Object Storage Service

Getting Started

Issue 01

Date 2024-09-29

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 Using OBS Console

OBS Console is a web-based GUI where you can manage your OBS resources with ease.

The following describes how to use basic functions on OBS Console, including creating a bucket (**test-example-bucket** as an example), uploading an object, as well as downloading and sharing the object.

Preparations

Before using OBS, you must create a HUAWEI ID, complete real-name authentication, and top up your account.

Step 1 Sign up for a HUAWEI ID and complete real-name authentication.

If you already have a HUAWEI ID, skip this step. Otherwise, do as follows:

1. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
2. Complete real-name authentication by referring to **Individual Real-Name Authentication** or .

Step 2 **Top up your account**.

Make sure your account balance is sufficient, so that you can properly use OBS and other related resources.

----End

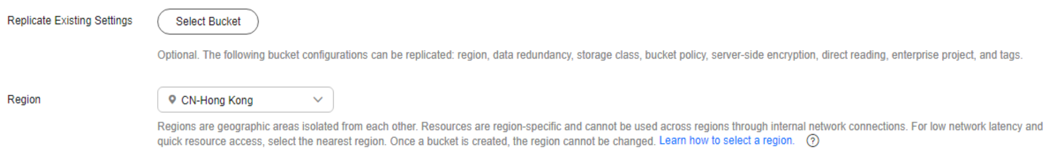
Creating a Bucket

Buckets are containers that store objects in OBS. Before you can upload objects, you must create a bucket.

This example only covers some key settings for creating a bucket. Retain the default values for other parameters. For more information, see **Creating a Bucket**.

Step 1 Visit the **bucket creation** page on OBS Console.

Step 2 Specify **General Configuration**.


Replicate Existing Settings: Select Bucket

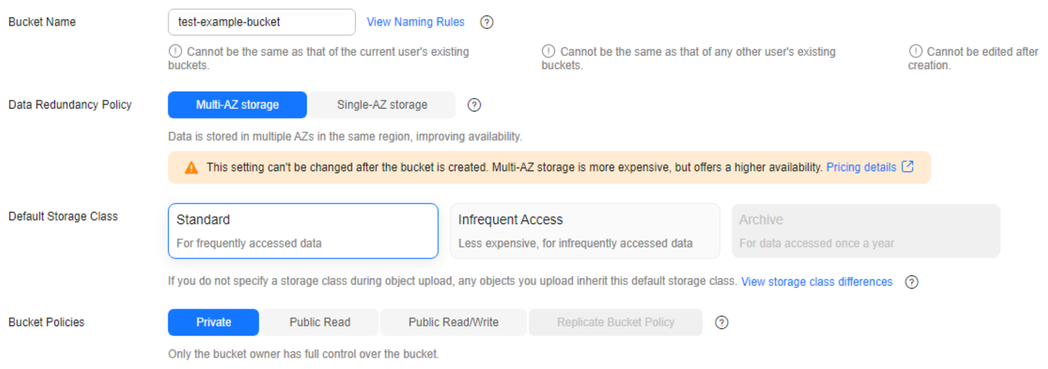
Optional. The following bucket configurations can be replicated: region, data redundancy, storage class, bucket policy, server-side encryption, direct reading, enterprise project, and tags.

Region: CN-Hong Kong

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region. Once a bucket is created, the region cannot be changed. Learn how to select a region.

Parameter	Value Example	Description
Region	CN-Hong Kong	Geographic area where a bucket is located. For lower latency and faster access, select the region nearest where your bucket will be accessed. Once a bucket is created, its region cannot be changed.

Step 3 Specify **Bucket Settings**. Retain the default settings for the other parameters. You can also modify them after the bucket is created.


Bucket Name: test-example-bucket View Naming Rules

Cannot be the same as that of the current user's existing buckets. Cannot be the same as that of any other user's existing buckets. Cannot be edited after creation.

Data Redundancy Policy: Multi-AZ storage (selected) | Single-AZ storage

Data is stored in multiple AZs in the same region, improving availability.

This setting can't be changed after the bucket is created. Multi-AZ storage is more expensive, but offers a higher availability. Pricing details

Default Storage Class: Standard (selected) — For frequently accessed data | Infrequent Access — Less expensive, for infrequently accessed data | Archive — For data accessed once a year

If you do not specify a storage class during object upload, any objects you upload inherit this default storage class. View storage class differences

Bucket Policies: Private (selected) | Public Read | Public Read/Write | Replicate Bucket Policy

Only the bucket owner has full control over the bucket.

Parameter	Value Example	Description
Bucket Name	test-example-bucket	Name of a bucket. After a bucket is created, its name cannot be changed. A bucket name: ● Must be globally unique, which means a bucket must have a different name from any existing bucket (including those created by other accounts). The name of a deleted bucket can be reused for another bucket or parallel file system at least 30 minutes 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.
Data Redundancy Policy	Multi-AZ storage	● **Multi-AZ storage**: Data is stored in multiple availability zones (AZs) within a region. This storage policy costs more, but has a higher reliability. ● **Single-AZ storage**: Data is stored in a single AZ, with lower costs. After a bucket is created, its data redundancy policy cannot be changed. Buckets with multi-AZ storage do not support the Archive storage class.
Storage Class	Standard	Storage classes of a bucket. You can choose a storage class that meets your needs for storage performance and costs. ● **Standard**: It is for storing a large number of hot files or small files that are frequently accessed (multiple times per month on average) and require fast access. ● **Infrequent Access**: It is for storing data that is less frequently accessed (less than 12 times per year on average), but when needed, the access has to be fast. ● **Archive**: It is for archiving data that is rarely accessed (once a year on average) and does not require fast access. For more information, see **Storage Classes**.

Parameter	Value Example	Description
Bucket Policies	Private	Controls read and write permissions for a bucket. ● **Private**: Only users granted permissions by the bucket ACL can access the bucket. ● **Public Read**: Anyone can read objects in the bucket. ● **Public Read/Write**: Anyone can read, write, or delete objects in the bucket.
Enterprise Project	default	Project where you can add your bucket for unified management. If you do not have any specific needs for enterprise project division and management, choose the **default** enterprise project. To learn more about how to manage OBS buckets using enterprise projects, see the **Enterprise Project** parameter in **Creating a Bucket**.

Step 4 In the lower right corner of the page, click **Create Now**. Then, confirm the information displayed.

If the bucket is successfully created, a message will be displayed and the bucket will appear in the bucket list.

----End

Uploading an Object

On OBS Console, you can upload a single file up to 5 GB in size, or you can upload up to 100 files (with a total size of no more than 5 GB) in a batch.

This example only covers some key settings for uploading an object. For more information, see **Uploading an Object**.

Step 1 In the bucket list, click the created bucket.

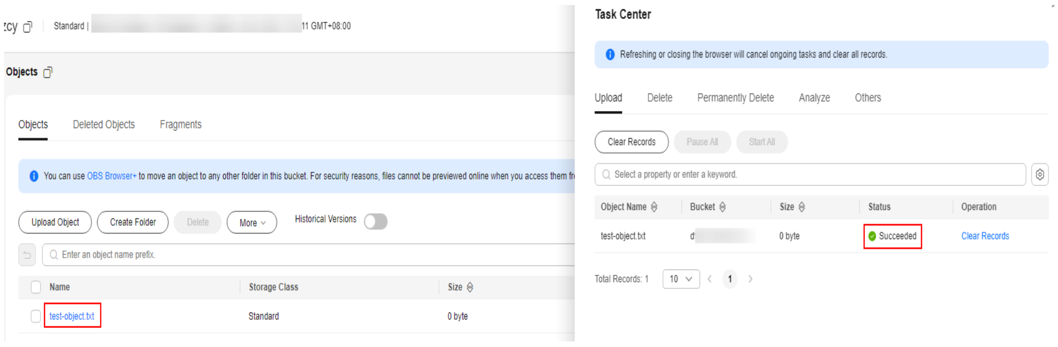
Step 2 On the **Objects** page, click **Upload Object** above the search box.

Step 3 In the **Upload Object** area, drag and drop the local file to upload.

Alternatively, click **add files** and choose the local file to upload.

Step 4 Retain the default settings for other parameters and click **Upload**.

The **Task Center** page is displayed on the right, where you can view the upload status. After the object is uploaded, it will appear in the object list.



You can use OBS Browser+ to move an object to any other folder in this bucket. For security reasons, files cannot be previewed online when you access them fr

Name	Storage Class	Size
test-object.txt	Standard	0 byte

Refreshing or closing the browser will cancel ongoing tasks and clear all records.

Object Name	Bucket	Size	Status	Operation
test-object.txt	d[redacted]	0 byte	Succeeded	Clear Records

----End

Downloading Objects

You can download objects from an OBS bucket to the default path or a specified local path.

Step 1 In the object list, find the uploaded object.

Step 2 Download the object.

- **To download a single object, do as follows**:

  Click **Download** in the **Operation** column of the object. The object is downloaded to the default path.

- **To download multiple objects in a batch, do as follows**:

  Select multiple objects and choose **More** > **Download** above the search box. The objects are downloaded to the default path.

----End

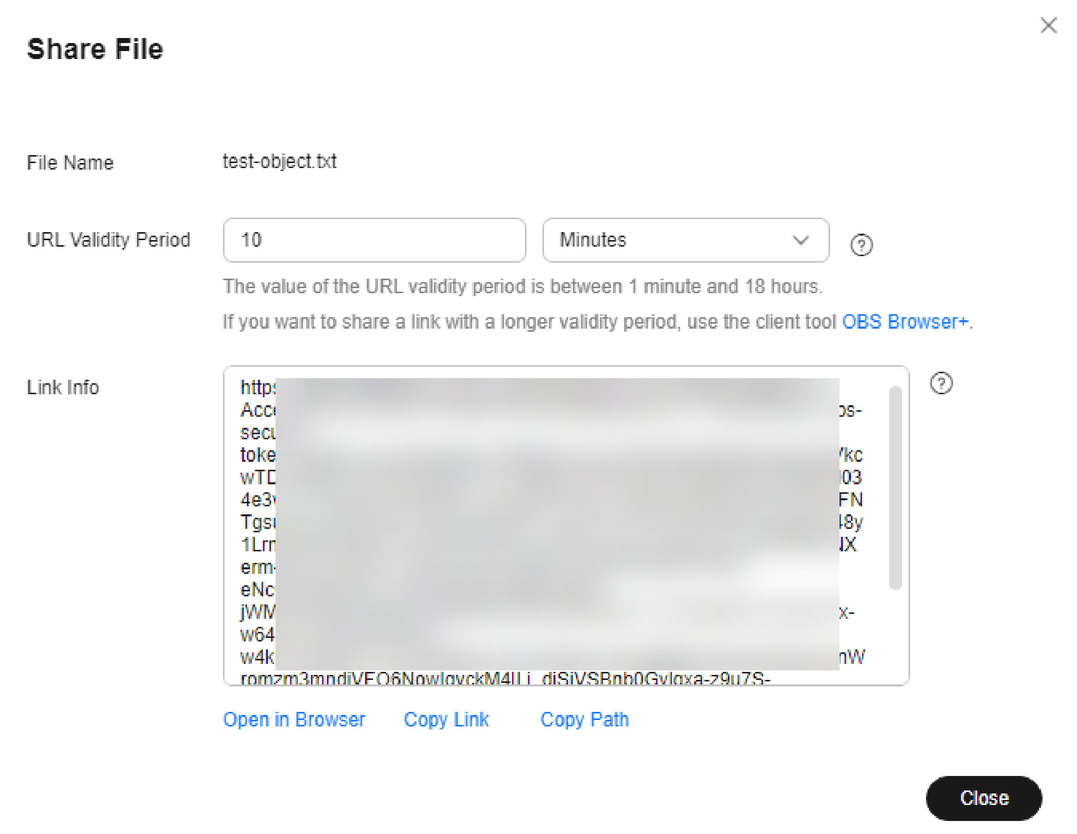
Sharing an Object

You can share the URL of an object with others, so that they can download or view the object.

Step 1 On the object list page, locate the object you want to share and click **Share** in the **Operation** column.

Step 2 In the **Share File** dialog box, specify a URL validity period (**10** minutes as an example).

- A validity period ranges from one minute to 18 hours. Within the validity period, anyone can use the URL to access the object.
- The moment the **Share File** dialog box is opened, the link information is in effect and valid for five minutes by default. Each time you change the URL validity period, the URL takes effect again and changes accordingly. You need to re-share the URL.


Share File

File Name	test-object.txt
URL Validity Period	10 Minutes The value of the URL validity period is between 1 minute and 18 hours. If you want to share a link with a longer validity period, use the client tool OBS Browser+.
Link Info	https [redacted] romzm3mndiVEO6NowlqvckM4ILi_diSiVSBnb0Gvlqxa-z9u7S-

Step 3 Share the object.

- Click **Copy Link** and share the link with others. Then, they can use this link to access the object in a browser.

  NOTE

  Accessing an object in a browser will make the object forcibly downloaded. If you want people to preview the object you shared, see **How Do I Preview OBS Objects in My Web Browser?**

- Click **Copy Path** and share the path with users who have access to the bucket that stores the shared object. Then, they can go to the object list of the bucket and paste the shared path in the object search box and press **Enter** to search for or access the object.

----End

Related Information

In addition to the operations mentioned above, the following advanced OBS functions are available for you to use.

- **Lifecycle management**: You can configure lifecycle rules to periodically transition objects between storage classes or delete objects.
- **Access control**: You can use IAM permissions, bucket policies, bucket ACLs, and object ACLs to implement refined access control over buckets and objects.
- **Domain name management**: You can add a user-defined domain name to a bucket and then use this domain name to access the bucket and objects in it..
- **Static website hosting**: You can upload a static website file to an OBS bucket, grant anonymous users the read permission for this file, and configure

static website hosting for the bucket, so that you can use the bucket domain name to access the hosted static website file.

- **Versioning**: You can enable versioning for a bucket to keep multiple versions of an object in the same bucket. This way, you can easily retrieve and restore each object version and recover data from unintended actions or application failures.
- **Cross-region replication**: You can configure a cross-region replication rule to replicate data from a source bucket to a destination bucket in a different region.
- **Server-side encryption**: You can use server-side encryption to encrypt data uploaded to OBS buckets.
- **WORM**: You can configure WORM for your bucket to prevent objects in the bucket from being deleted or tampered with within a specified period.
- **Cross-origin resource sharing (CORS)**: You can configure a CORS rule to enable cross-domain quests for OBS.

2 Using OBS Browser+

OBS Browser+ is a GUI client for easily managing data stored inOBS.

The following describes how to use basic functions on OBS Browser+, including creating a bucket (**test-example-bucket** as an example), uploading an object, as well as downloading and sharing the object.

Preparations

Before using OBS Browser+, you must create a HUAWEI ID, complete real-name authentication, and top up your account. Then, you need to install OBS Browser+ and obtain the access keys.

Step 1 Sign up for a HUAWEI ID and complete real-name authentication.

If you already have a HUAWEI ID, skip this step. Otherwise, do as follows:

1. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
2. Complete real-name authentication by referring to **Individual Real-Name Authentication** or .

Step 2 Top up your account by referring to **Topping Up an Account**.

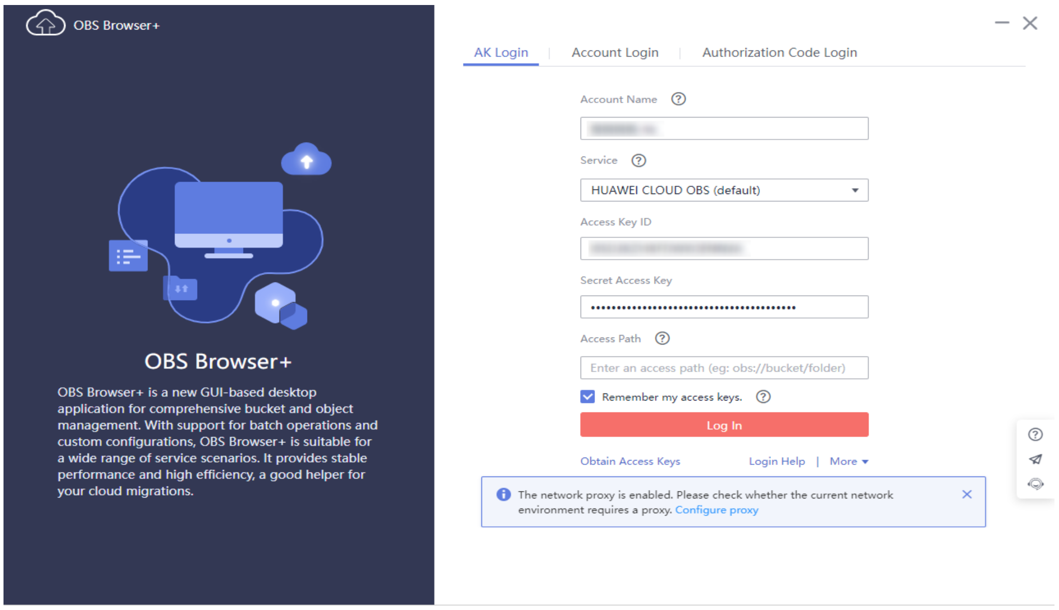
Make sure your account balance is sufficient, so that you can properly use OBS and other related resources.

Step 3 **Download** and install OBS Browser+.

Step 4 Obtain the access keys by referring to **Access Keys**.

----End

Using an AK/SK Pair to Log In to OBS Browser+


OBS Browser+

AK Login | Account Login | Authorization Code Login

Account Name

Service
HUAWEI CLOUD OBS (default)

Access Key ID

Secret Access Key
••

Access Path
Enter an access path (eg: obs://bucket/folder)

☑ Remember my access keys.

Log In

Obtain Access Keys   Login Help | More

The network proxy is enabled. Please check whether the current network environment requires a proxy. Configure proxy

OBS Browser+

OBS Browser+ is a new GUI-based desktop application for comprehensive bucket and object management. With support for batch operations and custom configurations, OBS Browser+ is suitable for a wide range of service scenarios. It provides stable performance and high efficiency, a good helper for your cloud migrations.

Parameter	Value Example	Description
Account Name	**exampleAccount**	It is user-defined and is a unique identifier that is different from the cloud service accounts you use to log in to OBS Browser+. An account name contains 3 to 63 characters, and cannot contain the following special characters: \ : * ? ' < > | ! @ # $ % ^ ~
Service	HUAWEI CLOUD OBS (default)	**HUAWEI CLOUD OBS (default)** is selected. Selecting this option allows access to buckets in all regions where Huawei Cloud is available.
Access Key ID & Secret Access Key	-	Enter the AK and SK. You can create a permanent AK/SK pair on the **My Credentials** page by referring to **Creating an Access Key**.

NOTE

For more login information, see **Logging In to OBS Browser+**.

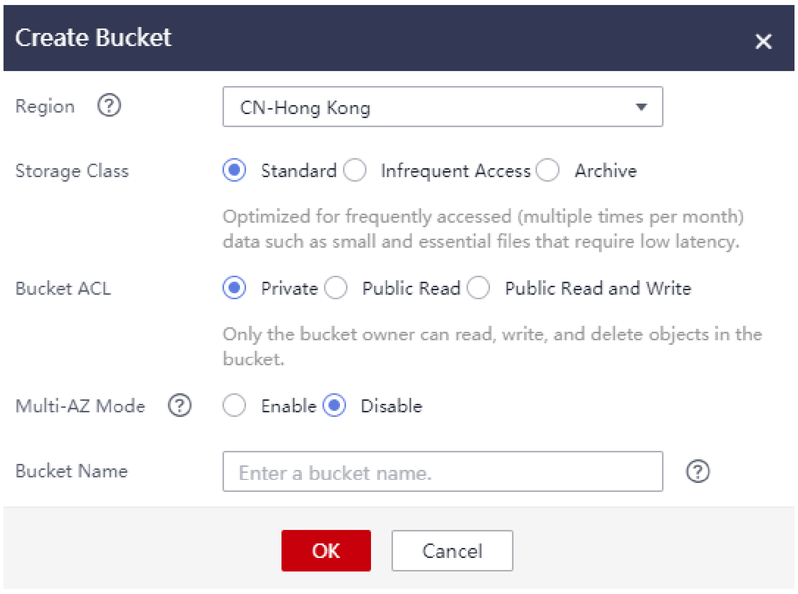
Creating a Bucket

Buckets are containers that store objects in OBS. Before you can store data, you must create a bucket.

This example only covers some key settings for creating a bucket. Retain the default values for other parameters. For more information, see **Creating a Bucket**.

Step 1 Log in to OBS Browser+. On the displayed bucket list page, click **Create Bucket** in the upper left corner.

Step 2 In the displayed dialog box, specify bucket information and click **OK**.



Region: CN-Hong Kong

Storage Class: [✓ Standard] Infrequent Access Archive

Optimized for frequently accessed (multiple times per month) data such as small and essential files that require low latency.

Bucket ACL: [✓ Private] Public Read Public Read and Write

Only the bucket owner can read, write, and delete objects in the bucket.

Multi-AZ Mode: Enable [✓ Disable]

Bucket Name: Enter a bucket name.

Parameter	Value Example	Description
Region	CN-Hong Kong	Geographic area where a bucket is located. For lower latency and faster access, select the region nearest where your bucket will be accessed. Once a bucket is created, its region cannot be changed.

Parameter	Value Example	Description
Storage Class	Standard	Storage classes of a bucket. You can choose a storage class that meets your needs for storage performance and costs. ● **Standard**: It is for storing a large number of hot files or small files that are frequently accessed (multiple times per month on average) and require fast access. ● **Infrequent Access**: It is for storing data that is less frequently accessed (less than 12 times per year on average), but when needed, the access has to be fast. ● **Archive**: It is for archiving data that is rarely accessed (once a year on average) and does not require fast access. For more information, see **Storage Classes**.
Bucket ACL	Private	Controls read and write permissions for a bucket. ● **Private**: Only users granted permissions by the bucket ACL can access the bucket. ● **Public Read**: Anyone can read objects in the bucket. ● **Public Read/Write**: Anyone can read, write, or delete objects in the bucket.
Bucket Name	test-example-bucket	Name of a bucket. After a bucket is created, its name cannot be changed. Based on the globally applied DNS naming conventions, an OBS bucket name: ● Must be globally unique, which means a bucket must have a different name from any existing bucket (including those created by other users). The name of a deleted bucket can be reused for another bucket or parallel file system at least 30 minutes 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.

Step 3 Check the bucket you just created in the bucket list.

----End

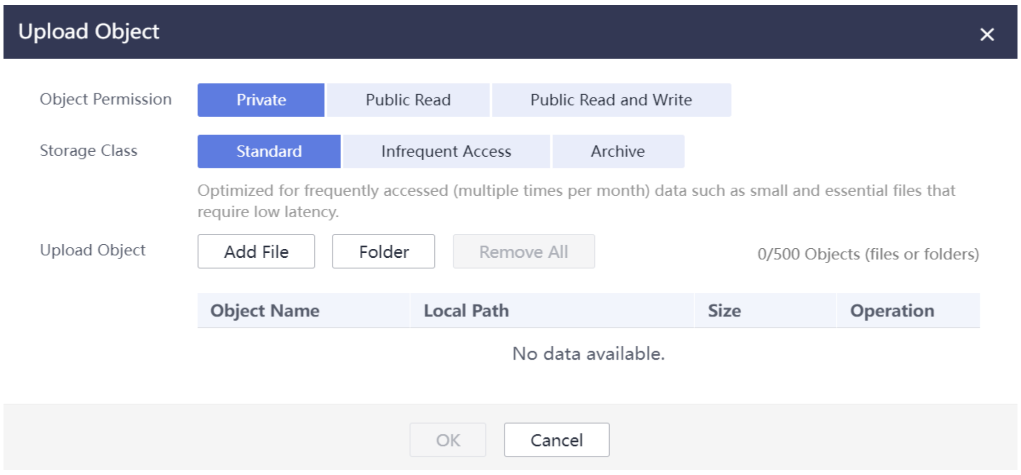
Uploading an Object

On OBS Browser+, you can upload a single file up to 48.8 TB in size, or you can upload up to 500 files (with a total size of no more than 48.8 TB) in a batch.

Step 1 In the bucket list, click the created bucket.

Step 2 On the object list page, click **Upload** in the upper left corner.

Step 3 Then, click **Add File** to choose a local file to upload.



Upload Object

Object Permission: Private | Public Read | Public Read and Write

Storage Class: Standard | Infrequent Access | Archive

Optimized for frequently accessed (multiple times per month) data such as small and essential files that require low latency.

Upload Object: Add File | Folder | Remove All — 0/500 Objects (files or folders)

Object Name	Local Path	Size	Operation

No data available.

OK | Cancel

Step 4 Retain the default settings for other parameters and click **OK**.

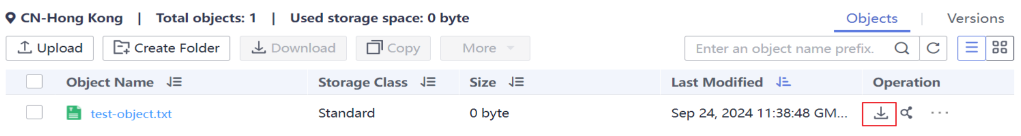
Then, you can check the object in the object list.

----End

Downloading an Object

You can download an object from the bucket to your local path.

Step 1 In the object list, locate the uploaded object and click ⤓ in the **Operation** column.



CN-Hong Kong | Total objects: 1 | Used storage space: 0 byte

Object Name	Storage Class	Size	Last Modified	Operation
test-object.txt	Standard	0 byte	Sep 24, 2024 11:38:48 GM...	

Step 2 In the displayed dialog box, select a local path and click **OK**.

Then, you can find the downloaded object in the local path you specified.

----End

Sharing an Object

You can share the URL of an object with others, so that they can download or view the object.

Step 1 Click the icon in the **Operation** column of the uploaded object.

Step 2 In the **Share File** dialog box, specify a URL validity period (1 day as an example).

- Within the validity period, anyone can use the URL to access the object.
- The moment the **Share File** dialog box is opened, the URL is in effect and valid for 1 day by default. Each time you change the URL validity period, the URL takes effect again and changes accordingly. You need to re-share the URL.

Step 3 Share the object.

- Click **Copy Link** and share the link with others. Then, they can use this link to access the object in a browser.

  NOTE

  Accessing an object in a browser will make the object forcibly downloaded. If you want people to preview the object you shared, see **How Do I Preview OBS Objects in My Web Browser?**
- Click **Copy Path** and share the path with users who have access to the bucket that stores the shared object. Then, they can go to the object list of the bucket and paste the shared path in the object search box and press **Enter** to search for or access the object.

----End

Related Information

In addition to the operations mentioned above, the following advanced OBS Browser+ functions are available for you to use.

- **Lifecycle management**: You can configure lifecycle rules to periodically transition objects between storage classes or delete objects.
- **Bucket ACL**: You (the bucket owner) can modify the bucket ACL on OBS Browser+ to implement fine-grained access control.
- **Bucket policies**: You (the bucket owner) can write bucket policies on OBS Browser+ to implement fine-grained access control.

3 Using obsutil

obsutil is a command line tool. If you are familiar with the command line interface (CLI), obsutil is a good choice in batch processing and automatic tasks.

The following describes how to use basic OBS functions with obsutil in Linux, including creating a bucket (**bucket-test** as an example), uploading an object, as well as downloading and sharing the object.

Preparations

Before using obsutil, you must create a HUAWEI ID, complete real-name authentication, and top up your account. Then, you need to install obsutil and obtain the access keys.

Step 1 Sign up for a HUAWEI ID and complete real-name authentication.

If you already have a HUAWEI ID, skip this step. Otherwise, do as follows:

1. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
2. Complete real-name authentication by referring to **Individual Real-Name Authentication** or .

Step 2 Top up your account by referring to **Topping Up an Account**.

Make sure your account balance is sufficient, so that you can properly use OBS and other related resources.

Step 3 Download and install obsutil by referring to **Downloading and Installing obsutil**.

Step 4 Obtain the access keys by referring to **Access Keys**.

----End

Creating a Bucket

Create a bucket (**bucket-test** as an example, which is user-defined and must be unique) in the CN-Hong Kong region:

./obsutil mb obs://bucket-test -location=ap-southeast-1

If the command output is as follows, the bucket is created.

```
Create bucket [bucket-test] successfully, request id [00000190CA0531DE4015FE3458C7B4C0]
```

Uploading an Object

Upload the local **test.txt** file to bucket **bucket-test**:

./obsutil cp /temp/test.txt obs://bucket-test/test.txt

If the command output is as follows, the file is uploaded.

```
Parallel:      5                   Jobs:          5
Threshold:     52428800            PartSize:      5242880
VerifyLength:  false               VerifyMd5:     false
CheckpointDir: /root/.obsutil_checkpoint

test.txt:[j1] [=============================================] 100.00% 48.47 KB/s 0s
Upload successfully, 4.44KB, /temp/test.txt --> obs://bucket-test1/test.txt
```

Downloading an Object

Download the **test.txt** object from bucket **bucket-test** to your local PC:

./obsutil cp obs://bucket-test/test.txt /temp/test1.txt

If the command output is as follows, the object is downloaded.

```
Parallel:      5                   Jobs:          5
Threshold:     52428800            PartSize:      5242880
VerifyLength:  false               VerifyMd5:     false
CheckpointDir: /root/.obsutil_checkpoint

test.txt:[============================================] 100.00% 775.52 KB/s 0s
Download successfully, 4.44KB, obs://bucket-test1/test.txt --> /temp/test1.txt
```

Sharing an Object

Step 1 Create a download URL for the object:

obsutil sign obs://bucket-test/test.txt

If the command output is as follows, the download URL is created.

```
Download url of [obs://bucket-test/test.txt] is:
  http://bucket-test.obs.ap-southeast-1.myhuaweicloud.com/bucket-test/test.txt?
AccessKeyId=xxxx&Expires=1552548758&Signature=xxxx
```

Step 2 Copy and share the URL with others. Then, they can use this URL to access the object in a browser.

NOTE

Accessing an object in a browser will make the object forcibly downloaded. If you want people to preview the object you shared, see **How Do I Preview OBS Objects in My Web Browser?**

----End

Related Information

In addition to the operations mentioned above, the following advanced obsutil functions are available for you to use.

- **Bucket policies**: obsutil provides bucket policies to enable fine-grained access control over buckets and objects.

4 Using OBS SDKs

OBS software development kits (SDKs) encapsulate the REST API provided by OBS to simplify development. You can call API functions provided by OBS SDKs to use OBS.

The following describes how to use basic OBS functions with SDK for Java, SDK for Python, and SDK for Go, including creating a bucket, as well as uploading, downloading, and listing objects.

Preparations

Before using the SDK for Java, you must create a HUAWEI ID, complete real-name authentication, and top up your account. Then, you need to obtain the access keys, set up a development environment, and install the SDK for Java.

Step 1 Sign up for a HUAWEI ID and complete real-name authentication.

If you already have a HUAWEI ID, skip this step. Otherwise, do as follows:

1. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
2. Complete real-name authentication by referring to **Individual Real-Name Authentication** or .

Step 2 Top up your account by referring to **Topping Up an Account**.

Make sure your account balance is sufficient, so that you can properly use OBS and other related resources.

Step 3 Obtain the access keys by referring to **Access Keys**.

Step 4 Set up a development environment by referring to **Setting Up a Development Environment**.

Step 5 Download and install the SDK for Java by referring to **SDK Download and Installation (SDK for Java)**.

----End

Creating a Bucket

The following example shows how to create a private bucket (**examplebucket** as an example) in the CN-Hong Kong (ap-southeast-1) region. The storage class is set to Standard. The redundancy is set to multi-AZ storage.

Java

```
import com.obs.services.ObsClient;
import com.obs.services.exception.ObsException;
import com.obs.services.model.AccessControlList;
import com.obs.services.model.AvailableZoneEnum;
import com.obs.services.model.CreateBucketRequest;
import com.obs.services.model.ObsBucket;
import com.obs.services.model.StorageClassEnum;

public class CreateBucket001 {
    public static void main(String[] args) {
        // Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        // Obtain an AK/SK pair on the management console.
        String ak = System.getenv("ACCESS_KEY_ID");
        String sk = System.getenv("SECRET_ACCESS_KEY_ID");

        // Enter the endpoint for the region where you want to create the bucket. The CN-Hong Kong (ap-
southeast-1) region is used here as an example. Replace it with the one in use.
        String endPoint = "https://obs.ap-southeast-1.myhuaweicloud.com";
        // Enter the endpoint for the region where you want to create the bucket.
        String endPoint = "https://your-endpoint";

        // Create an ObsClient instance.
        // Use the permanent AK/SK pair to initialize the ObsClient.
        ObsClient obsClient = new ObsClient(ak, sk,endPoint);

        try {
            CreateBucketRequest request = new CreateBucketRequest();
            //Specify a bucket name.
            String exampleBucket = "examplebucket";
            //Specify a bucket location.
            String exampleLocation = "ap-southeast-1";
            request.setBucketName(exampleBucket);
            // Set the bucket ACL to private (the default value).
            request.setAcl(AccessControlList.REST_CANNED_PRIVATE);
            // Set the bucket storage class to Standard.
            request.setBucketStorageClass(StorageClassEnum.STANDARD);
            // Specify the bucket location (CN-Hong Kong as an example). The location must match the
endpoint.
            request.setLocation(exampleLocation);
            // Specify the multi-AZ redundancy for the bucket. If this field is not configured, a single-AZ bucket
will be created.
            request.setAvailableZone(AvailableZoneEnum.MULTI_AZ);
            // Create the bucket.
            ObsBucket bucket = obsClient.createBucket(request);
            // The bucket is created.
            System.out.println("CreateBucket successfully");
            System.out.println("RequestId:"+bucket.getRequestId());


        } catch (ObsException e) {
            System.out.println("CreateBucket failed");
            // Request failed. Print the HTTP status code.
            System.out.println("HTTP Code: " + e.getResponseCode());
            // Request failed. Print the server-side error code.
            System.out.println("Error Code:" + e.getErrorCode());
            // Request failed. Print the error message.
            System.out.println("Error Message: " + e.getErrorMessage());
            // Request failed. Print the request ID.
            System.out.println("Request ID:" + e.getErrorRequestId());
            System.out.println("Host ID:" + e.getErrorHostId());
        } catch (Exception e) {
            System.out.println("CreateBucket failed");
            // Print other error information.
            e.printStackTrace();

        }
    }
}
```

Python

```
from obs import CreateBucketHeader, HeadPermission
from obs import ObsClient
import os
import traceback

# Obtain an AK and SK pair using environment variables or import the AK and SK pair in other ways. Using hard coding may result in leakage.
# Obtain an AK/SK pair on the management console.
ak = os.getenv("AccessKeyID")
sk = os.getenv("SecretAccessKey")
# Set server to the endpoint for the region where you want to create the bucket. The CN-Hong Kong (ap-southeast-1) region is used here as an example. Replace it with the one in use.
server = "https://obs.ap-southeast-1.myhuaweicloud.com"
# Set server to the endpoint for the region where you want to create the bucket.
server = "https://your-endpoint"

# Create an ObsClient instance.
obsClient = ObsClient(access_key_id=ak, secret_access_key=sk, server=server)
try:
    # Specify the additional headers to create a private bucket that is in the Standard storage class and supports multi-AZ storage.
    header = CreateBucketHeader(aclControl=HeadPermission.PRIVATE, storageClass="STANDARD", availableZone="3az")
    # Specify the region where you want to create the bucket. ap-southeast-1 is used here as an example. The region must be the one you specified in the endpoint.
    location = "ap-southeast-1"
    # Specify the region where you want to create the bucket. The region must be the one you specified in the endpoint.
    location = "region"
    bucketName = "examplebucket"
    # Create the bucket.
    resp = obsClient.createBucket(bucketName, header, location)
    # If status code 2xx is returned, the API call succeeds. Otherwise, the API call fails.
    if resp.status < 300:
        print('Create Bucket Succeeded')
        print('requestId:', resp.requestId)
    else:
        print('Create Bucket Failed')
        print('requestId:', resp.requestId)
        print('errorCode:', resp.errorCode)
        print('errorMessage:', resp.errorMessage)
except:
    print('Create Bucket Failed')
    print(traceback.format_exc())
```

Go

```
package main

import (
    "fmt"
    "os"
    "obs-sdk-go/obs"
    obs "github.com/huaweicloud/huaweicloud-sdk-go-obs/obs"
)

func main() {
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console.
    ak := os.Getenv("AccessKeyID")
    sk := os.Getenv("SecretAccessKey")
    // Enter the endpoint for the region where you want to create the bucket. The CN-Hong Kong (ap-
southeast-1) region is used here as an example. Replace it with the one in use.
    endPoint := "https://obs.ap-southeast-1.myhuaweicloud.com"
    // Enter the endpoint for the region where you want to create the bucket.
    endPoint := "https://your-endpoint"
    // Create an ObsClient instance.
    obsClient, err := obs.New(ak, sk, endPoint)
    if err != nil {
        fmt.Printf("Create obsClient error, errMsg: %s", err.Error())
    }
    input := &obs.CreateBucketInput{}
    // Specify a bucket name.
    input.Bucket = "examplebucket"
    // Specify the region where you want to create the bucket. ap-southeast-1 is used here as an example.
The region must be the one you specified in the endpoint.
    input.Location = "ap-southeast-1"
    // Specify the region where you want to create the bucket. The region must be the one you specified in
the endpoint.
    input.Location = "region"
    // Set the bucket ACL to obs.AclPrivate (an example).
    input.ACL = obs.AclPrivate
    // Specify the storage class of the bucket. obs.StorageClassWarm is used here as an example. If this
field is not specified, a Standard bucket will be created.
    input.StorageClass = obs.StorageClassWarm
    // Specify the AZ redundancy type. 3-AZ redundancy is used here as an example. If this field is not
specified or the region does not support multi-AZ storage, single-AZ redundancy will be applied.
    input.AvailableZone = "3az"
    // Create the bucket.
    output, err := obsClient.CreateBucket(input)
    if err == nil {
        fmt.Printf("Create bucket:%s successful!\n", input.Bucket)
        fmt.Printf("RequestId:%s\n", output.RequestId)
        return
    }
    fmt.Printf("Create bucket:%s fail!\n", input.Bucket)
    if obsError, ok := err.(obs.ObsError); ok {
        fmt.Println("An ObsError was found, which means your request sent to OBS was rejected with an error
response.")
        fmt.Println(obsError.Error())
    } else {
        fmt.Println("An Exception was found, which means the client encountered an internal problem when
attempting to communicate with OBS, for example, the client was unable to access the network.")
        fmt.Println(err)
    }
}
```

Uploading an Object

The following example shows how to upload the **localfile** file to bucket **examplebucket** and set the object name to **objectname**.

Java

```
import com.obs.services.ObsClient;
import com.obs.services.exception.ObsException;
import com.obs.services.model.PutObjectRequest;
import java.io.File;
public class PutObject004 {
    public static void main(String[] args) {
        // Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        // Obtain an AK/SK pair on the management console.
        String ak = System.getenv("ACCESS_KEY_ID");
        String sk = System.getenv("SECRET_ACCESS_KEY_ID");
        // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1) 
region is used here as an example. Replace it with the one in use.
        String endPoint = "https://obs.ap-southeast-1.myhuaweicloud.com";
        // Enter the endpoint for the region where the bucket is created.
        String endPoint = "https://your-endpoint";

        // Create an ObsClient instance.
        // Use the permanent AK/SK pair to initialize the ObsClient.
        ObsClient obsClient = new ObsClient(ak, sk,endPoint);

        try {
            // Upload a file.
            PutObjectRequest request = new PutObjectRequest();
            // Set the bucket name to examplebucket.
            request.setBucketName("examplebucket");
            // Specify the name of the file to be uploaded to the examplebucket bucket.
            request.setObjectKey("objectname");
            // Specify the path of the local file to be uploaded.
            request.setFile(new File("localfile"));
            obsClient.putObject(request);
            System.out.println("putObject successfully");
        } catch (ObsException e) {
            System.out.println("putObject failed");
            // Request failed. Print the HTTP status code.
            System.out.println("HTTP Code:" + e.getResponseCode());
            // Request failed. Print the server-side error code.
            System.out.println("Error Code:" + e.getErrorCode());
            // Request failed. Print the error message.
            System.out.println("Error Message:" + e.getErrorMessage());
            // Request failed. Print the request ID.
            System.out.println("Request ID:" + e.getErrorRequestId());
            System.out.println("Host ID:" + e.getErrorHostId());
            e.printStackTrace();
        } catch (Exception e) {
            System.out.println("putObject failed");
            // Print other error information.
            e.printStackTrace();
        }
    }
}
```

Python

```
from obs import ObsClient
from obs import PutObjectHeader
import os
import traceback

# Obtain an AK and SK pair using environment variables or import the AK and SK pair in other ways. Using 
hard coding may result in leakage.
# Obtain an AK/SK pair on the management console.
# Before running this sample code, ensure that the environment variables AccessKeyID and 
SecretAccessKey have been configured.
ak = os.getenv("AccessKeyID")
sk = os.getenv("SecretAccessKey")
# Set server to the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-
southeast-1) region is used here as an example. Replace it with the one in use.
server = "https://obs.ap-southeast-1.myhuaweicloud.com"
# Set server to the endpoint for the region where the bucket is created.
server = "https://your-endpoint"

# Create an ObsClient instance.
obsClient = ObsClient(access_key_id=ak, secret_access_key=sk, server=server)
try:
    # Specify the additional headers for object upload.
    headers = PutObjectHeader()
    # Specify the bucket name to examplebucket.
    bucketName = "examplebucket"
    # Specify the object name (the name after the file is uploaded).
    objectKey = "objectname"
    # Specify the local full path of the file or folder to be uploaded, for example, aa/bb.txt or aa/.
    file_path = 'localfile'
    #  Upload the file.
    resp = obsClient.putFile(bucketName, objectKey, file_path, headers)
    # If status code 2xx is returned, the API call succeeds. Otherwise, the API call fails.
    if resp.status < 300:
        print('Put File Succeeded')
        print('requestId:', resp.requestId)
        print('etag:', resp.body.etag)
        print('versionId:', resp.body.versionId)
        print('storageClass:', resp.body.storageClass)
    else:
        print('Put File Failed')
        print('requestId:', resp.requestId)
        print('errorCode:', resp.errorCode)
        print('errorMessage:', resp.errorMessage)
except:
    print('Put File Failed')
    print(traceback.format_exc())
```

Go

```
package main
import (
    "fmt"
    "os"
    "obs-sdk-go/obs"
    obs "github.com/huaweicloud/huaweicloud-sdk-go-obs/obs"
)
func main() {
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console.
    ak := os.Getenv("AccessKeyID")
    sk := os.Getenv("SecretAccessKey")
    // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1)
region is used here as an example. Replace it with the one in use.
    endPoint := "https://obs.ap-southeast-1.myhuaweicloud.com"
    // Enter the endpoint for the region where the bucket is created.
    endPoint := "https://your-endpoint"
    // Create an ObsClient instance.
    obsClient, err := obs.New(ak, sk, endPoint)
    if err != nil {
        fmt.Printf("Create obsClient error, errMsg: %s", err.Error())
    }
    input := &obs.PutFileInput{}
    // Specify a bucket name.
    input.Bucket = "examplebucket"
    // Set the object name to objectname (an example).
    input.Key = "objectname"
    // Specify the local file (localfile as an example).
    input.SourceFile = "localfile"
    // Upload the file.
    output, err := obsClient.PutFile(input)
    if err == nil {
        fmt.Printf("Put file(%s) under the bucket(%s) successful!\n", input.Key, input.Bucket)
        fmt.Printf("StorageClass:%s, ETag:%s\n",
            output.StorageClass, output.ETag)
        return
    }
    fmt.Printf("Put file(%s) under the bucket(%s) fail!\n", input.Key, input.Bucket)
    if obsError, ok := err.(obs.ObsError); ok {
        fmt.Println("An ObsError was found, which means your request sent to OBS was rejected with an error
response.")
        fmt.Println(obsError.Error())
    } else {
        fmt.Println("An Exception was found, which means the client encountered an internal problem when
attempting to communicate with OBS, for example, the client was unable to access the network.")
        fmt.Println(err)
    }
}
```

Downloading an Object

The following example shows how to download the **objectname** object from bucket **examplebucket**.

Java

```
import com.obs.services.ObsClient;
import com.obs.services.exception.ObsException;
import com.obs.services.model.ObsObject;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
public class GetObject001 {
    public static void main(String[] args) {
        // Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        // Obtain an AK/SK pair on the management console.
        String ak = System.getenv("ACCESS_KEY_ID");
        String sk = System.getenv("SECRET_ACCESS_KEY_ID");
        // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1)
region is used here as an example. Replace it with the one in use. To check the endpoint, see https://
support.huaweicloud.com/intl/en-us/usermanual-obs/obs_03_0312.html.
        String endPoint = "https://obs.ap-southeast-1.myhuaweicloud.com";
        // Enter the endpoint for the region where the bucket is created.
        String endPoint = "https://your-endpoint";

        // Create an ObsClient instance.
        // Use the permanent AK/SK pair to initialize the ObsClient.
        ObsClient obsClient = new ObsClient(ak, sk,endPoint);

        try {
            // Download the object using streaming.
            ObsObject obsObject = obsClient.getObject("examplebucket", "objectname");
            // Read the object content.
            System.out.println("Object content:");
            InputStream input = obsObject.getObjectContent();
            byte[] b = new byte[1024];
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            int len;
            while ((len = input.read(b)) != -1) {
                bos.write(b, 0, len);
            }
            System.out.println("getObjectContent successfully");
            System.out.println(new String(bos.toByteArray()));
            bos.close();
            input.close();
        } catch (ObsException e) {
            System.out.println("getObjectContent failed");
            // Request failed. Print the HTTP status code.
            System.out.println("HTTP Code:" + e.getResponseCode());
            // Request failed. Print the server-side error code.
            System.out.println("Error Code:" + e.getErrorCode());
            // Request failed. Print the error message.
            System.out.println("Error Message:" + e.getErrorMessage());
            // Request failed. Print the request ID.
            System.out.println("Request ID:" + e.getErrorRequestId());
            System.out.println("Host ID:" + e.getErrorHostId());
            e.printStackTrace();
        } catch (Exception e) {
            System.out.println("getObjectContent failed");
            // Print other error information.
            e.printStackTrace();
        }
    }
}
```

Python

```
from obs import GetObjectRequest
from obs import ObsClient
import os
import traceback

# Obtain an AK and SK pair using environment variables or import the AK and SK pair in other ways. Using 
hard coding may result in leakage.
# Obtain an AK/SK pair on the management console.
ak = os.getenv("AccessKeyID")
sk = os.getenv("SecretAccessKey")
# Set server to the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-
southeast-1) region is used here as an example. Replace it with the one in use.
server = "https://obs.ap-southeast-1.myhuaweicloud.com"
# Set server to the endpoint for the region where the bucket is created.
server = "https://your-endpoint"

# Create an ObsClient instance.
obsClient = ObsClient(access_key_id=ak, secret_access_key=sk, server=server)
try:
    # Specify the additional parameters for object download.
    getObjectRequest = GetObjectRequest()
    # Rewrite the Content-Type header in the response.
    getObjectRequest.content_type = 'text/plain'
    bucketName="examplebucket"
    objectKey="objectname"
    #Download the object using streaming.
    resp = obsClient.getObject(bucketName=bucketName,objectKey=objectKey, 
getObjectRequest=getObjectRequest, loadStreamInMemory=False)
    # If status code 2xx is returned, the API call succeeds. Otherwise, the API call fails.
    if resp.status < 300:
        print('Get Object Succeeded')
        print('requestId:', resp.requestId)
        # Read the object content.
        while True:
            chunk = resp.body.response.read(65536)
            if not chunk:
                break
            print(chunk)
        resp.body.response.close()
    else:
        print('Get Object Failed')
        print('requestId:', resp.requestId)
        print('errorCode:', resp.errorCode)
        print('errorMessage:', resp.errorMessage)
except:
    print('Get Object Failed')
    print(traceback.format_exc())
```

Go

```
package main
import (
    "fmt"
    "os"
    "obs-sdk-go/obs"
    obs "github.com/huaweicloud/huaweicloud-sdk-go-obs/obs"
)
func main() {
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard 
coding may result in leakage.
    //Obtain an AK/SK pair on the management console.
    ak := os.Getenv("AccessKeyID")
    sk := os.Getenv("SecretAccessKey")
    // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1)
region is used here as an example. Replace it with the one in use.
    endPoint := "https://obs.ap-southeast-1.myhuaweicloud.com"
    // Enter the endpoint for the region where the bucket is created.
    endPoint := "https://your-endpoint"
    // Create an ObsClient instance.
    obsClient, err := obs.New(ak, sk, endPoint)
    if err != nil {
        fmt.Printf("Create obsClient error, errMsg: %s", err.Error())
    }
    input := &obs.GetObjectInput{}
    // Specify a bucket name.
    input.Bucket = "examplebucket"
    // Specify the object (objectnameas an example) to download.
    input.Key = "objectname"
    // Download the object using streaming.
    output, err := obsClient.GetObject(input)
    if err == nil {
        // The output.Body must be closed after use to prevent connection leakage.
        defer output.Body.Close()
        fmt.Printf("Get object(%s) under the bucket(%s) successful!\n", input.Key, input.Bucket)
        fmt.Printf("StorageClass:%s, ETag:%s, ContentType:%s, ContentLength:%d, LastModified:%s\n",
            output.StorageClass, output.ETag, output.ContentType, output.ContentLength, output.LastModified)
        // Read the object content.
        p := make([]byte, 1024)
        var readErr error
        var readCount int
        for {
            readCount, readErr = output.Body.Read(p)
            if readCount > 0 {
                fmt.Printf("%s", p[:readCount])
            }
            if readErr != nil {
                break
            }
        }
        return
    }
    fmt.Printf("List objects under the bucket(%s) fail!\n", input.Bucket)
    if obsError, ok := err.(obs.ObsError); ok {
        fmt.Println("An ObsError was found, which means your request sent to OBS was rejected with an error
response.")
        fmt.Println(obsError.Error())
    } else {
        fmt.Println("An Exception was found, which means the client encountered an internal problem when
attempting to communicate with OBS, for example, the client was unable to access the network.")
        fmt.Println(err)
    }
}
```

Listing Objects

The following example shows how to list objects in bucket **examplebucket**:

Java

```
import com.obs.services.ObsClient;
import com.obs.services.exception.ObsException;
import com.obs.services.model.ObjectListing;
import com.obs.services.model.ObsObject;
public class ListObjects001 {
    public static void main(String[] args) {
        // Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        // Obtain an AK/SK pair on the management console.
        String ak = System.getenv("ACCESS_KEY_ID");
        String sk = System.getenv("SECRET_ACCESS_KEY_ID");
        // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1)
region is used here as an example. Replace it with the one in use.
```

```
        String endPoint = "https://obs.ap-southeast-1.myhuaweicloud.com";
        // Enter the endpoint for the region where the bucket is created.
        String endPoint = "https://your-endpoint";

        // Create an ObsClient instance.
        // Use the permanent AK/SK pair to initialize the ObsClient.
        ObsClient obsClient = new ObsClient(ak, sk,endPoint);

        try {
            // List objects.
            ObjectListing result = obsClient.listObjects("examplebucket");
            for (ObsObject obsObject : result.getObjects()) {
                System.out.println("listObjects successfully");
                System.out.println("ObjectKey:" + obsObject.getObjectKey());
                System.out.println("Owner:" + obsObject.getOwner());
            }
        } catch (ObsException e) {
            System.out.println("listObjects failed");
            // Request failed. Print the HTTP status code.
            System.out.println("HTTP Code:" + e.getResponseCode());
            // Request failed. Print the server-side error code.
            System.out.println("Error Code:" + e.getErrorCode());
            // Request failed. Print the error message.
            System.out.println("Error Message:" + e.getErrorMessage());
            // Request failed. Print the request ID.
            System.out.println("Request ID:" + e.getErrorRequestId());
            System.out.println("Host ID:" + e.getErrorHostId());
            e.printStackTrace();
        } catch (Exception e) {
            System.out.println("listObjects failed");
            // Print other error information.
            e.printStackTrace();
        }
    }
}
```

Python

```
from obs import ObsClient
import os
import traceback

# Obtain an AK and SK pair using environment variables or import the AK and SK pair in other ways. Using
hard coding may result in leakage.
# Obtain an AK/SK pair on the management console.
ak = os.getenv("AccessKeyID")
sk = os.getenv("SecretAccessKey")
# Set server to the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-
southeast-1) region is used here as an example. Replace it with the one in use.
server = "https://obs.ap-southeast-1.myhuaweicloud.com"
# Set server to the endpoint for the region where the bucket is created.
server = "https://your-endpoint"

# Create an ObsClient instance.
obsClient = ObsClient(access_key_id=ak, secret_access_key=sk, server=server)
try:
    bucketName = "examplebucket"
    # Specify an object prefix.
    prefix = 'test/'
    # Specify the maximum number (100 as an example) of objects to be listed at a time.
    max_keys = 100
    # List objects in the bucket.
    resp = obsClient.listObjects(bucketName, prefix, max_keys=max_keys, encoding_type='url')

    # If status code 2xx is returned, the API call succeeds. Otherwise, the API call fails.
    if resp.status < 300:
        print('List Objects Succeeded')
        print('requestId:', resp.requestId)
        print('name:', resp.body.name)
        print('prefix:', resp.body.prefix)
        print('max_keys:', resp.body.max_keys)
        print('is_truncated:', resp.body.is_truncated)
        index = 1
        for content in resp.body.contents:
            print('object [' + str(index) + ']')
            print('key:', content.key)
            print('lastModified:', content.lastModified)
            print('etag:', content.etag)
            print('size:', content.size)
            print('storageClass:', content.storageClass)
            print('owner_id:', content.owner.owner_id)
            print('owner_name:', content.owner.owner_name)
            index += 1
    else:
        print('List Objects Failed')
        print('requestId:', resp.requestId)
        print('errorCode:', resp.errorCode)
        print('errorMessage:', resp.errorMessage)
except:
    print('List Objects Failed')
    print(traceback.format_exc())
```

Go

```
package main
import (
    "fmt"
    "os"
    "obs-sdk-go/obs"
    obs "github.com/huaweicloud/huaweicloud-sdk-go-obs/obs"
)
func main() {
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard 
coding may result in leakage.
    //Obtain an AK/SK pair on the management console.
    ak := os.Getenv("AccessKeyID")
    sk := os.Getenv("SecretAccessKey")

    // Enter the endpoint for the region where the bucket is created. The CN-Hong Kong (ap-southeast-1) 
region is used here as an example. Replace it with the one in use.
    endPoint := "https://obs.ap-southeast-1.myhuaweicloud.com"
    // Enter the endpoint for the region where the bucket is created.
    endPoint := "https://your-endpoint"

    // Create an ObsClient instance.
    obsClient, err := obs.New(ak, sk, endPoint)
    if err != nil {
        fmt.Printf("Create obsClient error, errMsg: %s", err.Error())
    }
    input := &obs.ListObjectsInput{}
    // Specify a bucket name.
    input.Bucket = "examplebucket"
    // List objects in the bucket.
    output, err := obsClient.ListObjects(input)
    if err == nil {
        fmt.Printf("List objects under the bucket(%s) successful!\n", input.Bucket)
        fmt.Printf("RequestId:%s\n", output.RequestId)
        for index, val := range output.Contents {
            fmt.Printf("Content[%d]-OwnerId:%s, ETag:%s, Key:%s, LastModified:%s, Size:%d\n",
                index, val.Owner.ID, val.ETag, val.Key, val.LastModified, val.Size)
        }
        return
    }
    fmt.Printf("List objects under the bucket(%s) fail!\n", input.Bucket)
    if obsError, ok := err.(obs.ObsError); ok {
        fmt.Println("An ObsError was found, which means your request sent to OBS was rejected with an error 
response.")
        fmt.Println(obsError.Error())
    } else {
        fmt.Println("An Exception was found, which means the client encountered an internal problem when attempting to communicate with OBS, for example, the client was unable to access the network.")
        fmt.Println(err)
    }
}
```

Related Information

In addition to the operations mentioned above, the following advanced functions of OBS for Java SDK are available for you to use.

- **Lifecycle management**: You can configure lifecycle rules to periodically transition objects between storage classes or delete objects.
- **Bucket ACL**: You (the bucket owner) can write the bucket ACL using the SDK for Java to implement fine-grained access control.
- **Bucket policies**: You (the bucket owner) can write bucket policies using the SDK for Java to implement fine-grained access control.
- **Static website hosting**: You can upload a static website file to an OBS bucket, grant anonymous users the read permission for this file, and configure static website hosting for the bucket, so that you can use the bucket domain name to access the hosted static website file.
- **Versioning**: You can enable versioning for a bucket to keep multiple versions of an object in the same bucket. This way, you can easily retrieve and restore each object version and recover data from unintended actions or application failures.
- **Server-side encryption**: You can use server-side encryption to encrypt data uploaded to OBS buckets.
- **Cross-origin resource sharing (CORS)**: You can configure a CORS rule to enable cross-domain quests for OBS.

5 Common Tasks

After you perform basic operations, such as create a bucket, upload and download an object, you can experience common OBS tasks.

Table 5-1 Common tasks

Task		Description
Data security	**Enterprise Data Access Control**	After subscribing to OBS, enterprises can control access to their data: ● Grant different departments with only required permissions to isolate access to the enterprise data. ● Grant permissions to users from a different department or project to download the shared data but not to write or delete the shared data. ● Grant each department with required IAM user permissions and use bucket policies to grant the IAM users independent permissions on resources. ● Add external buckets on OBS Browser + to isolate bucket resources between departments.
Data migration	**Migrating Local Data to OBS**	Huawei Cloud offers diverse solutions for migrating data from on-premises storage servers to OBS in a cost-effective, secure, and efficient way. You can choose a migration solution based on your data volume, time arrangement, and budget.

Task		Description
	Migrating Data from Third-Party Cloud Service Vendors to OBS	With Huawei Cloud Object Storage Migration Service (OMS), you can easily migrate data from a third-party cloud vendor to OBS by only configuring connection parameters and migration tasks on OBS Console.
Data backup	**Using Backup Software to Back Up Local Data to OBS**	Third-party backup software, such as Commvault and AnyBackup Cloud, can connect to OBS for data backup. You can customize policies for secure and efficient backups.
Data access	**Accessing OBS from an ECS over the Intranet**	You can read, back up, and archive data on OBS from your ECS over an intranet.
	Accessing OBS Through an NGINX Reverse Proxy	For security purposes, some enterprises need to configure a blacklist or whitelist for external addresses, so a fixed IP address is required for accessing OBS. However, Huawei Cloud DNS always resolves the bucket's access domain name to different IP addresses for secure access. To address this problem, you can set up an NGINX reverse proxy server on an ECS to allow users to access OBS with a fixed IP address.
	Using a User-Defined Domain Name to Host a Static Website	If a company has a large number of static websites but does not want to set up servers, the company can host these websites in an OBS bucket, so that users can access the websites using the domain name bound to the OBS bucket.
Data transmissio n	**Using the PostObject API to Upload Data from a Web Client to OBS**	PostObject API can directly upload files from a web client to OBS, which is called browser-based upload. With this method, you do not have to upload data to the app server before uploading data to OBS. This makes data transmission faster and does not impose pressure on the server. Additionally, direct transmission with a signature returned by the server is more secure.

Task		Description
	Uploading Data from Mobile Apps to OBS	OBS is widely used as the storage for Android phones and iOS apps. To protect application data from leakage and unauthorized access, you are advised to use a temporary security credential or a presigned URL to upload data to OBS.
	Uploading Data from Mini Programs to OBS	Mini programs are now popular in a variety of scenarios. You can upload data to OBS from a mini program.
Data processing	**CDN for Download Acceleration**	This solution automatically caches data stored in OBS on demand to CDN points of presence (PoPs) in different regions, accelerating the access to and download of static resources.