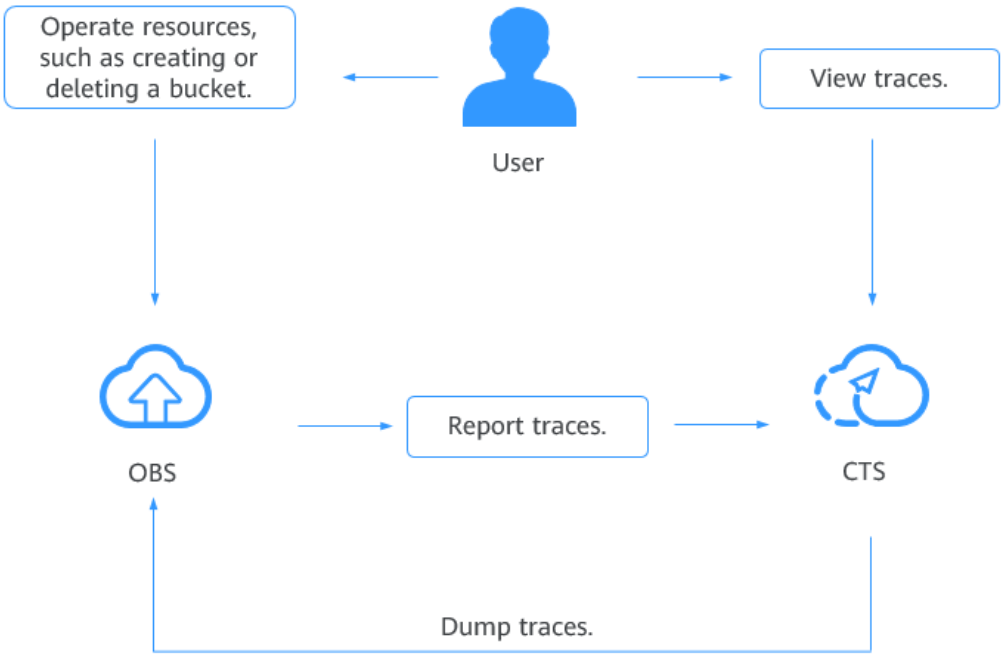
Cloud Trace Service (CTS) records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, track resource changes, audit compliance, and locate faults.

After you enable CTS and configure a tracker, CTS can record management and data traces of OBS for auditing.

For details about how to enable and configure CTS, see **Enabling CTS**.

For details about OBS management and data traces that can be traced by CTS, see **Cloud Trace Service**.

Figure 26-1 Cloud Trace Service



27 Online Decompression (OBT)

You can compress multiple files into a ZIP package and upload it to OBS.

OBS automatically decompresses ZIP packages after they are uploaded.

Scenarios

- Upload a ZIP package to an OBS bucket and then have the package automatically decompressed and stored in a specific directory.
- Upload a large number of small files at a time by using a ZIP package to save time and efforts. For batch upload constraints, see **OBS Batch Upload**.

Prerequisites

You have been assigned the **Tenant Administrator** role.

Constraints

- Online decompression is currently available only in the CN North-Beijing4, CN South-Guangzhou, and CN East-Shanghai1 regions.
- Currently, only ZIP packages are supported.
- A single ZIP package cannot exceed **1 GB** in size.
- A single ZIP package can contain a maximum of 65,536 files.
- A single decompressed file cannot exceed 40 GB in size.
- Decompressing a ZIP package takes 10 minutes at most.
- ZIP package names cannot contain Chinese punctuation marks, special characters, or special codes.
- To decompress the ZIP package that contains other ZIP packages, the event type of the online decompression policy must be set to **ObjectCreated:*** or **ObjectCreated:CompleteMultipartUpload**.
- Currently, no notification will be sent to users after decompression tasks are complete.
- Currently, only deflate-compressed ZIP packages can be decompressed. Deflate is different from the compression algorithms (such as Store and Normal) provided by WinRAR.

- The total length of the decompression path plus the name of any decompressed file cannot exceed 1024 characters.
- Currently, encrypted ZIP packages cannot be decompressed.

Precautions

- You are advised to set a precise prefix for a decompression policy. In the same bucket, trigger conditions (including events, prefixes, or suffixes) of different decompression policies cannot overlap with each other.

  For example, there are two decompression policies **event-0001** and **event-0002** in a bucket. If the prefix of **event-0001** is **aa**, the prefix of **event-0002** cannot be **aaaa**, because **aa** is contained in **aaaa**.
- If the prefix is left blank, the decompression policy applies to all the ZIP packages in the bucket by default. This may trigger cyclic decompression if a package contains other packages.

  For example, package **AA.zip** contains another package **BB.zip**. If the prefix is left blank, the system continues to decompress **BB.zip** after decompressing **AA.zip**. This issue will not happen if a prefix is set in the policy.
- You must set a directory for storing the decompressed files. If the directory is not set, decompressed files will be stored in the home directory of the current bucket.
- You are advised to encode file or folder names using UTF-8. Otherwise, names of decompressed files or folders may contain garbled characters, or the decompression may be interrupted.
- A ZIP package decompression will fail if it takes more than 10 minutes.

How to Use

Configure the policies for decompressing ZIP packages online through OBS Console or APIs.

Tool	Reference
OBS Console	**Creating an Online Decompression Policy**
API	**Configuring an Online Decompression Policy**

28 Change History

Date	What's New
2023-05-05	This is the sixth official release. • Updated the content related to folder sharing by URL.
2023-04-23	This is the fifth official release. • Added **WORM**.
2023-01-19	This is the fourth official release. • Removed the content related to event notifications.
2022-08-08	This is the third official release. • Added **Back to Source**.
2022-07-30	This is the second official release. • Added **Online Decompression (OBT)**.
2020-08-31	This is the first official release.

