

# Alibaba Cloud

## Object Storage Service Best Practices

Document Version: 20220711

# Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

# Document conventions

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

---

# Table of Contents

1.Overview .....	07
2.Websites and mobile apps .....	08
2.1. Upload data to OSS through Web applications .....	08
2.1.1. Overview .....	08
2.1.2. Use PostObject on the web to upload objects .....	09
2.1.3. Add signatures on the client by using JavaScript and u... ..	09
2.1.4. Obtain signature information from the server and uplo... ..	14
2.1.5. Add signatures on the server, configure upload callbac... ..	16
2.1.5.1. Overview .....	16
2.1.5.2. Python .....	23
2.1.5.3. Java .....	29
2.1.5.4. Go .....	36
2.1.5.5. Node.js .....	40
2.1.5.6. .NET .....	44
2.1.5.7. PHP .....	50
2.1.5.8. Ruby .....	55
2.2. Application server .....	59
2.2.1. Set up direct data transfer for mobile apps .....	59
2.2.2. Set up upload callback for mobile apps .....	64
2.3. Use ECS instances to configure reverse proxy for access to... ..	69
2.3.1. Use an ECS instance that runs Ubuntu to configure a ... ..	69
2.3.2. Use an ECS instance that runs CentOS to configure a ... ..	71
2.3.3. Use an ECS instance that runs Windows to configure a... ..	74
2.4. Check data transmission integrity by using CRC-64 .....	77
3.Data lake .....	80
3.1. Alibaba Cloud ecosystem .....	80

3.1.1. Configure Apache Ranger for authentication in EMR clu...	80
3.1.2. Use JindoFuse to access the OSS-HDFS service	82
3.1.3. Use MaxCompute to build a data warehouse based on ...	88
3.2. Open source ecosystem	90
3.2.1. Use self-managed Hadoop to access OSS-HDFS	90
3.2.2. Use HDP 2.6-based Hadoop to read and write OSS da...	99
3.2.3. Use JindoSDK with Apache Flink to process data store...	104
3.2.4. Use OSS-HDFS as the underlying storage of HBase	106
3.2.5. Use JindoSDK with Hive to process data stored in OSS...	108
3.2.6. Use JindoSDK with Impala to query data stored in OS...	111
3.2.7. Use JindoSDK with Spark to process data stored in OS...	113
3.2.8. Use JindoSDK with Presto to query data stored in OSS...	115
3.2.9. Use Kite SDK with Apache Sqoop to read and write d...	117
4.Content distributing and data processing	120
4.1. Create HLS streams based on OSS	120
4.2. Use CDN to accelerate access to OSS	124
5.Data backup and recovery	130
5.1. Back up buckets	130
6.Cost management	131
6.1. Configure lifecycle rules to manage object versions	131
7.Migrate data to OSS	135
7.1. Migrate data between OSS	135
7.1.1. Overview	135
7.1.2. Use CRR or SRR to migrate OSS data owned by the sa...	135
7.1.3. Use Data Online Migration to migrate data between OS...	137
7.2. Migrate data sources from a third party to OSS	141
7.3. Seamlessly migrate data from Amazon S3 to Alibaba Cloud..	142
7.4. Use ossimport to migrate data	145

- 7.5. Migrate data from HDFS to OSS ..... 147
- 8.Security ..... 158
  - 8.1. Reduce the risks of unauthorized access caused by AccessK..... 158
  - 8.2. Reduce the risks of unexpectedly high fees caused by mal..... 161
  - 8.3. Reduce the risks of data loss caused by accidental operati..... 164
  - 8.4. Sensitive data protection ..... 166
- 9.IaC automation ..... 170
  - 9.1. Terraform ..... 170
    - 9.1.1. Overview ..... 170
    - 9.1.2. Use Terraform to manage OSS ..... 171
- 10.Others ..... 176
  - 10.1. Use Function Compute to download multiple objects as a..... 176
  - 10.2. Use CloudMonitor to monitor OSS throttling information ... 179
  - 10.3. OSS performance and scalability best practices ..... 184

# 1. Overview

Alibaba Cloud Object Storage Service (OSS) Best Practices help you use OSS more efficiently to meet your business needs for operations such as data migration, data backup and disaster recovery (BDR), direct data transfer to or out of OSS, data processing and analysis, audio and video transcoding, and using Terraform to manage OSS.

## Migrate data to OSS

- [Migrate data between buckets in OSS](#)
- [Migrate data sources from a third party to OSS](#)
- [Seamlessly migrate data from Amazon S3 to Alibaba Cloud OSS](#)
- [Use ossimport to migrate data](#)

## BDR

[Back up buckets](#)

## Direct data transfer to or out of OSS

- [Upload data to OSS through Web applications](#)
- [Set up direct data transfer for mobile apps](#)

## Performance and scalability

- [OSS performance and scalability best practices](#)
- [Use CDN to accelerate access to OSS](#)

## Use ECS instances to configure reverse proxy for access to OSS

- [Use an ECS instance that runs CentOS to configure a reverse proxy for access to OSS](#)
- [Use an ECS instance that runs Ubuntu to configure a reverse proxy for access to OSS](#)

## Terraform

[Use Terraform to manage OSS](#)

# 2. Websites and mobile apps

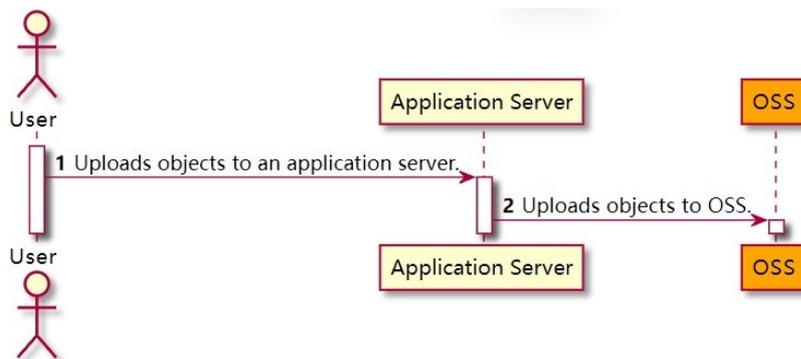
## 2.1. Upload data to OSS through Web applications

### 2.1.1. Overview

This topic describes how to use browsers to upload data to Object Storage Service (OSS).

#### Background information

When users use browsers or an application to upload an object, the object is generally uploaded first to the application server. Then, the application server uploads the object to OSS. The following figure shows the specific process.



Compared with direct data transfer, the preceding method has the following disadvantages:

- Time-consuming: Data is uploaded to the application server before the data can be uploaded to OSS. The consumed time period is greatly shortened if data is directly transferred to OSS from the client. In addition, OSS uses the Border Gateway Protocol (BGP) network to ensure high speed transmission of data between different Internet service providers (ISPs).
- Poor scalability: The performance of the application server can hit a bottleneck if the number of users increases.
- High cost: Multiple application servers must be prepared. You can save costs on additional application servers by using direct data transfer to OSS. In addition, you are not charged for the traffic generated by data upload.

#### Technical solutions

Currently, we provide the following two solutions for data transfer by using browsers:

- Use OSS SDK for Browser.js to upload objects to OSS

This solution uses OSS SDK for Browser.js to directly transfer data to OSS. For more information, see [Overview](#). You can use resumable upload to upload large objects when the network conditions are poor. This solution is compatible and supported by the following browsers: IE 10 and later, major versions of Microsoft Edge, Google Chrome, Firefox, and Safari, and most browsers for mobile phones that run Android, iOS, and Windows Phone.

- Use form upload to upload objects to OSS

Call the `PostObject` operation provided by OSS to upload an object to OSS by using form upload. This solution is supported by most browsers. However, you can only retry the upload when you fail to upload an object due to poor network conditions. For more information, see [Use `PostObject` on the web to upload objects](#).

## 2.1.2. Use `PostObject` on the web to upload objects

This topic describes how to use form upload on the web to directly upload data to OSS.

When you use the web to upload an object, you can use the browser or app to upload the object to an application server. The application server uploads the object to OSS. During this process, the application server is necessary to transfer the object, which delivers lower transmission efficiency than direct data upload.

Direct upload allows you to call the `PostObject` operation provided by OSS to upload data by using form upload. The following cases describe how to directly upload data to OSS by using form upload:

- On the client, add signatures by using JavaScript, and use form upload to upload data to OSS. For more information, see [Add signatures on the client by using JavaScript and upload data to OSS](#).
- Add signatures on the server. Use form upload to upload data to OSS. For more information, see [Add signatures on the server for object upload](#).
- Add signatures on the server. Configure upload callback on the server. Use form upload to upload data to OSS. After OSS receives the callback response, OSS returns the response to the client. For more information, see [Add signatures on the server, configure upload callback, and directly transfer data](#).

## 2.1.3. Add signatures on the client by using JavaScript and upload data to OSS

This topic describes how to use JavaScript to add signatures on the client based on POST policies and then upload data to Object Storage Service (OSS) by using form upload.

### Usage notes

- After you use JavaScript to add signatures on the client, you can directly upload data to OSS. However, your AccessKey ID and AccessKey secret may be exposed because they are included in the JavaScript code. We recommend that you add signatures on the server to upload data. For more information, see [Add signatures on the server for object upload](#) or [Use a temporary credential provided by STS to access OSS](#).
- The application server code provided in this topic supports protocols such as HTML 4, HTML 5, Flash, and Silverlight. Make sure that your browser supports these protocols. If a prompt such as "Your browser does not support Flash, Silverlight, or HTML5!" appears, upgrade your browser.

### Step 1: Download the client code running on browsers

Plupload is used in this example to upload form data (`PostObject`) to OSS. The sample code in JavaScript can be run on WeChat or browsers from computers or mobile phones. You can select multiple objects for uploads and specify the destination directory. In addition, you can specify whether the names of the objects to upload are the same as the names of the local files that you upload or are randomly specified by OSS. You can view the upload progress by using the progress bar.

1. Download the [browser client code](#).
2. Decompress the downloaded file.

 **Note** The frontend plug-in used in this example is Plupload. You can change the plug-in.

## Step 2: Modify the configuration file

Open the `upload.js` file and modify access configurations.

```
// Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account to access OSS because the account has permissions on all API operations. We recommend that you use a Resource Access Management (RAM) user to call API operations or perform routine O&M. To create a RAM user, log on to the RAM console.
accessid= '<yourAccessKeyId>';
accesskey= '<yourAccessKeySecret>';
host= '<yourHost>';
.....
new_multipart_params = {
    ....
    'OSSAccessKeyId': accessid,
    ....
};
// If you use Security Token Service (STS) access credentials to access OSS, modify the configurations.
accessid= 'accessid';
accesskey= 'accesskey';
host = 'http://post-test.oss-cn-hangzhou.aliyuncs.com';
STS_ROLE = ''; // eg: acs:ram::1069test7698:role/all
.....
new_multipart_params = {
    ....
    'OSSAccessKeyId': STS.accessID,
    'x-oss-security-token':STSToken,
    ....
};
//=====
host = 'http://post-test.oss-cn-hangzhou.aliyuncs.com';
```

- `accessid`: the AccessKey ID of your RAM user.
- `accesskey`: the AccessKey secret of your RAM user.
- `STS_ROLE`: the Alibaba Cloud Resource Name (ARN) of the specified role. For more information about RoleArn, see [AssumeRole](#).
- `STSToken`: your STS token. If you use STS for authentication, you must call the STS API operation to obtain the STS AccessKey ID, STS AccessKey secret, and security token. For more information, see [Use a temporary credential provided by STS to access OSS](#). If your AccessKey ID and AccessKey secret are

from an Alibaba Cloud account or a RAM user that has permanent permissions, you can leave this parameter empty.

- **host**: your OSS endpoint used to access the bucket. The format is `BucketName.Endpoint`. Example: `post-test.oss-cn-hangzhou.aliyuncs.com`. For more information about OSS endpoints, see [OSS domain names](#).

### Step 3: Configure CORS

When you use form upload to upload data from the client to OSS, the client includes the `Origin` header in the request and sends the request from the browser to OSS. OSS verifies whether a request that includes the `Origin` header is a cross-origin request. Therefore, you must configure cross-origin resource sharing (CORS) rules for the destination bucket to allow POST-based requests.

1. Log on to the [OSS console](#).
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**. The following figure shows the configurations of a rule.

**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

### Step 4: Use JavaScript to add signatures on the client and upload data to OSS

1. Open the `index.html` file in the decompressed directory of the client code.
2. Click **Select File**. Select one or more files to upload. Then, select a naming convention for the object to upload, and specify the directory in which you want to store the objects.

3. Click **Upload**. Wait until the uploads are complete.
4. After the uploads are complete, you can log on to the OSS console to view the upload results.

## Core code analysis

OSS supports POST requests. Therefore, you need to add a signature to a POST request when you use Plupload to send the request. The following code provides an example on the core code:

```
function set_upload_param(up, filename, ret)
{
    g_object_name = g_dirname;
    if (filename != '') {
        suffix = get_suffix(filename)
        calculate_object_name(filename)
    }
    new_multipart_params = {
        'key' : g_object_name,
        'policy': policyBase64,
        'OSSAccessKeyId': accessid,
        'success_action_status' : '200', // If you do not set success_action_status to 200, 204 status code is returned when the object is uploaded.
        'signature': signature,
    };
    up.setOption({
        'url': host,
        'multipart_params': new_multipart_params
    });
    up.start();
}
....
```

In the preceding code, `'key': g_object_name` indicates the path of the uploaded object. If you want to retain the original name for the uploaded object, set this field to `'key': '${filename}'`.

To upload objects to a specified directory such as the abc directory without changing the object names, change part of the preceding code to the following code:

```
new_multipart_params = {
    'key' : 'abc/' + '${filename}',
    'policy': policyBase64,
    'OSSAccessKeyId': accessid,
    'success_action_status' : '200', // Set success_action_status to 200 so that 200 is returned if the request is successful. By default, if this option is not specified, 204 is returned.
    'signature': signature,
};
```

- Specify that uploaded object names are randomly specified by the system

To instruct the system to randomly specify the name of an uploaded object without changing its name extension, use the following code to modify the function in the preceding code:

```
function check_object_radio() {
    g_object_name_type = 'random_name';
}
```

- Specify that the names of uploaded objects remain unchanged

To retain the original names of the objects, use the following code to modify the function in the preceding code:

```
function check_object_radio() {
    g_object_name_type = 'local_name';
}
```

- Configure the destination directory

You can upload objects to a specified directory. The following code provides an example on how to set the destination directory to `abc/`. The destination directory must end with a forward slash (`/`).

```
function get_dirname()
{
    g_dirname = "abc/";
}
```

- Sign policyText

When you upload objects, you must sign policyText. Example:

```
var policyText = {
    "expiration": "setDurationSeconds", // Set a proper validity period for the policy in
    UTC. After the validity period ends, you can not upload objects to OSS by using this poli
    cy.
    "conditions": [
        ["content-length-range", 0, 1048576000] // Set the size limit for the uploaded object
        . If the uploaded object exceeds the size limit, OSS reports an error.
    ]
}
```

## FAQ

- How do I specify the format of an object to upload?

You can set conditions for objects to upload by using the filters attributes of Plupload. For example, you can set conditions on the type of images and the size of objects. For more information, see [Overview](#).

- How do I obtain the URL of an uploaded object?

You can obtain the URL of an uploaded object based on the endpoint of the destination bucket and the object path. For more information, see [How do I obtain the URL of an uploaded object?](#).

- How do I obtain the MD5 hash of an uploaded object?

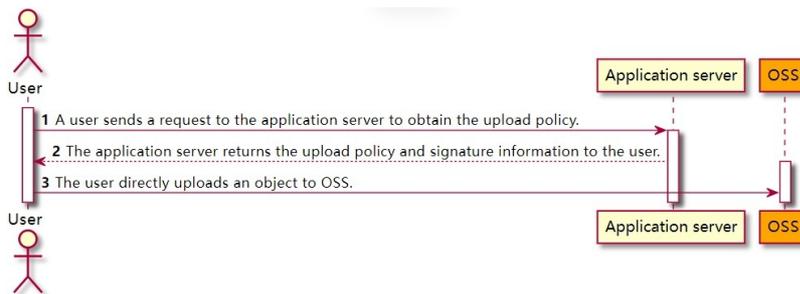
Open the developer tools of your browser, and then upload an object. After the object is uploaded, you can view the MD5 hash in the response header.

If you have other questions, [submit a ticket](#) for help.

## 2.1.4. Obtain signature information from the server and upload data to OSS

This topic describes how to obtain signature information from the server in various programming languages based on POST policies and directly upload data to Object Storage Service (OSS) by using form upload. In this method, the AccessKey pair used to generate the signature is not included in the code of the client. Therefore, this method is more secure than the method in which the signature is generated by a JavaScript client.

### Process and code analysis



1. A user sends a request to the application server to obtain the upload policy.

In the `upload.js` file of the client source code package, set the value of the `serverUrl` variable in the following snippet to the URL of the application server.

```

// serverUrl specifies the URL of the application server that returns signature information and upload policies. Replace the sample IP address and the port number with actual values.
serverUrl = 'http://88.88.88.88:8888'

```

The client sends GET requests to the application server whose URL is specified by `serverUrl` to obtain signature information and upload policies. You can download the client source code from the following address: [aliyun-oss-appserver-js-master.zip](http://aliyun-oss-appserver-js-master.zip).

Upload callback is not involved in the scenario described in this topic. Therefore, you must comment out the `'callback' : callbackbody` field in the `upload.js` file of the client source code to disable the upload callback feature.

```

{
  'key' : key + '${filename}',
  'policy': policyBase64,
  'OSSAccessKeyId': accessid,
  // Set the status code returned by the server to 200. By default, the 204 status code is returned.
  'success_action_status' : '200',
  'callback' : callbackbody,
  'signature': signature,
}

```

2. The application server returns the upload policy and signature to the user.

A service is deployed on the application server to respond to the GET request sent by the client and return the signature information that is required for object upload. You can modify the code of the service so that the application server returns correct information to the client.

The following code provides an example on the body content returned to the client by the application server:

```
{
  "accessid": "LTAI5tBDFVarlhoq****",
  "host": "http://post-test.oss-cn-hangzhou.aliyuncs.com",
  "policy": "eyJleHBpcmF0aW9uIjoiMjAxNS0xMS0wNVQyMDoyMzoyMloiLCJxb25kaXRpb25zIjpbWyJjcb250ZW50LWxlbmd0aC1yYW5nZSIsMCwxMDQ4NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwiaWJGtleSIsInVzZXItZGlyXC8i****",
  "signature": "Vsx0cOudx*****z93CLaXPz+4s=",
  "expire": 1446727949,
  "dir": "user-dirs/"
}
```

The following table describes the fields that are contained in the body.

Field	Description
accessid	The required AccessKey ID.
host	The domain name from which the user sends the upload request.
policy	The policy for form upload. The policy is a Base64-encoded string. For more information, see <a href="#">PostObject</a> .
signature	The signature string of the policy. For more information, see the Post Signature section in <a href="#">PostObject</a> .
expire	The expiration time of the policy specified by the server, which is in the UNIX timestamp format (the number of seconds that have elapsed since January 01, 1970 00:00:00 UTC).
dir	The prefix of objects that are allowed to be uploaded.

### 3. The user directly sends an object upload request to OSS.

```
new_multipart_params = {
    // key specifies the full path of the object in the bucket. Example: exampledir/exampleobject.txtObject. The path cannot contain the bucket name.
    // filename specifies the name of the local file to upload.
    'key' : key + '${filename}',
    'policy': policyBase64,
    'OSSAccessKeyId': accessid,
    // Set the returned status code by the server to 200. By default, the returned status code is 204.
    'success_action_status' : '200',
    'signature': signature,
};
```

## Sample code

For more information about the code for various programming languages that are used to obtain signature information from the server, configure upload callback, and directly upload data to OSS, see the following topics:

- [Java](#)
- [Python](#)
- [PHP](#)
- [Go](#)
- [Node.js](#)
- [Ruby](#)
- [.NET](#)

## References

In general, the application server needs to be informed of the information about uploaded objects, such as the users who upload the objects and the names of the uploaded objects. If a user uploads an image, the application server needs to be informed of the image size. You can configure upload callback to meet these requirements. For more information, see [Overview](#).

## 2.1.5. Add signatures on the server, configure upload callback, and directly transfer data

### 2.1.5.1. Overview

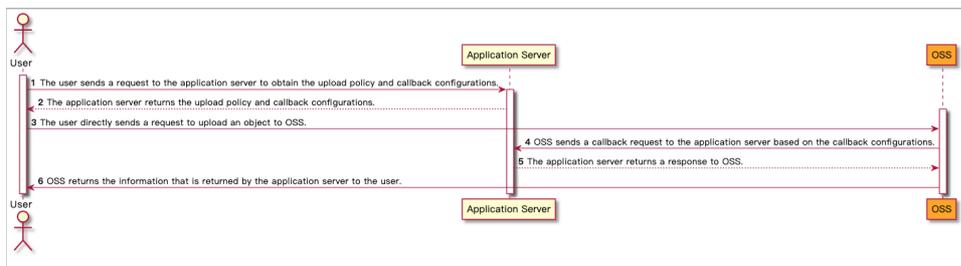
This topic describes how to obtain signature information from the server in various programming languages based on POST policies, configure upload callbacks, and then use form upload to upload data to Object Storage Service (OSS).

### Background information

When users upload data by using the solution that is described in [Obtain signature information from the server and upload data to OSS](#), the information about the upload and names of the objects must be sent to the application server. When users upload images, information about the sizes of the images must be sent to the application server. To meet this requirement, OSS provides the upload callback feature.

### Process

The following figure shows how upload callback works.



If a user wants the application server to receive an upload callback request when the user uploads an object to OSS, configure a callback function to ensure that OSS sends the upload callback request to the application server. After the user uploads the object, the application server returns a response for the upload callback request to OSS. Then, OSS forwards the response to the user. This response is the upload result.

## Examples

To obtain signature information from the server, configure upload callbacks, and then use form upload to upload data to OSS by using one of the following programming languages:

- PHP
- Java
- Python
- Go
- Node.js
- .NET
- Ruby

## Process analysis

The following process describes the core code that is used and the messages that can be returned.

1. A user sends a request to the application server to obtain the upload policy and callback configurations.

In the `upload.js` file of the client source code, the value of the `serverUrl` variable can be used to configure the URL of the application server. After you configure the URL of the application server, the client sends a GET request to `serverUrl` to obtain the required information.

```
// serverUrl specifies the URL of the application server that returns information about
// the signature and upload policies. Replace the sample IP address and port number with a
// ctual values in your business scenario.
serverUrl = 'http://88.88.88.88:8888'
```

2. The application server returns the upload policy and the code that is used to configure upload callbacks to the user.

The application server processes the GET request that is sent from the client for direct data transfer based on the services that are used to obtain information about the signature of the server. You can modify the corresponding code to ensure that the application server returns the correct message to the client. The configuration documents for different programming languages provide clear instructions for your reference.

The following example describes a message body that is returned to a client. The message body is used as an important parameter to upload an object from the client.

```
{
  "accessid": "LTAI5tAzivUnv4ZF1azP****",
  "host": "http://post-test.oss-cn-hangzhou.aliyuncs.com",
  "policy": "eyJleHBpcmF0aW9uIjoiMjAxNS0xMS0wNVQyMDoyMzoyMloiLCJxb25kaXRpb25zIjpbWyJjcb250ZW50LWxlbnmd0aC1yYW5nZSIsMCwxMDQ4NTc2MDAwXSxbInN0YXJ0cy13aXRoIiwiaWJGtleSIsInVzZXItZGlyXC8i****",
  "signature": "I2u57FWjTKqX/AE6doIdyff1****",
  "expire": 1446727949,
  "callback": "eyJjYWxsYmFja1VybCI6Imh0dHA6Ly9vc3MtZGVtby5hbG15dW5jcy5jb206MjMONTAiLAoiY2FsbGJhY2tCb2R5IjoiZmlsZW5hbWU9JHtvYmplY3R9JnNpemU9JHtzaXplfSZtaW1lVHlwZT0ke21pbWVUeXB1fSZoZWlnaHQ9JHtpbWFnZUluZm8uaGVpZ2h0fSZ3aWR0aD0ke21tYWdlSW5mby53aWR0aH0iLAoiY2FsbGJhY2tCb2R5VHlwZSI6ImFwcGxpY2F0aW9uL3gtZ3d3LWZvc0tdXJsZW5jb2RlZCJ9",
  "dir": "user-dirs/"
}
```

The content that is returned for a callback in the preceding example is encoded in Base64. The following code shows the content decoded in Base64:

```
{"callbackUrl": "http://oss-demo.aliyuncs.com:23450",
"callbackBody": "filename=${object} &size=${size} &mimeType=${mimeType} &height=${imageInfo.height} &width=${imageInfo.width}",
"callbackBodyType": "application/x-www-form-urlencoded"}
```

**Note** The callback method that is used in the preceding example is provided only for reference. You can configure a callback method by modifying the server code.

Parameter	Description
callbackUrl	The URL of the application server to which OSS sends the request.
callbackHost	The Host header that is included in the request that is sent by OSS to the application server.
callbackBody	The content that OSS sends to the application server. The content can be an object or an image. You can send information about the name, size, or type of an object. You can send information about the height and width of an image.
callbackBodyType	The content type of the request.

3. The user sends an object upload request to OSS.

In the `upload.js` file of the client source code, the value of `callbackbody` is included in the callback body that is returned by the application server to the client in [Step 2](#).

```

new_multipart_params = {
    'key' : key + '${filename}',
    'policy': policyBase64,
    'OSSAccessKeyId': accessid,
    // Set the status code that is returned by the server to 200. If you do not config
    ure this parameter, the 204 status code is returned.
    'success_action_status' : '200',
    'callback': callbackbody,
    'signature': signature,
};

```

#### 4. OSS sends a callback request to the application server based on callback configurations.

After you upload the object to OSS, OSS analyzes the upload callback configurations of the client and sends the POST callback request to the application server. The following sample code shows the content of the request message:

```

Hypertext Transfer Protocol
POST / HTTP/1.1\r\n
Host: 47.97.168.53\r\n
Connection: close\r\n
Content-Length: 76\r\n
Authorization: fsNxFF0w*****MNAoFb//a8x6v2lI1*****h3nFUDALgku9bhC+cWQsnxuCo*****
tBUmnDI6k1PofggA4g==\r\n
Content-MD5: eiEMyp7lbL8KStPBzMdr9w==\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Date: Sat, 15 Sep 2018 10:24:12 GMT\r\n
User-Agent: aliyun-oss-callback\r\n
x-oss-additional-headers: \r\n
x-oss-bucket: signedcallback\r\n
x-oss-owner: 137918634953****\r\n
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNvbS9jYWxsYmFja19wdWJfa2V5X3Yx
LnaH****\r\n
x-oss-request-id: 534B371674E88A4D8906****\r\n
x-oss-requester: 137918634953****\r\n
x-oss-signature-version: 1.0\r\n
x-oss-tag: CALLBACK\r\n
eagleeye-rpcid: 0.1\r\n
\r\n
[Full request URI: http://47.xx.xx.53/]
[HTTP request 1/1]
[Response in frame: 39]
File Data: 76 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "filename" = ".snappython.png"
Form item: "size" = "6014"
Form item: "mimeType" = "image/png"
Form item: "height" = "221"

```

#### 5. The application server returns a response to OSS.

The application server verifies the access credentials based on the `authorization` method that is included in the message that is sent by OSS. If the verification is successful, the application server returns the following message to OSS in the JSON format :

```
{
  "String value": "ok",
  "Key": "Status"
}
```

6. OSS forwards the response from the application server to the user.

## Analysis of client source code

To download the client source code, click [aliyun-oss-appserver-js-master.zip](#).

**Note** Plupload is used for the JavaScript code of the client. Plupload is a simple, easy-to-use, and powerful tool that provides advanced features to allow you to upload files. The tool supports multiple upload methods, including uploads by using HTML, Flash, Silverlight, and HTML4. Plupload detects the current environment and selects the most suitable upload method. The tool assigns the highest priority to HTML5 for uploads. For more information about Plupload, visit [Plupload.com](#).

The following sample code describes common features:

- Specify random names for the objects that you upload

If you want to specify random object names without changing the name extension, you can use the following code to modify the function:

```
function check_object_radio() {
  g_object_name_type = 'random_name';
}
```

- Retain the original object names

If you want to retain the original object names, you can use the following code to modify the function:

```
function check_object_radio() {
  g_object_name_type = 'local_name';
}
```

- Configure the directory to which you want to upload the object

The directory to which the object is uploaded is determined by the server. You can upload objects only to a specified directory. This helps ensure data isolation. The following code shows how to set the path of the directory to abc/ by using PHP. The path of the directory must end with a forward slash (/).

```
$dir = 'abc/';
```

- Configure upload conditions

You can use Plupload filters to configure upload conditions, such as uploading only images, configuring limits for the size of objects that you can upload, and preventing repeated uploads of an object.

```

var uploader = new plupload.Uploader({
  ...
  filters: {
    mime_types : [
      // Specify that only images and ZIP objects can be uploaded.
      { title : "Image files", extensions : "jpg,gif,png,bmp" },
      { title : "Zip files", extensions : "zip" }
    ],
    // Specify that only objects that are smaller than 400 KB in size can be uploaded
    .
    max_file_size : '400kb',
    // Specify that the repeat upload of an object is denied.
    prevent_duplicates : true
  },

```

- `mime_types`: specifies the extensions of the objects that you want to upload.
- `max_file_size`: specifies the maximum size of the objects that you want to upload.
- `prevent_duplicates`: specifies that an object cannot be repeatedly uploaded.
- Obtain the names of uploaded objects

You can obtain the names of the objects that are uploaded by using Plupload to call the `FileUploaded` event.

```

FileUploaded: function(up, file, info) {
    if (info.status == 200)
    {
        document.getElementById(file.id).getElementsByTagName('b')[0].innerHTML =
'upload to oss success, object name:' + get_uploaded_object_name(file.name);
    }
    else
    {
        document.getElementById(file.id).getElementsByTagName('b')[0].innerHTML =
info.response;
    }
}

```

You can use the `get_uploaded_object_name(file.name)` function to obtain the names of the objects that are uploaded to OSS. The `file.name` parameter specifies the names of the objects before the objects are uploaded.

- Upload the signature

You can use the following sample core code to obtain the `policyBase64`, `accessid`, and `signature` variables from the server by using JavaScript.



### Policy analysis:

```
{ "expiration": "2015-11-05T20:23:23Z",  
  "conditions": [ [ "content-length-range", 0, 1048576000 ],  
                  [ "starts-with", "$key", "user-dir/" ] ] }
```

In the preceding example, the `starts-with` field that is added to the policy specifies that the object name must start with `user-dir`. You can configure this field. The `starts-with` field is added to the policy because each application corresponds to a bucket in most scenarios. To prevent data from being overwritten, you can specify a prefix for the name of each object that you upload to OSS. After the user obtains the policy, the user can upload multiple objects, change the prefix of an object, and upload the object to the directory that is accessed by another user when the policy is valid. To resolve this issue, the prefix is specified by the application server. This way, even if the user obtains the policy, the user cannot upload the object to the directory that is accessed by another user. This helps ensure data security.

- Configure the URL of the application server

In the `upload.js` file of the client source code, the value of the `serverUrl` variable can be used to configure the URL of the application server. After you configure the URL of the application server, the client sends a GET request to `serverUrl` to obtain the required information.

```
// serverUrl specifies the URL of the application server that returns information about the  
signature and upload policies. Replace the sample IP address and port number with actual  
values in your business scenario.  
serverUrl = 'http://88.88.88.88:8888'
```

## FAQ

How do I upload multiple objects to OSS at a time?

The service does not support API operations for uploading multiple objects to OSS at a time. If you want to upload multiple objects to OSS at a time, you can repeatedly perform the steps for uploading a single object.

### 2.1.5.2. Python

This topic describes how to calculate signatures by using Python code on the server, configure upload callback, and then use form upload to upload data to Object Storage Service (OSS).

#### Prerequisites

- The domain name of the application server can be accessed over the Internet.
- Python 2.6 or later is installed on the application server. To view the Python version, run the `python --version` command.
- JavaScript is supported by the browser on your PC.

#### Step 1: Configure the application server

1. Download the [application server source code](#) package in Python.
2. In this example, Ubuntu 16.04 is used. Decompress the source code package to the `/home/aliyun/aliyun-oss-appserver-python` directory.
3. Go to the directory, open the `appserver.py` file, and then modify the following snippet:

```
# The AccessKey pair of an Alibaba Cloud account has permissions on all API operations.
Using these credentials to access OSS is a high-risk operation. We recommend that you use
a RAM user to call API operations or perform routine O&M. To create a RAM user, log on
to the RAM console.
access_key_id = 'yourAccessKeyId'
access_key_secret = 'yourAccessKeySecret'
# Specify the access address of the host in the https://bucketname.endpoint format.
host = 'https://examplebucket.oss-cn-hangzhou.aliyuncs.com';
# Specify the URL of the application server to which an upload callback request is sent
. This URL is used for communication between the application server and OSS. After you
upload an object, OSS uses the URL to send information about the uploaded object to the
application server.
callback_url = "https://192.168.0.0:8888";
# Specify the directory in which you want to store uploaded objects. You can also leave
this parameter empty. If you do not specify the directory, objects are uploaded to the
root directory of the bucket.
upload_dir = 'exampledir/'
```

## Step 2: Configure the client

1. Download the [client source code](#) package.
2. Decompress the package to a directory. In this example, the `D:\aliyun\aliyun-oss-appserver-js` directory is used.
3. Go to the directory, open the `upload.js` file, and then find the following code:

```
// serverUrl specifies the URL of the application server that returns information such
as the signature and upload policies. Replace the value of serverUrl with the actual IP
address and port number of the application server.
serverUrl = 'http://192.0.2.0:8888'
```

4. Set `serverUrl` to the URL of the application server that returns information such as the signature and upload policies to the client. In this example, `serverUrl` is set to `https://192.168.0.0:8888`.

## Step 3: Configure CORS

When you use form upload to upload data from the client to OSS, the client sends a request that contains the `Origin` header to OSS by using the browser. Then, OSS checks whether the request that contains the `Origin` header matches the cross-origin resource sharing (CORS) rules that you configure for a specific bucket. Before you use the POST method to upload data to a bucket, configure CORS rules for the bucket.

- 1.
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**. The following figure shows the configurations of a rule.

 **Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

## Step 4: Send an upload callback request

1. Start the application server.

In the `/home/aliyun-oss-appserver-python` directory, run the `python appserver.py 11.22.33.44 1234` command to start the application server.

 **Note** Replace the IP address and port number with those of the application server that you configure.

2. Start the client.

On the client that is installed on your PC, open the `index.html` file in the directory of the client source code.

 **Notice** The `index.html` file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

3. Upload a file.

Click **Select File**, select a file of the specified type, and then click **Upload**. After you upload the file, the content that is returned by the application server is displayed.

## Analysis of the core code of the application server

The source code of the application server contains the following complete sample code for signature-based upload and upload callback:

```
# -*- coding: UTF-8 -*-
import socket
import base64
import sys
import time
import datetime
import json
import hmac
from hashlib import sha1 as sha
import httpserver

# The AccessKey pair of an Alibaba Cloud account has permissions on all API operations. Using these credentials to access OSS is a high-risk operation. We recommend that you use a RAM user to call API operations or perform routine O&M. To create a RAM user, log on to the RAM console.
access_key_id = 'yourAccessKeyId'
access_key_secret = 'yourAccessKeySecret'

# Specify the access address of the host in the https://bucketname.endpoint format.
host = 'https://examplebucket.oss-cn-hangzhou.aliyuncs.com';
# Specify the URL of the application server to which an upload callback request is sent. This URL is used for communication between the application server and OSS. After you upload an object, OSS uses the URL to send information about the uploaded object to the application server.
callback_url = "https://192.168.0.0:8888";
# Specify the directory in which you want to store uploaded objects. You can also leave this parameter empty. If you do not specify the directory, objects are uploaded to the root directory of the bucket.
upload_dir = 'exampledir/'
expire_time = 30
def get_iso_8601(expire):
    gmt = datetime.datetime.utcnow().timestamp().isoformat()
    gmt += 'Z'
    return gmt
def get_token():
    now = int(time.time())
    expire_syncpoint = now + expire_time
    expire_syncpoint = 1612345678
    expire = get_iso_8601(expire_syncpoint)
    policy_dict = {}
    policy_dict['expiration'] = expire
    condition_array = []
    array_item = []
    array_item.append('starts-with');
    array_item.append('$key');
    array_item.append(upload_dir);
    condition_array.append(array_item)
    policy_dict['conditions'] = condition_array
    policy = json.dumps(policy_dict).strip()
    policy_encode = base64.b64encode(policy.encode())
    h = hmac.new(access_key_secret.encode(), policy_encode, sha)
    sign_result = base64.encodestring(h.digest()).strip()
    callback_dict = {}
```

```

callback_dict['callbackUrl'] = callback_url;
callback_dict['callbackBody'] = 'filename=${object}&size=${size}&mimeType=${mimeType}'
\
                                '&height=${imageInfo.height}&width=${imageInfo.width}';
callback_dict['callbackBodyType'] = 'application/x-www-form-urlencoded';
callback_param = json.dumps(callback_dict).strip()
base64_callback_body = base64.b64encode(callback_param.encode());
token_dict = {}
token_dict['accessid'] = access_key_id
token_dict['host'] = host
token_dict['policy'] = policy_encode.decode()
token_dict['signature'] = sign_result.decode()
token_dict['expire'] = expire_syncpoint
token_dict['dir'] = upload_dir
token_dict['callback'] = base64_callback_body.decode()
result = json.dumps(token_dict)
return result
def get_local_ip():
    """
    Obtain the IPv4 address of the host.
    :return: If the operation is successful, the IPv4 address of the host is returned. Otherwise, no results are returned.
    """
    try:
        csocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        csocket.connect(('198.51.100.0', 80))
        (addr, port) = csocket.getsockname()
        csocket.close()
        return addr
    except socket.error:
        return ""
def do_POST(server):
    """
    Use the POST method to call processing logic.
    :param server: Web HTTP Server services
    :return:
    """
    print("***** do_POST ")
    # get public key
    pub_key_url = ''
    try:
        pub_key_url_base64 = server.headers['x-oss-pub-key-url']
        pub_key = httpserver.get_pub_key(pub_key_url_base64)
    except Exception as e:
        print(str(e))
        print('Get pub key failed! pub_key_url : ' + pub_key_url)
        server.send_response(400)
        server.end_headers()
        return
    # get authorization
    authorization_base64 = server.headers['authorization']
    # get callback body
    content_length = server.headers['content-length']
    callback_body = server.rfile.read(int(content_length))
    # compose authorization string

```

```
# Compose authorization string
auth_str = ''
pos = server.path.find('?')
if -1 == pos:
    auth_str = server.path + '\n' + callback_body.decode()
else:
    auth_str = httpserver.get_http_request_unquote(server.path[0:pos]) + server.path[pos:] + '\n' + callback_body
result = httpserver.verify(auth_str, authorization_base64, pub_key)
if not result:
    print('Authorization verify failed!')
    print('Public key : %s' % (pub_key))
    print('Auth string : %s' % (auth_str))
    server.send_response(400)
    server.end_headers()
    return
# response to OSS
resp_body = '{"Status":"OK"}'
server.send_response(200)
server.send_header('Content-Type', 'application/json')
server.send_header('Content-Length', str(len(resp_body)))
server.end_headers()
server.wfile.write(resp_body.encode())
def do_GET(server):
    """
    Use the GET method to call processing logic.
    :param server: Web HTTP Server services
    :return:
    """
    print("***** do_GET ")
    token = get_token()
    server.send_response(200)
    server.send_header('Access-Control-Allow-Methods', 'POST')
    server.send_header('Access-Control-Allow-Origin', '*')
    server.send_header('Content-Type', 'text/html; charset=UTF-8')
    server.end_headers()
    server.wfile.write(token.encode())
if '__main__' == __name__:
    # In the application server, 0.0.0.0 indicates all IPv4 addresses of the host.
    # If a host has two IP addresses, such as 192.0.2.0 and 192.0.2.254, and a service on the host listens to the IP address 0.0.0.0, the service can be accessed by using both of the two IP addresses.
    # If you want to listen to the IPv4 address of the host over the Internet, you can comment out the next line of code and use the following line of code instead of server_ip = get_local_ip().
    server_ip = "0.0.0.0"
    server_port = 8080
    if len(sys.argv) == 2:
        server_port = int(sys.argv[1])
    if len(sys.argv) == 3:
        server_ip = sys.argv[1]
        server_port = int(sys.argv[2])
    print("App server is running on http://%s:%s " % (server_ip, server_port))
    server = httpserver.MyHTTPServer(server_ip, server_port)
    server.serve_forever()
```

For more information about the API operation that you can call to configure upload callback, see [Callback](#).

### 2.1.5.3. Java

This topic describes how to calculate signatures by using Java code on the server, configure upload callback, and then use form upload to upload data to Object Storage Service (OSS).

#### Prerequisites

- The domain name of the application server can be accessed over the Internet.
- Java 1.6 or later is installed on the application server. To view the Java version, run the `java -version` command.
- JavaScript is supported by the browser on your PC.

#### Step 1: Configure the application server

1. Download the [application server source code](#) package in Java.
2. In this example, `Ubuntu 16.04` is used. Decompress the downloaded package to the `/home/aliyun/aliyun-oss-appserver-java` directory.
3. Go to the directory, open the `CallbackServer.java` file, and then modify the file based on your business requirements. The following code provides an example on how to modify the file:

```
// The AccessKey pair of an Alibaba Cloud account has permissions on all API operations
. Using these credentials to perform operations in OSS is a high-risk operation. We rec
ommend that you use a RAM user to call API operations or perform routine O&M. To create
a RAM user, log on to the RAM console.
String accessId = "yourAccessKeyId";
String accessKey = "yourAccessKeySecret";
// In this example, the endpoint of the China (Hangzhou) region is used. Specify your a
ctual endpoint.
String endpoint = "oss-cn-hangzhou.aliyuncs.com";
// Specify the bucket name. Example: examplebucket.
String bucket = "examplebucket";
// Specify the access address of the host in the https://bucketname.endpoint format.

String host = "https://examplebucket.oss-cn-hangzhou.aliyuncs.com";
// Specify the URL of the application server to which an upload callback request is sen
t. This URL is used for communication between the application server and OSS. After you
upload an object, OSS uses the URL to send information about the uploaded object to the
application server.
String callbackUrl = "https://192.168.0.0:8888";
// Specify the directory in which you want to store uploaded objects. You can also leav
e this parameter empty. If you do not specify the directory, objects are uploaded to th
e root directory of the bucket.
String dir = "exampledir/";
```

#### Step 2: Configure the client

1. Download the [client source code](#) package.

2. Decompress the package to a directory. In this example, the `D:\aliyun\aliyun-oss-appserver-js` directory is used.

3. Go to the directory, open the `upload.js` file, and then find the following code:

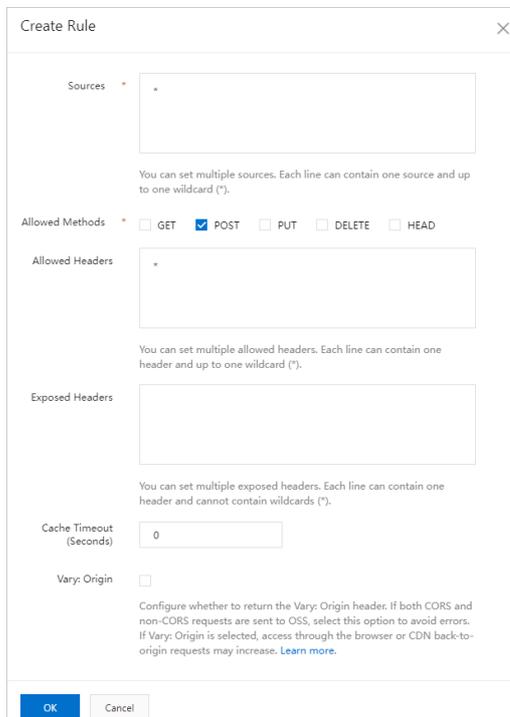
```
// serverUrl specifies the URL of the application server that returns information such as the signature and upload policies. Replace the value of serverUrl with the actual IP address and port number of the application server.
serverUrl = 'http://192.0.2.0:8888'
```

4. Set `serverUrl` to the URL of the application server that returns information such as the signature and upload policies to the client. In this example, `serverUrl` is set to `https://192.168.0.0:8888`.

### Step 3: Configure CORS

When you use form upload to upload data from the client to OSS, the client sends a request that contains the `Origin` header to OSS by using the browser. Then, OSS checks whether the request that contains the `Origin` header matches the cross-origin resource sharing (CORS) rules that you configure for a specific bucket. Before you use the POST method to upload data to a bucket, configure CORS rules for the bucket.

- 1.
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule** and configure the parameters as shown in the following figure.



**Note** To ensure data security, we recommend that you specify the domain name from which you want OSS to allow requests in **Sources**. For more information about how to configure the parameters, see [Configure CORS](#).

## Step 4: Send an upload callback request

1. Start the application server.

In the `/home/aliyun/aliyun-oss-appserver-java` directory, run the `mvn package` command to compile and package the code. Then, run the `java -jar target/appservermaven-1.0.0.jar 1234` command to start the application server.

 **Note** Replace the IP address and port number with those of the application server that you configure.

You also can use an integrated development environment (IDE), such as Eclipse or IntelliJ IDEA, to export the JAR package on the PC and copy the JAR package to the application server. Then, run the JAR package to start the application server.

2. Start the client.

- i. On the client that is installed on your PC, open the `index.html` file in the directory of the client source code.

 **Notice** The `index.html` file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

- ii. Click **Select File**, select a file of the specified type, and then click **Upload**.

After you upload the file, the content that is returned by the application server is displayed.

## Analysis of the core code of the application server

The source code of the application server contains the following complete sample code for signature-based upload and upload callback:

```
package com.aliyun.oss.appservermaven;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URI;
import java.security.KeyFactory;
import java.security.PublicKey;
import java.security.spec.X509EncodedKeySpec;
import java.sql.Date;
import java.util.LinkedHashMap;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import com.aliyun.oss.OSSClient;
import com.aliyun.oss.common.utils.BinaryUtil;
import com.aliyun.oss.model.MatchMode;
```

```
import com.aliyun.oss.model.PolicyConditions;
import com.aliyun.oss.model.PolicyConditions;
import org.junit.Assert;
import net.sf.json.JSONObject;
@SuppressWarnings("deprecation")
@WebServlet(asyncSupported = true)
public class CallbackServer extends HttpServlet {
    /**
     *
     */
    private static final long serialVersionUID = 5522372203700422672L;
    /**
     *Configure signature-based upload and upload callback in GET requests.
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // The AccessKey pair of an Alibaba Cloud account has permissions on all API operations. Using these credentials to perform operations in OSS is a high-risk operation. We recommend that you use a RAM user to call API operations or perform routine O&M. To create a RAM user, log on to the RAM console.
        String accessId = "yourAccessKeyId";
        String accessKey = "yourAccessKeySecret";
        // In this example, the endpoint of the China (Hangzhou) region is used. Specify your actual endpoint.
        String endpoint = "oss-cn-hangzhou.aliyuncs.com";
        // Specify the bucket name. Example: examplebucket.
        String bucket = "examplebucket";
        // Specify the access address of the host in the https://bucketname.endpoint format.
        String host = "https://examplebucket.oss-cn-hangzhou.aliyuncs.com";
        // Specify the URL of the application server to which an upload callback request is sent. This URL is used for communication between the application server and OSS. After you upload an object, OSS uses the URL to send information about the uploaded object to the application server.
        String callbackUrl = "https://192.168.0.0:8888";
        // Specify the directory in which you want to store uploaded objects. You can also leave this parameter empty. If you do not specify the directory, objects are uploaded to the root directory of the bucket.
        String dir = "exampledir/";
        OSSClient client = new OSSClient(endpoint, accessId, accessKey);
        try {
            long expireTime = 30;
            long expireEndTime = System.currentTimeMillis() + expireTime * 1000;
            Date expiration = new Date(expireEndTime);
            PolicyConditions policyConds = new PolicyConditions();
            policyConds.addConditionItem(PolicyConditions.COND_CONTENT_LENGTH_RANGE, 0, 1048576000);
            policyConds.addConditionItem(MatchMode.StartsWith, PolicyConditions.COND_KEY, dir);

            String postPolicy = client.generatePostPolicy(expiration, policyConds);
            byte[] binaryData = postPolicy.getBytes("utf-8");
            String encodedPolicy = BinaryUtil.toBase64String(binaryData);
            String postSignature = client.calculatePostSignature(postPolicy);
            Map<String, String> respMap = new LinkedHashMap<String, String>();
            respMap.put("accessid", accessId);
```

```

        respMap.put("policy", encodedPolicy);
        respMap.put("signature", postSignature);
        respMap.put("dir", dir);
        respMap.put("host", host);
        respMap.put("expire", String.valueOf(expireEndTime / 1000));
        // respMap.put("expire", formatISO8601Date(expiration));
        JSONObject jasonCallback = new JSONObject();
        jasonCallback.put("callbackUrl", callbackUrl);
        jasonCallback.put("callbackBody",
            "filename=${object}&size=${size}&mimeType=${mimeType}&height=${imageInfo.height}&width=${imageInfo.width}");
        jasonCallback.put("callbackBodyType", "application/x-www-form-urlencoded");
        String base64CallbackBody = BinaryUtil.toBase64String(jasonCallback.toString().getBytes());
        respMap.put("callback", base64CallbackBody);
        JSONObject jal = JSONObject.fromObject(respMap);
        // System.out.println(jal.toString());
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "GET, POST");
        response(request, response, jal.toString());
    } catch (Exception e) {
        // Assert.fail(e.getMessage());
        System.out.println(e.getMessage());
    }
}
/**
 * Obtain the public key.
 *
 * @param url
 * @return
 */
@SuppressWarnings({ "finally" })
public String executeGet(String url) {
    BufferedReader in = null;
    String content = null;
    try {
        // Specify HttpClient.
        @SuppressWarnings("resource")
        DefaultHttpClient client = new DefaultHttpClient();
        // Create an instance for the HTTP method.
        HttpGet request = new HttpGet();
        request.setURI(new URI(url));
        HttpResponse response = client.execute(request);
        in = new BufferedReader(new InputStreamReader(response.getEntity().getContent()));

        StringBuffer sb = new StringBuffer("");
        String line = "";
        String NL = System.getProperty("line.separator");
        while ((line = in.readLine()) != null) {
            sb.append(line + NL);
        }
        in.close();
        content = sb.toString();
    } catch (Exception e) {

```

```
        } finally {
            if (in != null) {
                try {
                    in.close();// Disable BufferedReader.
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            return content;
        }
    }
}
/**
 * Obtain the body of a POST request.
 *
 * @param is
 * @param contentLen
 * @return
 */
public String GetPostBody(InputStream is, int contentLen) {
    if (contentLen > 0) {
        int readLen = 0;
        int readLengthThisTime = 0;
        byte[] message = new byte[contentLen];
        try {
            while (readLen != contentLen) {
                readLengthThisTime = is.read(message, readLen, contentLen - readLen);
                if (readLengthThisTime == -1) { // Should not happen.
                    break;
                }
                readLen += readLengthThisTime;
            }
            return new String(message);
        } catch (IOException e) {
        }
    }
    return "";
}
/**
 * Verify the upload callback request.
 *
 * @param request
 * @param ossCallbackBody
 * @return
 * @throws NumberFormatException
 * @throws IOException
 */
protected boolean VerifyOSSCallbackRequest(HttpServletRequest request, String ossCallbackBody)
    throws NumberFormatException, IOException {
    boolean ret = false;
    String authorizationInput = new String(request.getHeader("Authorization"));
    String pubKeyInput = request.getHeader("x-oss-pub-key-url");
    byte[] authorization = BinaryUtil.fromBase64String(authorizationInput);
    byte[] pubKey = BinaryUtil.fromBase64String(pubKeyInput);
}
```

```

String pubKeyAddr = new String(pubKey);
if (!pubKeyAddr.startsWith("http://gosspublic.alicdn.com/")
    && !pubKeyAddr.startsWith("https://gosspublic.alicdn.com/")) {
    System.out.println("pub key addr must be oss addrss");
    return false;
}
String retString = executeGet(pubKeyAddr);
retString = retString.replace("-----BEGIN PUBLIC KEY-----", "");
retString = retString.replace("-----END PUBLIC KEY-----", "");
String queryString = request.getQueryString();
String uri = request.getRequestURI();
String decodeUri = java.net.URLDecoder.decode(uri, "UTF-8");
String authStr = decodeUri;
if (queryString != null && !queryString.equals("")) {
    authStr += "?" + queryString;
}
authStr += "\n" + ossCallbackBody;
ret = doCheck(authStr, authorization, retString);
return ret;
}
/**
 * Configure signature-based upload and upload callback in POST requests.
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String ossCallbackBody = GetPostBody(request.getInputStream(),
        Integer.parseInt(request.getHeader("content-length")));
    boolean ret = VerifyOSSCallbackRequest(request, ossCallbackBody);
    System.out.println("verify result : " + ret);
    // System.out.println("OSS Callback Body:" + ossCallbackBody);
    if (ret) {
        response(request, response, "{\"Status\":\"OK\"}", HttpServletResponse.SC_OK);
    } else {
        response(request, response, "{\"Status\":\"verdifly not ok\"}", HttpServletResponse.SC_BAD_REQUEST);
    }
}
/**
 * Verify the RSA-based signature.
 *
 * @param content
 * @param sign
 * @param publicKey
 * @return
 */
public static boolean doCheck(String content, byte[] sign, String publicKey) {
    try {
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        byte[] encodedKey = BinaryUtil.fromBase64String(publicKey);
        PublicKey pubKey = keyFactory.generatePublic(new X509EncodedKeySpec(encodedKey));

        java.security.Signature signature = java.security.Signature.getInstance("MD5withRSA");
        signature.initVerify(pubKey);
        signature.update(content.getBytes());
    }
}

```

```
        signature.update(content.getBytes());
        boolean bverify = signature.verify(sign);
        return bverify;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
/**
 * The response returned by the application server.
 *
 * @param request
 * @param response
 * @param results
 * @param status
 * @throws IOException
 */
private void response(HttpServletRequest request, HttpServletResponse response, String
results, int status)
    throws IOException {
    String callbackFunName = request.getParameter("callback");
    response.addHeader("Content-Length", String.valueOf(results.length()));
    if (callbackFunName == null || callbackFunName.equalsIgnoreCase(""))
        response.getWriter().println(results);
    else
        response.getWriter().println(callbackFunName + "( " + results + " )");
    response.setStatus(status);
    response.flushBuffer();
}
/**
 * The response returned by the application server.
 */
private void response(HttpServletRequest request, HttpServletResponse response, String
results) throws IOException {
    String callbackFunName = request.getParameter("callback");
    if (callbackFunName == null || callbackFunName.equalsIgnoreCase(""))
        response.getWriter().println(results);
    else
        response.getWriter().println(callbackFunName + "( " + results + " )");
    response.setStatus(HttpServletResponse.SC_OK);
    response.flushBuffer();
}
}
```

For more information about the API operation that you can call to configure upload callback, see [Callback](#).

## 2.1.5.4. Go

This topic describes how to calculate signatures in Go on the server, configure upload callback, and use form upload to upload data to OSS.

### Prerequisites

- The domain name of the application server is accessible over the Internet.
- The application server has Go 1.6 or later installed. To verify the Go version, run the `go version` command.
- The browser on the PC supports JavaScript.

## Step 1: Configure the application server

1. Download the [application server source code](#) that is in Go.
2. Ubuntu 16.04 is used in the example. Download the source code to the `/home/aliyun/aliyun-oss-appserver-go` directory.
3. Go to the directory. Open the `appserver.go` file that contains the source code. Modify the following snippet:

```
// Enter your AccessKey ID.
var accessKeyId string = "<yourAccessKeyId>"
// Enter your AccessKey secret.
var accessKeySecret string = "<yourAccessKeySecret>"
// Set host to a value that is in the format of bucketname.endpoint.
var host string = "https://bucket-name.oss-cn-hangzhou.aliyuncs.com"
// Set the URL of the server to which an upload callback request is sent. Replace the IP address and port number with your actual information.
var callbackUrl string = "http://88.88.88.88:8888";
// Specify the prefix for the name of the object you want to upload.
var upload_dir string = "user-dir-prefix/"
// Set the validity period in seconds for the upload policy.
var expire_time int64 = 30
```

- `accessKeyId`: Enter your AccessKey ID.
- `accessKeySecret`: Enter your AccessKey secret.
- `host`: The format is `https://bucketname.endpoint`. Example: `https://bucket-name.oss-cn-hangzhou.aliyuncs.com`. For more information about endpoints, see the "Endpoint" section in [Terms](#).
- `callbackUrl`: Set the URL of the server to which an upload callback request is sent. This URL is used to communicate between the application server and OSS. After an object is uploaded, OSS sends the object upload information to the application server by using this URL. In this example, set the URL to `var callbackUrl string="http://11.22.33.44:1234";`.
- `dir`: Specify the prefix for the name of the object. You can also leave this parameter unspecified.

## Step 2: Configure the client

1. Download the [client source code](#) to the local directory on the PC.
2. Decompress the package. Open the `upload.js` file. Find the following code:

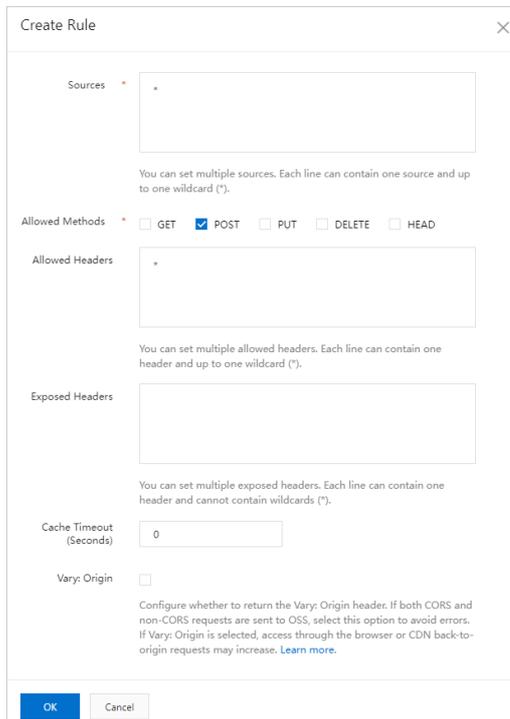
```
// serverUrl specifies the URL of the application server used to obtain information such as the signature and policy. Replace the IP address and port number with your actual information.
serverUrl ='http://88.88.88.88:8888'
```

3. Set `serverUrl` to the URL of the application server. In this example, specify `serverUrl ='http://11.22.33.44:1234'`.

### Step 3: Modify CORS configurations

When you use form upload to upload data from the client to OSS, the client includes the `Origin` header in the request and sends the request to OSS by using the browser. OSS verifies the request message that includes the `Origin` header for cross-origin resource sharing (CORS) verification. To use the POST method, configure CORS rules for a bucket.

1. Log on to the [OSS console](#).
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**. The following figure shows the configurations of a rule.



**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

### Step 4: Send an upload callback request

1. Start the application server.  
In the `/home/aliyun/aliyun-oss-appserver-go` directory, run the `go run appserver.go 11.22.33.44 1234` command.

**Note** Replace the IP address and port number with those of the application server you configured.

2. Start the client.

- i. On the PC, open the *index.html* file in the directory that contains the client source code.

 **Notice** The *index.html* file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

- ii. Click **Select File**. Select the file of a specified type. Click **Upload**. After the object is uploaded, the content returned by the callback server is displayed.

## Core code analysis of the application server

The source code of the application server is used to implement signature-based upload and upload callback.

- Signature-based upload

During signature-based upload, the application server responds to the GET message that is sent from the client. An example of the snippet:

```
func handlerRequest(w http.ResponseWriter, r *http.Request) {
    if (r.Method == "GET") {
        response := get_policy_token()
        w.Header().Set("Access-Control-Allow-Methods", "POST")
        w.Header().Set("Access-Control-Allow-Origin", "*")
        io.WriteString(w, response)
    }
}
```

- Upload callback

During upload callback, the application server responds to the POST message that is sent from OSS. An example of the snippet:

```
if (r.Method == "POST") {
    fmt.Println("\nHandle Post Request ... ")
    // Get PublicKey bytes
    bytePublicKey, err := getPublicKey(r)
    if (err != nil) {
        responseFailed(w)
        return
    }
    // Get Authorization bytes : decode from Base64String
    byteAuthorization, err := getAuthorization(r)
    if (err != nil) {
        responseFailed(w)
        return
    }
    // Get MD5 bytes from Newly Constructed Authorization String.
    byteMD5, err := getMD5FromNewAuthString(r)
    if (err != nil) {
        responseFailed(w)
        return
    }
    // verifySignature and response to client
    if (verifySignature(bytePublicKey, byteMD5, byteAuthorization)) {
        // do something you want according to callback_body ...
        responseSuccess(w) // response OK : 200
    } else {
        responseFailed(w) // response FAILED : 400
    }
}
```

For more information, see the "(Optional) Step 4: Sign the callback request" section in [Callback](#) of the OSS API Reference.

## 2.1.5.5. Node.js

This topic describes how to add signatures to requests by using Node.js on application servers, configure upload callback, and then use form upload to upload data to Object Storage Service (OSS).

### Prerequisites

- The domain name of the application server can be accessed over the Internet.
- Node.js 8.0 or later is installed on the application server. You can run the `node -v` command to obtain the Node.js version used on the application server.
- The browser used by the client must support HTML4, HTML5, Flash, and Silverlight.

### Step 1: Configure the app server

1. Download the [application server source code](#).
2. Ubuntu 16.04 is used in the example. Decompress the source code to the `/home/aliyun/aliyun-oss-appserver-node.js` directory.
3. Run the `npm install` command in the root directory of your project.
4. Find the `app.js` source code file in the root directory of your project and modify the following snippet:

```
const config = {
  // Security risks may arise if you use the AccessKey pair of an Alibaba Cloud account
  // to access OSS because the account has permissions on all API operations. We recommend t
  // hat you use a Resource Access Management (RAM) user to call API operations or perform r
  // outine O&M. To create a RAM user, log on to the RAM console.
  accessKeyId: 'yourAccessKeyId',
  accessKeySecret: 'yourAccessKeySecret',
  // Specify the bucket name.
  bucket: 'yourBucket',
  // Set the URL of the application server to which an upload callback request is sent.
  // This URL is used to communicate between the application server and OSS. After an object
  // is uploaded, OSS uses the URL to send information about the object to the application s
  // erver. For example, you can set callbackUrl to https://oss-demo.aliyuncs.com:23450.
  callbackUrl: 'yourCallBackUrl',
  // Configure the parameter if you want to specify a prefix for the objects to upload.
  // Otherwise, you can leave the parameter empty.
  dir: 'yourPrefix'
}
```

## Step 2: Configure the client

1. Download the [client source code](#).
2. Decompress the downloaded package. Decompress the package to a directory. The `D:\aliyun\aliyun-oss-appserver-js` directory is used in this example.
3. Go to the directory. Open the `upload.js` file. Find the following code:

```
// serverUrl specifies the URL of the application server that users use to obtain infor
// mation such as the signature and upload policy. Replace the IP address and port number
// with your actual information.
serverUrl = 'http://88.88.88.88:8888'
```

4. Set `serverUrl` to the URL of the application server. For example, you can set `serverUrl` to `serverUrl = 'https://oss-demo.aliyuncs.com:23450'` so that the client can use the URL to obtain required information such as the signature and upload policy.

## Step 3: Configure CORS

When you use form upload to upload data from the client to OSS, the client sends a request that contains the `Origin` header to OSS by using the browser. Then, OSS verifies the request that contains the `Origin` header against the cross-origin resource sharing (CORS) rules that you configure for a specific bucket. Therefore, you must configure CORS rules for the bucket before users can use the POST method to upload data to the bucket.

1. Log on to the [OSS console](#).
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule** and configure the parameters showed in the following figure.

**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about how to configure the parameters, see [Configure CORS rules](#).

### Step 4: Send an upload callback request

1. In the root directory of the application server, run the `npm run server` command to start the application server.
2. On the PC client, open the `index.html` file in the directory of the client source code.

**Notice** Internet Explorer below IE 10 may be incompatible with the `index.html` file. You must troubleshoot the errors yourself if you use browsers below IE 10.

3. Click **Select File**. Select the file of a specified type. Click **Upload**.

After the object is uploaded, the content returned by the application server is displayed.

### Analysis of the application server source code

The application server source code contains a complete sample code for adding signatures, uploading objects to OSS, and configuring upload callback. The following code provides only the core code snippet. For more information about how to implement features, download [application server source code](#) and view the details.

- Add signatures to requests and upload objects to OSS

The following snippet provides an example on how to obtain parameter values used for signature and the URL of the application server:

```

app.get("/", async (req, res) => {
  const client = new OSS(config);
  const date = new Date();
  date.setDate(date.getDate() + 1);
  const policy = {
    expiration: date.toISOString(), // Set the UNIX timestamp that is in seconds since the
    // UTC time January 1, 1970 to identify the timeout period of the upload request.
    conditions: [
      ["content-length-range", 0, 1048576000] // Set the size limit for the object to upload.
    ],
  };
  // Configure the CORS rules.
  res.set({
    "Access-Control-Allow-Origin": req.headers.origin || "*",
    "Access-Control-Allow-Methods": "PUT,POST,GET",
  });
  // Use the SDK to obtain the signature.
  const formData = await client.calculatePostSignature(policy);
  // Specify the endpoint of the bucket over the Internet.
  const host = `http://${config.bucket}.${
    (await client.getBucketLocation()).location
  }.aliyuncs.com`.toString();
  // Configure upload callback.
  const callback = {
    callbackUrl: config.callbackUrl, // Set the URL of the application server used to send
    // upload callback requests. Example: http://oss-demo.aliyuncs.com:23450.
    callbackBody: // Specify the content that you want the callback request to contain such
    // as the ETag and the mimeType of the uploaded object.
    `filename=${object}&size=${size}&mimeType=${mimeType}&height=${imageInfo.height}&width=${imageInfo.width}`,
    callbackBodyType: "application/x-www-form-urlencoded", // Set the content type of the
    // callback information.
  };
  // Specify the required parameters contained in the response.
  const params = {
    expire: moment().add(1, "days").unix().toString(),
    policy: formData.policy, // The policy obtained from OSS.
    signature: formData.Signature, // The signature obtained from OSS.
    accessid: formData.OSSAccessKeyId, // The AccessKey ID obtained from OSS.
    host, // The format is https://bucketname.endpoint. Example: https://bucket-name.oss-cn-hangzhou.aliyuncs.com.
    callback: Buffer.from(JSON.stringify(callback)).toString("base64"), // The Base64-encoded
    // JSON by using Buffer.from.
    dir: config.dir, // The obtained prefix for the uploaded object.
  };
  res.json(params);
});

```

- Upload callback

The following snippet provides an example on how to configure the POST response returned by the application server to the callback request sent by OSS:

```
// Listen the POST requests in the /result path.
app.post("/result", (req, res) => {
  // Decode the address of the OSS public key by using Base64.
  const pubKeyAddr = Buffer.from(
    req.headers["x-oss-pub-key-url"],
    "base64"
  ).toString("ascii");
  // Determine whether x-oss-pub-key-url in the request header comes from OSS.
  if (
    !pubKeyAddr.startsWith("https://gosspublic.aliqdn.com/") &&
    !pubKeyAddr.startsWith("https://gosspublic.aliqdn.com/")
  ) {
    System.out.println("pub key addr must be oss address");
    // If not, return "verify not ok", which indicates that the upload callback fails.
    res.json({ Status: "verify not ok" });
  }
  // If x-oss-pub-key-url comes from OSS, this indicates that the upload callback succeeds.
  res.json({ Status: "Ok" });
});
```

For more information about upload callback, see [Callback](#).

## 2.1.5.6. .NET

This topic describes how to calculate signatures in .NET on the server, configure upload callback, and use form upload to upload data to OSS.

### Prerequisites

- The domain name of the application server is accessible over the Internet.
- The application server has Visual Studio 2012 or later installed.
- The application server has .Net Framework 4.5 or later installed.

### Step 1: Configure the application server

1. Download the [application server source code](#) that is in .NET.
2. The Windows system is used in the example. Decompress the source code to the `C:\callback-server-dotnet` directory.
3. Go to the directory. Open the `TinyHttpServer.cs` file that contains the source code. Modify the following snippet:

```
// Enter your AccessKey ID.
public static string accessKeyId = "<yourAccessKeyId>";
// Enter your AccessKey secret.
public static string accessKeySecret = "<yourAccessKeySecret>";
// Set host to a value that is in the format of https://bucketname.endpoint.
public static string host = "https://bucketname.oss-cn-hangzhou.aliyuncs.com";
// Set the URL of the server to which an upload callback request is sent. Replace the IP
// address and port number with your actual information.
public static string callbackUrl = "http://192.168.0.1:8888";
// Specify the prefix for the name of the object you want to upload.
public static string uploadDir = "user-dir-prefix/";
```

- o `accessKeyId`: Enter your AccessKey ID.
  - o `accessKeySecret`: Enter your AccessKey secret.
  - o `host`: The format is `https://bucketname.endpoint`. Example: `https://bucket-name.oss-cn-hangzhou.aliyuncs.com`. For more information about endpoints, see the "Endpoint" section in [Terms](#).
  - o `callbackUrl`: Set the URL of the server to which an upload callback request is sent. This URL is used to communicate between the application server and OSS. After an object is uploaded, OSS sends the object upload information to the application server by using this URL. In this example, set the URL to `String callbackUrl = "http://10.10.10.10:1234";` .
  - o `uploadDir`: Specify the prefix of the object to prevent overwriting an existing object with the same name. You can also leave this parameter unspecified.
4. Open the `aliyun-oss-net-callback-server.sln` file by using Visual Studio. Click **Enable** to generate executable file `aliyun-oss-net-callback-server.exe`.

## Step 2: Configure the client

1. Download the [client source code](#).
2. Decompress the package to a folder. The `D:\aliyun\aliyun-oss-appserver-js` directory is used in the example.
3. Go to the directory. Open the `upload.js` file. Find the following code:

```
// serverUrl specifies the URL of the application server used to obtain information such as the signature and policy. Replace the IP address and port number with your actual information.
serverUrl = 'http://192.168.0.1:8888'
```

4. Set `serverUrl` to the URL of the application server. In this example, specify `serverUrl = 'http://10.10.10.10:1234'` .

## Step 3: Modify CORS configurations

When you use form upload to upload data from the client to OSS, the client includes the `Origin` header in the request and sends the request to OSS by using the browser. OSS verifies the request message that includes the `Origin` header for cross-origin resource sharing (CORS) verification. To use the POST method, configure CORS rules for a bucket.

1. Log on to the [OSS console](#).
- 2.

3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**. The following figure shows the configurations of a rule.

**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

### Step 4: Send an upload callback request

1. Start the application server.

In the command line of the application server, go to the directory where the *aliyun-oss-net-callback-server.exe* executable file is generated. Run the command **aliyun-oss-net-callback-server.exe `{ip}` `{port}`** to start the application server.

```
cd C:\Users\Desktop\callback-server-dotnet\aliyun-oss-net-callback-server\bin\Debug\
aliyun-oss-net-callback-server.exe 10.10.10.10 1234
```

**Note**

- The actual directory applies. When you use Visual Studio to generate the *aliyun-oss-net-callback-server.exe* executable file, you can view the directory.
- Change `{ip}` and `{port}` to the IP address and port number of the application server. Example: **aliyun-oss-net-callback-server.exe 10.10.10.10 1234**.

2. Start the client.

- i. On the PC, open the *index.html* file in the directory that contains the client source code.

 **Notice** The *index.html* file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

- ii. Click **Select File**. Select the file of a specified type. Click **Upload**.

After the object is uploaded, the content returned by the callback server is displayed.

## Core code analysis of the application server

The source code of the application server is used to implement signature-based upload and upload callback.

- During signature-based upload, the application server responds to the GET message that is sent from the client. An example of the snippet:

```
private static string GetPolicyToken()
{
    //expireTime
    var expireDateTime = DateTime.Now.AddSeconds(expireTime);
    // example of policy
    //{
    //  "expiration": "2020-05-01T12:00:00.000Z",
    //  "conditions": [
    //    ["content-length-range", 0, 1048576000]
    //    ["starts-with", "$key", "user-dir-prefix/"]
    //  ]
    //}
    //policy
    var config = new PolicyConfig();
    config.expiration = FormatIso8601Date(expireDateTime);
    config.conditions = new List<List<Object>>();
    config.conditions.Add(new List<Object>());
    config.conditions[0].Add("content-length-range");
    config.conditions[0].Add(0);
    config.conditions[0].Add(1048576000);
    config.conditions.Add(new List<Object>());
    config.conditions[1].Add("starts-with");
    config.conditions[1].Add("$key");
    config.conditions[1].Add(uploadDir);
    var policy = JsonConvert.SerializeObject(config);
    var policy_base64 = EncodeBase64("utf-8", policy);
    var signature = ComputeSignature(accessKeySecret, policy_base64);
    //callback
    var callback = new CallbackParam();
    callback.callbackUrl = callbackUrl;
    callback.callbackBody = "filename=${object}&size=${size}&mimeType=${mimeType}&height=${
imageInfo.height}&width=${imageInfo.width}";
    callback.callbackBodyType = "application/x-www-form-urlencoded";
    var callback_string = JsonConvert.SerializeObject(callback);
    var callback_string_base64 = EncodeBase64("utf-8", callback_string);
    var policyToken = new PolicyToken();
    policyToken.accessid = accessKeyId;
```

```

policyToken.host = host;
policyToken.policy = policy_base64;
policyToken.signature = signature;
policyToken.expire = ToUnixTime(expireDateTime);
policyToken.callback = callback_string_base64;
policyToken.dir = uploadDir;
return JsonConvert.SerializeObject(policyToken);
}
public void DoGet()
{
    Console.WriteLine("DoGet request: {0}", this.httpURL);
    var content = GetPolicyToken();
    this.swResponse.WriteLine("HTTP/1.0 200 OK");
    this.swResponse.WriteLine("Content-Type: application/json");
    this.swResponse.WriteLine("Access-Control-Allow-Origin: *");
    this.swResponse.WriteLine("Access-Control-Allow-Method: GET, POST");
    this.swResponse.WriteLine($"Content-Length: {content.Length.ToString()}");
    this.swResponse.WriteLine("Connection: close");
    this.swResponse.WriteLine("");
    this.swResponse.WriteLine(content);
}

```

- During upload callback, the application server responds to the POST message that is sent from OSS. An example of the snippet:

```

public bool VerifySignature()
{
    // Get the Authorization Base64 from Request
    if (this.httpHeadersDict["authorization"] != null)
    {
        this.strAuthorizationRequestBase64 = this.httpHeadersDict["authorization"].ToString();
    }
    else if (this.httpHeadersDict["Authorization"] != null) {
        this.strAuthorizationRequestBase64 = this.httpHeadersDict["Authorization"].ToString();
    }
    if (this.strAuthorizationRequestBase64 == "")
    {
        Console.WriteLine("authorization property in the http request header is null. ");
        return false;
    }
    // Decode the Authorization from Request
    this.byteAuthorizationRequest = Convert.FromBase64String(this.strAuthorizationRequestBase64);
    // Decode the URL of PublicKey
    this.strPublicKeyURLBase64 = this.httpHeadersDict["x-oss-pub-key-url"].ToString();
    var bytePublicKeyURL = Convert.FromBase64String(this.strPublicKeyURLBase64);
    var strAsciiPublicKeyURL = System.Text.Encoding.ASCII.GetString(bytePublicKeyURL);
    // Get PublicKey from the URL
    ServicePointManager.ServerCertificateValidationCallback = new RemoteCertificateValidationCallback(validateServerCertificate);
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(strAsciiPublicKeyURL);
    HttpWebResponse response = (HttpWebResponse)request.GetResponse();
    StreamReader srPublicKey = new StreamReader(response.GetResponseStream(), Encoding.UTF8);
}

```

```
..
    this.strPublicKeyBase64 = srPublicKey.ReadToEnd();
    response.Close();
    srPublicKey.Close();
    this.strPublicKeyContentBase64 = this.strPublicKeyBase64.Replace("-----BEGIN PUBLIC KEY
-----\n", "").Replace("-----END PUBLIC KEY-----", "").Replace("\n", "");
    this.strPublicKeyContentXML = this.RSAPublicKeyString2XML(this.strPublicKeyContentBase6
4);
    // Generate the New Authorization String according to the HttpRequest
    String[] arrURL;
    if (this.httpURL.Contains('?'))
    {
        arrURL = this.httpURL.Split('?') ;
        this.strAuthSourceForMD5 = String.Format("{0}?{1}\n{2}", System.Web.HttpUtility.UrlDe
code(arrURL[0]), arrURL[1], this.httpBody);
    }
    else
    {
        this.strAuthSourceForMD5 = String.Format("{0}\n{1}", System.Web.HttpUtility.UrlDecode
(this.httpURL), this.httpBody);
    }
    // MD5 hash bytes from the New Authorization String
    var byteAuthMD5 = byteMD5Encrypt32(this.strAuthSourceForMD5);
    // Verify Signature
    System.Security.Cryptography.RSACryptoServiceProvider RSA = new System.Security.Cryptog
raphy.RSACryptoServiceProvider();
    try
    {
        RSA.FromXmlString(this.strPublicKeyContentXML);
    }
    catch (System.ArgumentNullException e)
    {
        throw new ArgumentNullException(String.Format("VerifySignature Failed : RSADeformatte
r.VerifySignature get null argument : {0} .", e));
    }
    catch (System.Security.Cryptography.CryptographicException e)
    {
        throw new System.Security.Cryptography.CryptographicException(String.Format("VerifySi
gnature Failed : RSA.FromXmlString Exception : {0} .", e));
    }
    System.Security.Cryptography.RSAPKCS1SignatureDeformatter RSADeformatter = new System.S
ecurity.Cryptography.RSAPKCS1SignatureDeformatter(RSA);
    RSADeformatter.SetHashAlgorithm("MD5");
    var bVerifyResult = false;
    try
    {
        bVerifyResult = RSADeformatter.VerifySignature(byteAuthMD5, this.byteAuthorizationReq
uest);
    }
    catch (System.ArgumentNullException e)
    {
        throw new ArgumentNullException(String.Format("VerifySignature Failed : RSADeformatte
r.VerifySignature get null argument : {0} .", e));
    }
    catch (System.Security.Cryptography.CryptographicUnexpectedOperationException e)
```

```
{
    throw new System.Security.Cryptography.CryptographicUnexpectedOperationException(
        String.Format("VerifySignature Failed : RSADeformatter.VerifySignature Exception : {0} .", e)
    );
}
return bVerifyResult;
}
public void DoPost()
{
    this.GetPostBody();
    // Verify Signature
    try
    {
        if (this.VerifySignature())
        {
            Console.WriteLine("\nVerifySignature Successful . \n");
            // do something according to callback_body ...
            this.HttpResponseSuccess();
        }
        else
        {
            Console.WriteLine("\nVerifySignature Failed . \n");
            this.HttpResponseFailure();
        }
    }
    catch
    {
        Console.WriteLine("\nVerifySignature Failed . \n");
        this.HttpResponseFailure();
    }
}
```

## 2.1.5.7. PHP

This topic describes how to sign a URL in PHP on the server, configure upload callback, and use form upload to upload data to Object Storage Service (OSS).

### Prerequisites

- A web server is deployed.
- The domain name of the web server is accessible over the Internet.
- The web server can parse PHP. To view the PHP version, run the `php -v` command.
- The browser on the PC supports JavaScript.

### Step 1: Configure the web server

This topic uses Ubuntu 16.04 in the example to describe different environment configurations for different web servers. Configure the environment based on your requirements.

- If you use Apache as the web server, configure the environment based on the following description. Apache 2.4.18 is used in this example.

- Set the IP address of the web server over the Internet to `192.0.2.11` . You can modify the public IP address by adding `ServerName 192.0.2.11` to the `/etc/apache2/apache2.conf` configuration file.
- Set the listening port of the web server to `8080` . You can set the listening port to `Listen 8080` in the `/etc/apache2/ports.conf` configuration file.
- Ensure that Apache can parse PHP files. To install PHP 5, run the `sudo apt-get install libapache2-mod-php5` command. If other web servers are used, install PHP 5 and modify configurations based on your actual environment.

You can modify the IP address and listening port of your web server based on your actual environment. After you update the configurations, you must run `/etc/init.d/apache2 restart` to restart the Apache server.

- If you use NGINX as the web server, configure the environment based on the following description. NGINX 1.19.7 is used in this example.

Set the IP address of the web server over the Internet to `192.0.2.11` and set the listening port to `8080` . You can modify the IP address of the web server over the Internet and the listening port in the `/etc/nginx/nginx.conf` configuration file. The following code provides an example on how to modify the configuration file:

```
server {
    listen 8080;
    server_name 192.0.2.11;
    root /var/www/html;
    index index.html index.php;
    location ~* \.php$ {
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

You can modify the IP address and listening port of your web server based on your actual environment. After you update the configurations, you must restart the NGINX web server.

## Step 2: Configure the application server

1. Download the [application server source code](#) in PHP.
2. Decompress the application server source code to the corresponding directory of the application server. In this example, decompress the application server source code to the `/var/www/html/aliyun-oss-appserver-php` directory of Ubuntu 16.04.
3. Access the `http://192.0.2.11:8080/aliyun-oss-appserver-php/index.html` URL of the application server from a browser on the PC. If the displayed page is the same as the [homepage of test sample](#), the verification is passed.
4. If you use Apache as the web server, enable the feature to capture the Authorization field in the HTTP headers. If you use NGINX as the web server, you can skip this step.

The callback response received by your application server may not have an Authorization header. This is because some web application servers automatically parse the Authorization header. In this case, you can modify the configuration file so that the Authorization header is not parsed. Apache

2 is used in the following example:

- i. Open the `/etc/apache2/apache2.conf` configuration file of Apache 2. Find and modify the following snippet:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

- ii. In the `/var/www/html/aliyun-oss-appserver-php` directory, create a file named `.htaccess` and enter the following content:

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{HTTP:Authorization} .
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

If other web servers or other versions of Apache are used, install and configure the application server based on your actual environment.

5. Modify the configurations of the application server.

In the `/var/www/html/aliyun-oss-appserver-php/php` directory, open the `get.php` file. Modify the following snippet:

```
$id= '<yourAccessKeyId>'; // Enter your AccessKey ID.
$key= '<yourAccessKeySecret>'; // Enter your AccessKey secret.
// Set $host to a value that is in the format of https://bucketname.endpointx. Modify the URL based on your actual information.
$host = 'https://bucket-name.oss-cn-hangzhou.aliyuncs.com';
// Set $callbackUrl to the URL of the callback server. Replace the IP address and port number with your actual information.
$callbackUrl = 'http://192.0.2.11:8080/aliyun-oss-appserver-php/php/callback.php';
$dir = 'user-dir-prefix/'; // Specify the prefix for the name of the object you want to upload.
```

Parameter	Required	Example	Description
id	Yes	LTAn***** *****	The AccessKey ID and AccessKey secret of an Alibaba Cloud account or of a Resource Access Management (RAM) user. For more information, see <a href="#">Obtain an AccessKey pair</a> .
key	Yes	zbnK***** *****	
host	Yes	https://bucket-name.oss-cn-hangzhou.aliyuncs.com	The access address of the application server, and the format is <code>https://BucketName.Endpoint</code> . For more information about endpoints, see <a href="#">Regions and endpoints</a> .

Parameter	Required	Example	Description
callbackUrl	Yes	http://192.0.2.11:8080/aliyun-oss-appserver-php/php/callback.php	The URL of the callback server which is used to communicate between the application server and OSS. After an object is uploaded, OSS sends the object upload information to the application server by using this URL.
dir	No	exampledir/	The prefix of the object uploaded to OSS. Configure this parameter based on your requirements.  If you do not need to configure the prefix of the objects uploaded to OSS, leave the parameter empty.

### Step 3: Configure the client

In the `/var/www/html/aliyun-oss-appserver-php` directory of the application server, modify the `upload.js` file.

For the application server source code in PHP, you do not need to modify the `upload.js` file because relative paths can also properly function. If modification is required, find the `serverUrl` `= './php/get.php'` snippet. Then, change `serverUrl` to the address where the application server is deployed to communicate between the browser and the application server. In this example, change `serverUrl` to `serverUrl = 'http://192.0.2.11:8080/aliyun-oss-appserver-php/php/get.php'`.

### Step 4: Modify CORS configurations

When you use form upload to upload data from the client to OSS, the client sends a request that contains the `Origin` header to OSS from the browser. Then, OSS verifies the request that contains the `Origin` header against the cross-origin resource sharing (CORS) rules that you configure for a specific bucket. Therefore, you must configure CORS rules for the bucket before you can use the POST method to upload data to the bucket.

1. Log on to the [OSS console](#).
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**. The following figure shows the configurations of a rule.

**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

### Step 5: Send an upload callback request

1. Open the web browser on the PC, and enter `http://192.0.2.11:8080/aliyun-oss-appserver-php/index.html`.

**Notice** The `index.html` file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

2. Click **Select File**. Select the file of a specified type. Then, click **Upload**.  
After the object is uploaded, the content returned by the callback server is displayed.

### Core code analysis for the application server

The source code of the application server is used to implement signature-based upload and upload callback.

- Signature-based upload

During signature-based upload, the application server responds to the GET message sent from the client. The code file is `aliyun-oss-appserver-php/php/get.php`. The following code provides an example on the snippet:

```
$response = array();
$response['accessid'] = $id;
$response['host'] = $host;
$response['policy'] = $base64_policy;
$response['signature'] = $signature;
$response['expire'] = $end;
$response['callback'] = $base64_callback_body;
$response['dir'] = $dir;
```

- **Upload callback**

During upload callback, the application server responds to the POST message that is sent from OSS. The code file is *aliyun-oss-appserver-php/php/callback.php*.

The following code provides an example on the snippet:

```
// 6.Verify the signature.
$ok = openssl_verify($authStr, $authorization, $pubKey, OPENSSL_ALGO_MD5);
if ($ok == 1)
{
    header("Content-Type: application/json");
    $data = array("Status"=>"Ok");
    echo json_encode($data);
}
```

For more information, see [Callback](#) in OSS API Reference.

## 2.1.5.8. Ruby

This topic describes how to calculate signatures in Ruby on the server, configure upload callback, and use form upload to upload data to OSS.

### Prerequisites

- The domain name of the application server is accessible over the Internet.
- The application server has Ruby 2.0 or later installed. To view the Ruby version, run the `ruby -v` command.
- The browser on the PC supports JavaScript.

### Step 1: Configure the application server

1. Download the [application server source code](#) that is in Ruby.
2. Ubuntu 16.04 is used in the example. Decompress the source code to the */home/aliyun/aliyun-oss-appserver-ruby* directory.
3. Go to the directory. Open the *appserver.rb* file that contains the source code. Modify the following snippet:

```
# Enter your AccessKey ID.
$access_key_id = '<yourAccessKeyId>'
# Enter your AccessKey secret.
$access_key_secret = '<yourAccessKeySecret>'
# Set $host to a value that is in the format of bucketname.endpoint.
$host = 'https://bucket-name.oss-cn-hangzhou.aliyuncs.com';
# Set the URL of the server to which an upload callback request is sent. Replace the IP
address and port number with your actual information.
$callback_url = "http://88.88.88.88:8888";
# Specify the prefix for the name of the object you want to upload.
$upload_dir = 'user-dir-prefix/'
```

- o `$access_key_id`: Enter your AccessKey ID.
- o `$access_key_secret`: Enter your AccessKey secret.
- o `$host`: The format is `https://bucket name.endpoint`. Example: `https://bucket-name.oss-cn-hangzhou.aliyuncs.com`. For more information about endpoints, see the "Endpoint" section in [Terms](#).
- o `$callback_url`: Set the URL of the server to which an upload callback request is sent. This URL is used to communicate between the application server and OSS. After an object is uploaded, OSS sends the object upload information to the application server by using this URL. In this example, set the URL to `$callback_url="http://11.22.33.44:1234";`.
- o `$upload_dir`: Specify the prefix for the name of the object. You can also leave this parameter unspecified.

## Step 2: Configure the client

1. Download the [client source code](#).
2. Decompress the package to a directory. The `D:\aliyun\aliyun-oss-appserver-js` directory is used in the example.
3. Go to the directory. Open the `upload.js` file. Find the following code:

```
// serverUrl specifies the URL of the application server used to obtain information such as the signature and policy. Replace the IP address and port number with your actual information.
serverUrl = 'http://88.88.88.88:8888'
```

4. Set `serverUrl` to the URL of the application server. In this example, specify `serverUrl = 'http://11.22.33.44:1234'`.

## Step 3: Modify CORS configurations

When you use form upload to upload data from the client to OSS, the client includes the `Origin` header in the request and sends the request to OSS by using the browser. OSS verifies the request message that includes the `Origin` header for cross-origin resource sharing (CORS) verification. To use the POST method, configure CORS rules for a bucket.

1. Log on to the [OSS console](#).
- 2.
3. In the left-side navigation pane, choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.

4. Click **Create Rule**. The following figure shows the configurations of a rule.

**Note** To ensure data security, we recommend that you specify the actual domain name from which you want OSS to allow requests in Sources. For more information about CORS configurations, see [Configure CORS rules](#).

## Step 4: Send an upload callback request

1. Start the application server.

In the `/home/aliyun/aliyun-oss-appserver-ruby` directory, run the `ruby appserver.rb 11.22.33.44 1234` command to start the application server.

**Note** Replace the IP address and port number with those of the application server you configured.

2. Start the client.

On the PC, open the `index.html` file in the directory that contains the client source code.

**Notice** The `index.html` file may be incompatible with Internet Explorer 10 or earlier. If you encounter any problems when you use Internet Explorer 10 or earlier, you must perform debugging.

3. Upload an object.

Click **Select File**. Select the file of a specified type. Click **Upload**. After the object is uploaded, the content returned by the callback server is displayed.

## Core code analysis of the application server

The source code of the application server is used to implement signature-based upload and upload callback.

- Signature-based upload

During signature-based upload, the application server responds to the GET message that is sent from the client. An example of the snippet:

```
def get_token()
  expire_syncpoint = Time.now.to_i + $expire_time
  expire = Time.at(expire_syncpoint).utc.iso8601()
  response.headers['expire'] = expire
  policy_dict = {}
  condition_array = Array.new
  array_item = Array.new
  array_item.push('starts-with')
  array_item.push('$key')
  array_item.push($upload_dir)
  condition_array.push(array_item)
  policy_dict["conditions"] = condition_array
  policy_dict["expiration"] = expire
  policy = hash_to_jason(policy_dict)
  policy_encode = Base64.strict_encode64(policy).chomp;
  h = OpenSSL::HMAC.digest('sha1', $access_key_secret, policy_encode)
  hs = Digest::MD5.hexdigest(h)
  sign_result = Base64.strict_encode64(h).strip()
  callback_dict = {}
  callback_dict['callbackBodyType'] = 'application/x-www-form-urlencoded';
  callback_dict['callbackBody'] = 'filename=${object}&size=${size}&mimeType=${mimeType}
&height=${imageInfo.height}&width=${imageInfo.width}';
  callback_dict['callbackUrl'] = $callback_url;
  callback_param = hash_to_jason(callback_dict)
  base64_callback_body = Base64.strict_encode64(callback_param);
  token_dict = {}
  token_dict['accessid'] = $access_key_id
  token_dict['host'] = $host
  token_dict['policy'] = policy_encode
  token_dict['signature'] = sign_result
  token_dict['expire'] = expire_syncpoint
  token_dict['dir'] = $upload_dir
  token_dict['callback'] = base64_callback_body
  response.headers["Access-Control-Allow-Methods"] = "POST"
  response.headers["Access-Control-Allow-Origin"] = ""
  result = hash_to_jason(token_dict)
  result
end
get '/' do
  puts "***** GET "
  get_token()
end
```

- Upload callback

During upload callback, the application server responds to the POST message that is sent from OSS. An example of the snippet:

```
post '/' do
  puts "***** POST"
  pub_key_url = Base64.decode64(get_header('x-oss-pub-key-url'))
  pub_key = get_public_key(pub_key_url)
  rsa = OpenSSL::PKey::RSA.new(pub_key)
  authorization = Base64.decode64(get_header('authorization'))
  req_body = request.body.read
  if request.query_string.empty? then
    auth_str = CGI.unescape(request.path) + "\n" + req_body
  else
    auth_str = CGI.unescape(request.path) + '?' + request.query_string + "\n" + req_body
  end
  valid = rsa.public_key.verify(
    OpenSSL::Digest::MD5.new, authorization, auth_str)
  if valid
    #body({'Status' => 'OK'}.to_json)
    body(hash_to_jason({'Status' => 'OK'}))
  else
    halt 400, "Authorization failed!"
  end
end
```

For more information, see the "(Optional) Step 4: Sign the callback request" section in [Callback](#) of the OSS API Reference.

## 2.2. Application server

### 2.2.1. Set up direct data transfer for mobile apps

This topic describes how to set up a direct data transfer service for a mobile app in less than 30 minutes by using the Security Token Service (STS) policy. Direct data transfer allows a mobile app to directly connect to Object Storage Service (OSS) for data uploads and downloads, while only control flow is sent to the application server.

#### Prerequisites

- OSS is activated. For more information, see [Activate OSS](#).
- A bucket is created. For more information, see [Create buckets](#).

#### Context

In the mobile Internet era, increasing amounts of data are uploaded by using mobile apps. Developers can leave their data storage concerns to OSS and concentrate on app development.

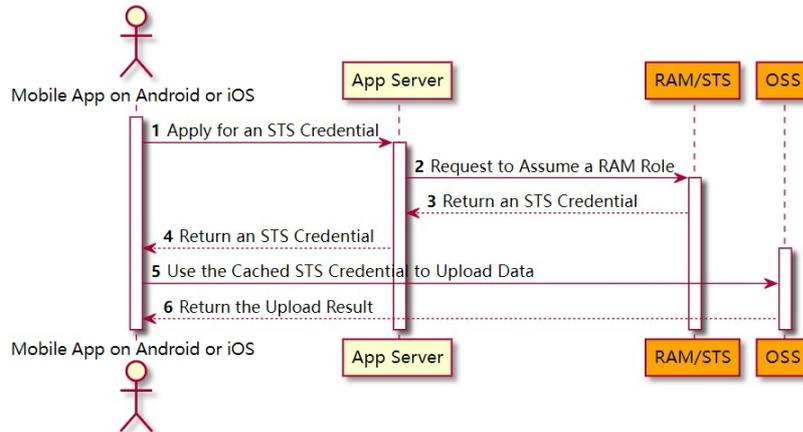
OSS-based direct data transfer for mobile apps has the following benefits:

- **Data security:** OSS allows mobile apps to upload and download data based on flexible authorization and authentication methods, which ensures better security.
- **Cost-effectiveness:** Fewer app servers are required, which minimizes costs. The mobile app is directly connected to OSS for data uploads and downloads, while only the control flow is sent to the app server.
- **High concurrency:** OSS supports concurrent access from a large number of users.
- **Elastic scalability:** OSS provides storage space that can be scaled.

- Data processing: OSS provides Image Processing (IMG) and ApsaraVideo Media Processing to facilitate flexible data processing.

## Process

The following figure shows the development process of the direct data transfer service for mobile apps.



Role description:

- Android or iOS app: an app on a mobile device that applies for and uses an STS credential from the app server.
- OSS: processes data requests from the mobile app.
- Resource Access Management (RAM)/STS: generates a temporary upload credential, which is a token.
- App server: the backend service developed for the Android or iOS app. The app server is used to manage the tokens used for data uploads and downloads by using the app and the metadata of the app-uploaded data.

Procedure:

1. The mobile app applies for a temporary upload credential from the app server.

AccessKey pairs cannot be stored in Android and iOS apps due to security concerns. Therefore, the app must apply for a token from the app server. The token is valid only for a specified period of time. If the app server developer sets the validity period of the token to 30 minutes, the Android or iOS app can use this token to upload data to or download data from OSS within 30 minutes after the token is issued. After 30 minutes, the app must request a new token to upload or download data.

2. The app server checks the validity of the request and then returns a token to the app.
3. The Android or iOS app uses this token to upload data to or download data from OSS.

The following section describes how the app server generates a token and how to use the Android or iOS app to obtain a token.

### Step 1: Activate STS

1. Log on to the [OSS console](#).
2. In the left-side navigation pane, click **Overview**.
3. In the **Common Features** section, choose **Security Token (Authorization for RAM User) > RAM Console**.

#### 4. Click **Start**, and complete authorization.

After the authorization is complete, save the AccessKey ID, AccessKey secret, and RoleArn parameters of the RAM user.

## Step 2: Configure the app server

### 1. Download the app server code.

Language	Download link
PHP	<a href="#">sts-server.zip</a>
Java	<a href="#">AppTokenServerDemo.zip</a>
Ruby	<a href="#">sts-app-server-master.zip</a>
Node.js	<a href="#">sts-app-server-node.zip</a>
Go	<a href="#">test_token_server.zip</a>

### 2. Modify the configuration file.

The following sample code is written in PHP. Each downloaded package for a specific programming language contains a *config.json* configuration file, which contains the following information:

```
{
  "AccessKeyID" : "",
  "AccessKeySecret" : "",
  "RoleArn" : "",
  "TokenExpireTime" : "900",
  "PolicyFile": "policy/bucket_write_policy.txt"
}
```

Parameter description:

- AccessKeyID: Enter the obtained AccessKey ID.
- AccessKeySecret: Enter the obtained AccessKey secret.
- RoleArn: Enter the obtained RoleArn.
- TokenExpireTime: specifies the validity period of the token obtained by using the Android or iOS app. The default value is the minimum value, which is 900 seconds.
- PolicyFile: specifies the file that lists the permissions that the token grants. The default value can be retained.

This topic describes two policy files that define the most common permissions. The files are stored in the policy directory. To use the policy file, replace `$BUCKET_NAME` and `$OBJECT_PREFIX` with specified values.

- `bucket_read_policy.txt`: specifies that the token grants permissions to read data from the specified bucket and objects whose names contain the specified prefix for the account.
- `bucket_write_policy.txt`: specifies that the token grants permissions to write data to the specified bucket and objects whose names contain the specified prefix for the account.

For more information about permissions, see examples and implementation of policies in [Overview](#). Follow the principle of least privilege (PoLP) based on business requirements. If you specify all resources (resource:\*) or all actions (action:\*), security risks such as data leaks may arise because of excess permissions.

 **Warning** The sample code is only for reference. The online system must implement permission isolation for different users or devices. This way, tokens that grant different permissions are generated.

### 3. Run the sample code.

Run the sample code by using the following methods:

- Run the sample code in PHP: After the package is downloaded and decompressed, modify the `config.json` file. Run the `php sts.php` command to generate a token. Deploy the program to the app server address.
- Run the sample code that depends on JDK 1.7 in Java: After the package is downloaded and decompressed, modify the `config.json` file. Reopen the package and run the `java -jar app-token-server.jar (port)` command. If you do not specify a port, the program starts listening on port 7080 by default. You can specify the port used for listening, but you cannot specify a port that already exists.

You must run the `java -jar app-token-server.jar` command to start the services of the app server and call `AssumeRole` to obtain the response.

Sample responses:

```
// Sample success responses.
{
  "StatusCode":200,
  "AccessKeyId":"STS.3p***dgagdasdg",
  "AccessKeySecret":"rpnw09***tGdrddgsR2YrTtI",
  "SecurityToken":"CAES+wMIARKAAZhjH0EUOIhJMqBmJRywXq7MQ/cjLYg80Aho1ek0Jm63Xmhr90c5s`ð
`ð3qaPer8p1YaX1NTDiCFZWFkvlHf1pQhuxfKBC+mRR9KAbHUefqH+rdjZqjTF7p2m1wJXP8S6k+G2MphrUe6TY
BkJ43GhhTVFMuM3BZajY3VjZWOXBIODRIR1FKZjIiEjMzMzE0MjY0NzM5MTE4NjkkMSoLY2xpZGSSDgSDGAGESG
TETqOio6c2RrLWRlbW8vKgoUYWNzOm9zczoqOio6c2RrLWRlbW9KEDExNDg5MzAxMDcyNDY4MThSBTI2ODQyWg9
Bc3N1bWVkUm9sZVVzZXJgAGoSMzMzMTQyNjQ3MzkxMTg2OTExcglzZGstZGVtbzI=",
  "Expiration":"2017-12-12T07:49:09Z"
}
// Sample error responses.
{
  "StatusCode":500,
  "ErrorCode":"InvalidAccessKeyId.NotFound",
  "ErrorMessage":"Specified access key is not found."
}
```

A token in a sample success response consists of the following variables:

- `StatusCode`: the status of the operation to obtain the token. If the token is obtained, 200 is returned.
- `AccessKeyId`: the AccessKey ID obtained when the Android or iOS app initializes OSSClient.
- `AccessKeySecret`: the AccessKey secret obtained when the Android or iOS app initializes OSSClient.

- SecurityToken: the token obtained when the Android or iOS app initializes OSSClient.
- Expiration: the time when the token expires. OSS SDK for Android determines whether the token is valid. If the token becomes invalid, another token is automatically obtained.

Error code description:

- StatusCode: indicates the status of the operation to obtain the token. If the operation fails, 500 is returned.
- ErrorCode: indicates the cause of the error.
- ErrorMessage: indicates the error message.

### Step 3: Download and install the mobile app

1. Download the app source code.

Download link:

- Android: [Download](#)
- iOS: [Download](#)

You can use the mobile app on devices that run Android or iOS to upload images to OSS. Simple upload and resumable upload are supported. If the network quality is poor, we recommend that you use resumable upload. You can also use IMG to resize an image to obtain a thumbnail and add watermarks to the image.

2. Open the mobile app and configure the app parameters.
  - App server: the app server address specified in [Step 2: Configure the app server](#).
  - Destination bucket: the bucket to which the data is uploaded from a mobile app.
  - Region: the region in which the destination bucket is located.
  - OSS object name: The name must contain the prefix specified in the policy configuration file of the app server.
3. Click **Configure**.

### Step 4: Implement direct data transfer for mobile apps

1. Open the mobile app.
2. Click **Select Image**. Select the image to upload and specify the object name.
3. After the object is uploaded, check the upload result in the console.

### Core code analysis

The following code provides examples on how to use OSS SDK for Android and OSS SDK for iOS to initialize OSSClient instances.

- For Android apps:

```
// We recommend that you use OSSAuthCredentialsProvider. The token is automatically updated after it expires.
String stsServer = "App server address such as https://example.com:8080"
OSSCredentialProvider credentialProvider = new OSSAuthCredentialsProvider(stsServer);
// Configure the following parameters.
ClientConfiguration conf = new ClientConfiguration();
conf.setConnectionTimeout(15 * 1000); // The connection timeout period in seconds. Default value: 15.
conf.setSocketTimeout(15 * 1000); // The socket timeout period in seconds. Default value: 15.
conf.setMaxConcurrentRequest(5); // The maximum number of concurrent requests. Default value: 5.
conf.setMaxErrorRetry(2); // The maximum number of retry attempts. Default value: 2.
OSS oss = new OSSClient(getApplicationContext(), endpoint, credentialProvider, conf);
```

- For iOS apps:

```
OSSClient * client;
...
// We recommend that you use OSSAuthCredentialProvider. The token is automatically updated after it expires.
id<OSSCredentialProvider> credential = [[OSSAuthCredentialProvider alloc] initWithAuthServerUrl:@"App server address such as https://example.com:8080"];
client = [[OSSClient alloc] initWithEndpoint:endPoint credentialProvider:credential];
```

## 2.2.2. Set up upload callback for mobile apps

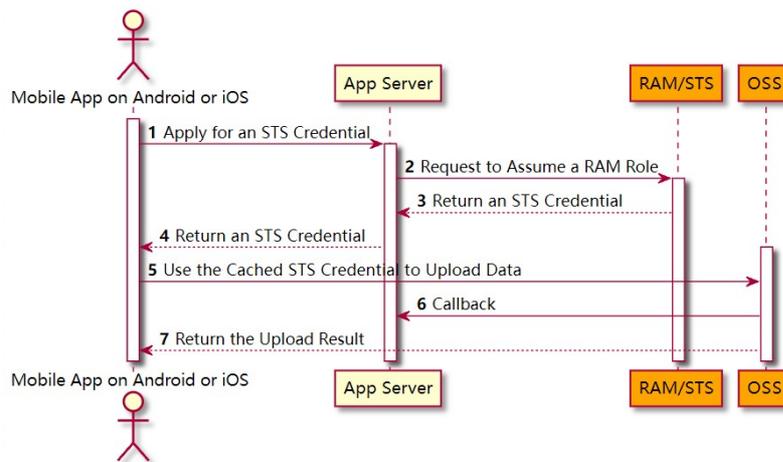
This topic describes how to set up a direct data transfer service for mobile apps based on Object Storage Service (OSS) and configure upload callback.

### Background information

[Set up direct data transfer for mobile apps](#) describes how to set up an OSS-based direct data transfer service for a mobile app. After the OSS-based direct transfer service is set up for an Android or iOS app, a Security Token Service (STS) token is issued to an app user when the app user sends an upload request. The app user can use the STS token to upload data to OSS multiple times until the token becomes invalid. As a result, the app server cannot identify the data uploaded by the app user, which complicates data management for the app developers. To address this issue, OSS provides the upload callback feature.

### Procedure

The following figure shows how upload callback works.



After OSS receives an upload request sent from an Android or iOS app (Step 5), OSS triggers an upload callback task (Step 6) to send a callback request to the app server. Then, the app server returns a response to OSS, and OSS sends the upload result to the Android or iOS app based on the response (Step 7). For more information, see [Callback](#) in OSS API Reference.

## Purposes

- You can use upload callback to send the basic information about the uploaded object to the app server.

The following table describes the variables that can be included in the returned basic information. The format of the returned content is specified when the object is uploaded by using the Android or iOS app.

System variable	Description
bucket	The bucket to which the object is uploaded by using the mobile app.
object	The name of the object after the object is uploaded to OSS by using the mobile app.
etag	The ETag of the uploaded object, which is the ETag field included in the upload result returned to the mobile app user.
size	The size of the uploaded object.
mimeType	The type of the uploaded data.
imageInfo.height	The height of the uploaded image.
imageInfo.width	The width of the uploaded image.
imageInfo.format	The format of the uploaded image. Examples: JPG and PNG.

- You can use upload callback to specify custom callback parameters for the app server to obtain information about the app client.

For example, assume that you are an app developer and you want to obtain information about the app version, operating system version, GPS location, and mobile phone model. In this case, you can specify the following custom parameters for the app when objects are uploaded by using the Android or iOS app:

- `x:version`: the app version
- `x:system`: the operating system version
- `x:gps`: the GPS coordinates
- `x:phone`: the mobile phone model

After you configure the preceding parameters, these parameters are carried when the Android or iOS app client uploads an object to OSS. Then, OSS includes these parameters in `callbackBody` sent to the app server. This way, the app server receives the information about the app client.

## Requirements on the app server for upload callback

- A service that receives POST requests is deployed. This service must have a public IP address.  
Example: `http://example.com/callback.php`.
- This service must correctly respond to OSS, and the response must be in the JSON format. You can specify a custom response. OSS returns the response from the app server to the Android or iOS app. For more information, see [Callback](#) in OSS API Reference.

## Configure upload callback requests on the mobile app side

To trigger upload callback when OSS receives an upload request, you must construct the upload request on the mobile app to include the following content in the upload request:

- `callbackUrl`: The URL of the app server to which the callback request is sent, such as `http://example.com/callback.php`. This URL must be accessible over the Internet.
- `callbackBody`: The preceding system variables OSS returns to the app server. You can specify one or more variables returned in the `callbackBody`.

For example, assume that you are an app developer, and the URL of the app server is `http://example.com/callback.php`. You want to obtain the object name and size and specify `x:phone` as the mobile phone model and `x:system` as the operating system version.

The following section describes two types of requests for upload callback:

- Sample request for upload callback on an iOS app:

```

OSSPutObjectRequest * request = [OSSPutObjectRequest new];
request.bucketName = @"<bucketName>";
request.objectKey = @"<objectKey>";
request.uploadingFileURL = [NSURL URLWithString:@"<filepath>"];
// Set callback parameters.
request.callbackParam = @{
    @"callbackUrl": @"http://example.com/callback.php",
    @"callbackBody": @"filename=${object}&size=${size}&phone=${x:phone}&system=${x:system}"
};
// Define variables.
request.callbackVar = @{
    @"x:phone": @"iphone6s",
    @"x:system": @"ios9.1"
};

```

- Sample request for upload callback on an Android app:

```

PutObjectRequest put = new PutObjectRequest(testBucket, testObject, uploadFilePath);
ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("application/octet-stream");
put.setMetadata(metadata);
put.setCallbackParam(new HashMap<String, String>() {
    {
        put("callbackUrl", "http://example.com/callback.php");
        put("callbackBody", "filename=${object}&size=${size}&phone=${x:phone}&system=${x:system}");
    }
});
put.setCallbackVars(new HashMap<String, String>() {
    {
        put("x:phone", "IPOHE6S");
        put("x:system", "YunOS5.0");
    }
});

```

## Sample upload callback request received by the app server

The upload callback requests received by the app server differ in URLs and callbackBody content based on the preceding configurations of callbackUrl and callbackBody. Example:

```

POST /index.html HTTP/1.0
Host: 121.43.113.8
Connection: close
Content-Length: 81
Content-Type: application/x-www-form-urlencoded
User-Agent: ehttp-client/0.0.1
authorization: kKQeGTRccDKyHB3H9vF+xYMSrmmMZjzzl2/kdD1ktNVgbWE***G0G2SU/RaHBovRCE8OkQDjC3uG33esH2txA==
x-oss-pub-key-url: aHR0cDovL2dvc3NwdWJsaWMuYWxpY2RuLmNv***YWxsYmFja19wdWJfa2V5X3YxLnBlbQ==
filename=test.txt&size=5&phone=iphone6s&system=ios9.1

```

For more information, see [Callback](#) in OSS API Reference.

## The app server determines whether the upload callback request is sent from OSS

If your app server is attacked and receives invalid requests, the app server needs to determine whether the callback request is sent from OSS.

To determine whether the callback request is sent from OSS, the app server uses the `x-oss-pub-key-url` and `authorization` parameters in the header sent by OSS for RSA signature verification. Only callback requests that pass RSA signature verification are considered as from OSS. The sample programs provided in this topic include the parameters for RSA signature verification for your reference.

## The app server processes the callback request

After the app server determines that the callback request is sent from OSS, the format of the request to the app server is specified. Example:

```
filename=test.txt&size=5&phone=iphone6s&system=ios9.1
```

The app server can parse the content sent by OSS to obtain the desired data. After the data is obtained, the app server can store the data for subsequent management.

## OSS processes the response from the app server

One of the following scenarios may occur:

- OSS sends the callback request to the app server. The app server fails to receive the request or is inaccessible. In that case, OSS sends the status code 203 to the Android or iOS app. However, data has been stored in OSS.
- The app server receives the callback request from OSS and responds to OSS properly. In that case, OSS returns the status code 200 and the response from the app server to the Android or iOS app.

## Download sample programs

The sample programs only examine the signature received by the app server. You must manually add the code that is executed to parse the format of the callback request received by the app server.

- Java
  - Click [AppCallbackServer.zip](#) to download a program.
  - To run the sample program, decompress the downloaded package and run the `java -jar oss-callback-server-demo.jar 9000` file. 9000 is the default port. You can change the port number.

 **Note** The JAR package example runs on OSS SDK for Java 1.7. If an error occurs, modify the code based on the code sample. This package contains the code for a Maven project.

- PHP
  - Click [callback-php-demo.zip](#) to download a program.
  - To run the sample program, decompress the downloaded package to an Apache environment. This specific environment is required to obtain some headers due to PHP-specific features. Modify the sample code based on the actual environment.
- Python
  - Click [callback\\_app\\_server.py.zip](#) to download a program.

- To run the sample program, decompress the downloaded package and run the `python callback_app_server.py` file. This program functions as a simple HTTP server. To run this program, you may need to install the dependencies for RSA.
- Ruby
  - Click `oss-callback-server` to download a program.
  - To run the sample program, run the `ruby aliyun_oss_callback_server.rb` file.

## 2.3. Use ECS instances to configure reverse proxy for access to OSS

### 2.3.1. Use an ECS instance that runs Ubuntu to configure a reverse proxy for access to OSS

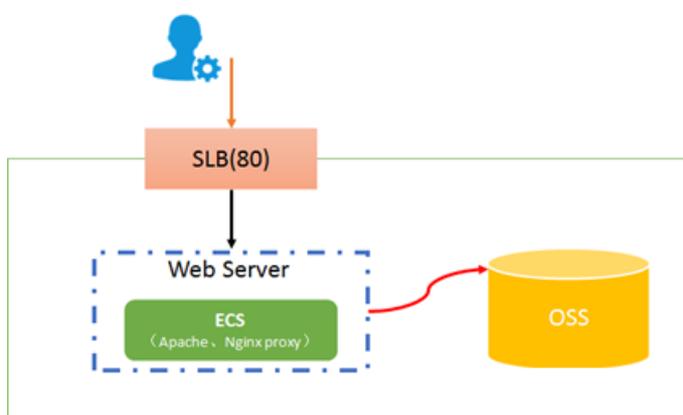
The IP address used to access an Object Storage Service (OSS) bucket dynamically changes. You can configure a reverse proxy on an Elastic Compute Service (ECS) instance to access the OSS bucket by using a static IP address.

#### Context

OSS uses Restful APIs to provide services. You can access a bucket by using the default endpoint of the bucket or custom domain names that are mapped to the bucket. However, you may need to use a static IP address to access OSS in some scenarios.

- For security reasons, some enterprises must configure outbound rules to specify that internal employees and business systems can access only the specified public IP addresses. However, the IP addresses used to access a bucket in OSS dynamically change. In this case, enterprises must frequently modify firewall rules.
- Limited by the network architecture of Alibaba Finance Cloud, internal network-specific buckets in Alibaba Finance Cloud can be accessed only by requests from Alibaba Finance Cloud but not those from the Internet.

To resolve these issues, you can use an ECS instance to configure a reverse proxy for access to OSS.



#### Procedure

1. Create an ECS instance that runs Ubuntu. Make sure that the instance and the bucket you want to

access are located within the same region.

The Ubuntu 18.04 (64-bit) system is used in this example. For more information, see [Create an ECS instance](#).

2. Log on to the ECS instance as the root user. Update the APT sources.

```
root@test:~# apt-get update
```

3. Install NGINX.

```
root@test:~# apt-get install nginx
```

 **Note** By default, the files of NGINX are installed in the following paths:

/usr/sbin/nginx	Stores the NGINX executable.
/etc/nginx	Stores configuration files.
/usr/share/nginx	Stores static files.
/var/log/nginx	Stores log files.

4. Open the configuration file of NGINX.

```
root@test:~# vi /etc/nginx/nginx.conf
```

5. Add the following content to the http context of the configuration file:

```
server {
    listen 80;
    server_name 47.**.**.73;
    location / {
        proxy_pass http://bucketname.oss-cn-beijing-internal.aliyuncs.com;
        #proxy_set_header Host $host;
    }
}
```

- `server_name`: the IP address used to provide the reverse proxy service, which is the public IP address of the ECS instance.
- `proxy_pass`: the endpoint for redirection.
  - When the ECS instance and the bucket that you want to access are located within the same region, specify the internal endpoint of the bucket. For more information about endpoints, see [OSS domain names](#).
  - When the ECS instance and the bucket that you want to access are located in different regions, specify the public endpoint of the bucket.
  - For security reasons, when you access an image or web page object in a bucket by using the default domain name of the bucket in a browser, the object is downloaded. To preview the object in your browser, map a custom domain name to the bucket in which the object is stored and add the custom domain name to the value of `proxy_pass`. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).
- `proxy_set_header Host $host`: If you add this parameter, the host value is replaced with the IP address of the ECS instance when NGINX sends a request to OSS. You must add this parameter in the following scenarios:

- Signature errors occur.
- The custom domain name that is mapped to the bucket is resolved to the public IP address of the ECS instance. You must preview image or web page objects in the bucket by using a browser. You can map the custom domain name to the bucket for which a reverse proxy is configured without adding a CNAME record for the custom domain name. In this case, you can set proxy\_pass to the internal or public endpoint of the bucket. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).

#### Notice

- A demo environment is used in this topic as an example. For data security reasons, we recommend that you configure the https context. For more information, see [Configure HTTPS for your custom domain name in OSS by using reverse proxy](#).
- You can configure a reverse proxy for only one bucket if you use this configuration method.

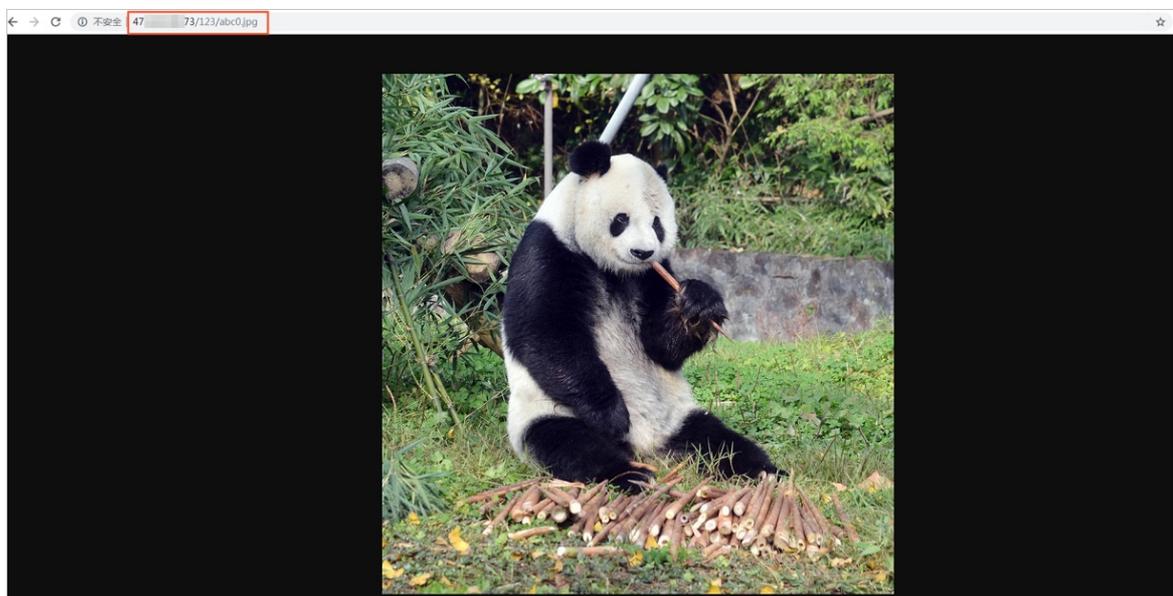
6. Go to the directory in which the NGINX executable file is located. Start NGINX.

```
root@test:~# cd /usr/sbin/  
root@test:~# ./nginx
```

7. Enable TCP port 80 of the ECS instance.

By default, NGINX uses TCP port 80. Therefore, you must enable TCP port 80 when you configure a security group for the ECS instance. For more information, see [Add a security group rule](#).

8. Add the object path to the public IP address of the ECS instance to access OSS resources.



## References

[Use an ECS instance that runs CentOS to configure a reverse proxy for access to OSS](#)

## 2.3.2. Use an ECS instance that runs CentOS to configure a reverse proxy for access to OSS

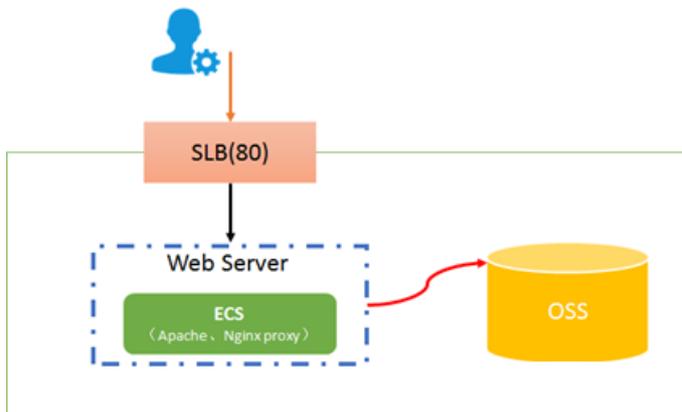
The IP address used to access an Object Storage Service (OSS) bucket dynamically changes. You can configure a reverse proxy on an Elastic Compute Service (ECS) instance to access the OSS bucket by using a static IP address.

## Context

OSS uses Restful APIs to provide services. You can access a bucket by using the default endpoint of the bucket or custom domain names that are mapped to the bucket. However, you may need to use a static IP address to access OSS in some scenarios.

- For security reasons, some enterprises must configure outbound rules to specify that internal employees and business systems can access only the specified public IP addresses. However, the IP addresses used to access a bucket in OSS dynamically change. In this case, enterprises must frequently modify firewall rules.
- Limited by the network architecture of Alibaba Finance Cloud, internal network-specific buckets in Alibaba Finance Cloud can be accessed only by requests from Alibaba Finance Cloud but not those from the Internet.

To resolve these issues, you can use an ECS instance to configure a reverse proxy for access to OSS.



## Procedure

1. Create an ECS instance that runs CentOS. Make sure that the instance and the bucket you want to access are located within the same region.

The CentOS 7.6 (64-bit) system is used in this example. For more information, see [Create an ECS instance](#).

2. Log on to the ECS instance as the root user. Install NGINX.

```
root@test:~# yum install -y nginx
```

**Note** By default, the files of NGINX are installed in the following paths:

/usr/sbin/nginx	Stores the NGINX executable.
/etc/nginx	Stores configuration files.
/usr/share/nginx	Stores static files.
/var/log/nginx	Stores log files.

3. Open the configuration file of NGINX.

```
root@test:~# vi /etc/nginx/nginx.conf
```

#### 4. Add the following content to the http context of the configuration file:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name 47.**.**.43;
    root /usr/share/nginx/html;
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    location / {
        proxy_pass https://bucketname.oss-cn-beijing-internal.aliyuncs.com;
        proxy_set_header Host $host;
    }
}
```

- `server_name`: the IP address used to provide the reverse proxy service, which is the public IP address of the ECS instance.
- `proxy_pass`: the endpoint for redirection.
  - When the ECS instance and the bucket that you want to access are located within the same region, specify the internal endpoint of the bucket. For more information about endpoints, see [OSS domain names](#).
  - When the ECS instance and the bucket that you want to access are located in different regions, specify the public endpoint of the bucket.
  - For security reasons, when you access an image or web page object in a bucket by using the default domain name of the bucket in a browser, the object is downloaded. To preview the object in your browser, map a custom domain name to the bucket in which the object is stored and add the custom domain name to the value of `proxy_pass`. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).
- `proxy_set_header Host $host`: If you add this parameter, the host value is replaced with the IP address of the ECS instance when NGINX sends a request to OSS. You must add this parameter in the following scenarios:
  - Signature errors occur.
  - The custom domain name that is mapped to the bucket is resolved to the public IP address of the ECS instance. You must preview image or web page objects in the bucket by using a browser. You can map the custom domain name to the bucket for which a reverse proxy is configured without adding a CNAME record for the custom domain name. In this case, you can set `proxy_pass` to the internal or public endpoint of the bucket. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).

 **Note** A demo environment is used in this topic as an example. For data security reasons, we recommend that you configure the https context. For more information, see [Configure HTTPS for your own domain name in OSS by using reverse proxy](#).

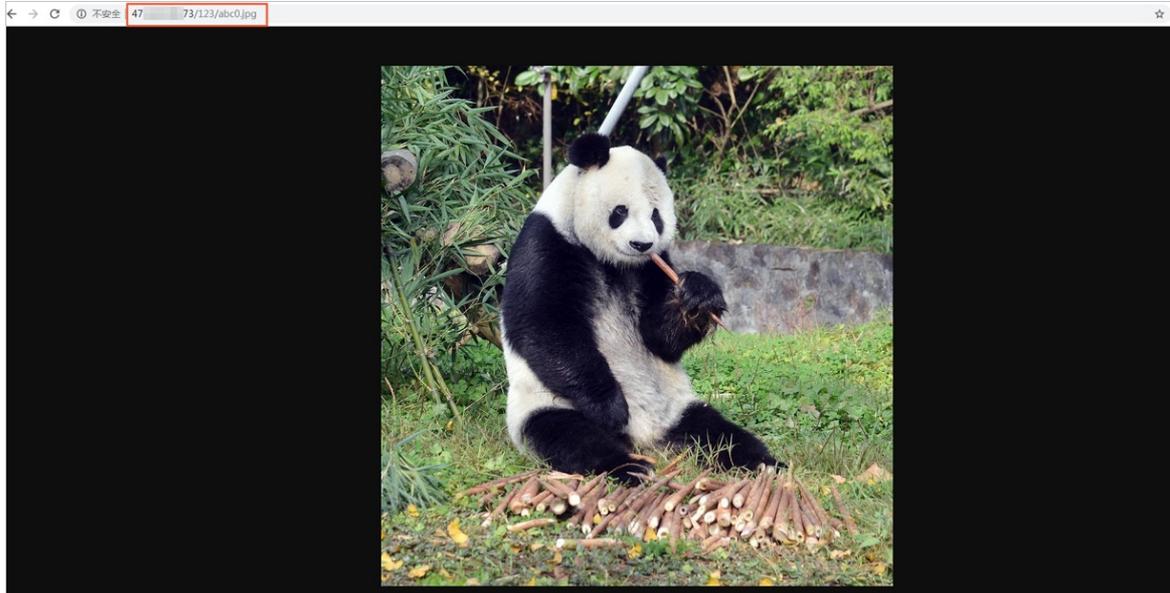
#### 5. Go to the directory in which the NGINX executable file is located. Start NGINX.

```
root@test:~# cd /usr/sbin/
root@test:~# ./nginx
```

6. Enable TCP port 80 of the ECS instance.

By default, NGINX uses TCP port 80. Therefore, you must enable TCP port 80 when you configure a security group for the ECS instance. For more information, see [Add a security group rule](#).

7. Add the object path to the public IP address of the ECS instance to access OSS resources.



If the object ACL is private, you must sign the object URL. For more information, see [Add signatures to URLs](#).

## References

[Use an ECS instance that runs Ubuntu to configure a reverse proxy for access to OSS](#)

### 2.3.3. Use an ECS instance that runs Windows to configure a reverse proxy for access to OSS

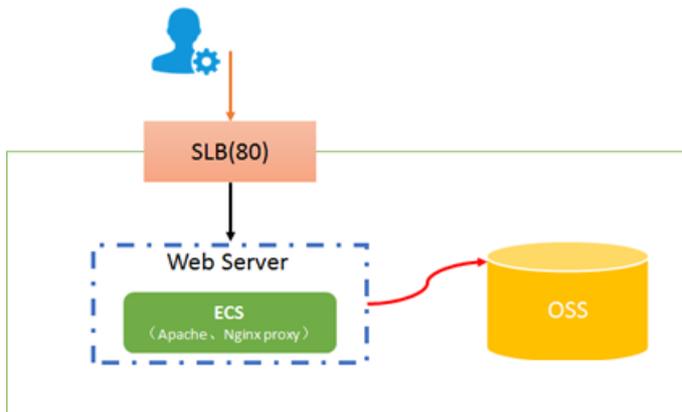
The IP address used to access an Object Storage Service (OSS) bucket dynamically changes. You can configure a reverse proxy on an Elastic Compute Service (ECS) instance to access the OSS bucket by using a static IP address.

#### Context

You can access a bucket by using the default endpoint of the bucket or custom domain names that are mapped to the bucket. However, you may need to use a static IP address to access OSS in some scenarios.

- For security reasons, some enterprises must configure outbound rules to specify that internal employees and business systems can access only the specified public IP addresses. However, the IP addresses used to access a bucket in OSS dynamically change. In this case, enterprises must frequently modify firewall rules.
- Limited by the network architecture of Alibaba Finance Cloud, internal network-specific buckets in Alibaba Finance Cloud can be accessed only by requests from Alibaba Finance Cloud.

To resolve these issues, you can use an ECS instance to configure a reverse proxy for access to OSS.



## Procedure

1. Create an ECS instance that runs Windows Server. Make sure that the instance and the bucket you want to access are located within the same region.

The Windows Server 2019 Datacenter edition 64-bit system is used in this example. For more information, see [Quick start for Windows instances](#).

2. Download NGINX and decompress the package.

NGINX-1.19.2 is used in this example. For more information about the downloading URLs of NGINX, visit [Nginx](#).

3. Modify the configuration file.

- i. Go to the conf directory and open the *nginx.conf* file by using a notepad.
- ii. Modify the content of the configuration file.

```
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    keepalive_timeout 65;
    server {
        listen 80;
        server_name 47.**.**.43;
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root html;
        }
        location / {
            proxy_pass http://bucketname.oss-cn-hangzhou-internal.aliyuncs.com;
            #proxy_set_header Host $host;
        }
    }
}
```

- **server\_name**: the IP address used to provide the reverse proxy service, which is the public IP address of the ECS instance.

- `proxy_pass`: the endpoint for redirection.
  - When the ECS instance and the bucket that you want to access are located within the same region, specify the internal endpoint of the bucket. For more information about endpoints, see [OSS domain names](#).
  - When the ECS instance and the bucket that you want to access are located within different regions, specify the public endpoint of the bucket.
  - For security reasons, when you access an image or web page object in a bucket by using the default domain name of the bucket in a browser, the object is downloaded. To preview the object in your browser, map a custom domain name to the bucket in which the object is stored and add the custom domain name to the value of `proxy_pass`. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).
- `proxy_set_header Host $host`: If you add this parameter, the host value is replaced with the IP address of the ECS instance when NGINX sends a request to OSS. You must add this parameter in the following scenarios:
  - Signature errors occur.
  - The custom domain name that is mapped to the bucket is resolved to the public IP address of the ECS instance. You must preview image or web page objects in the bucket by using a browser. You can map the custom domain name to the bucket for which a reverse proxy is configured without adding a CNAME record for the custom domain name. In this case, you can set `proxy_pass` to the internal or public endpoint of the bucket. For more information about how to map a custom domain name to a bucket, see [Map custom domain names](#).

 Notice

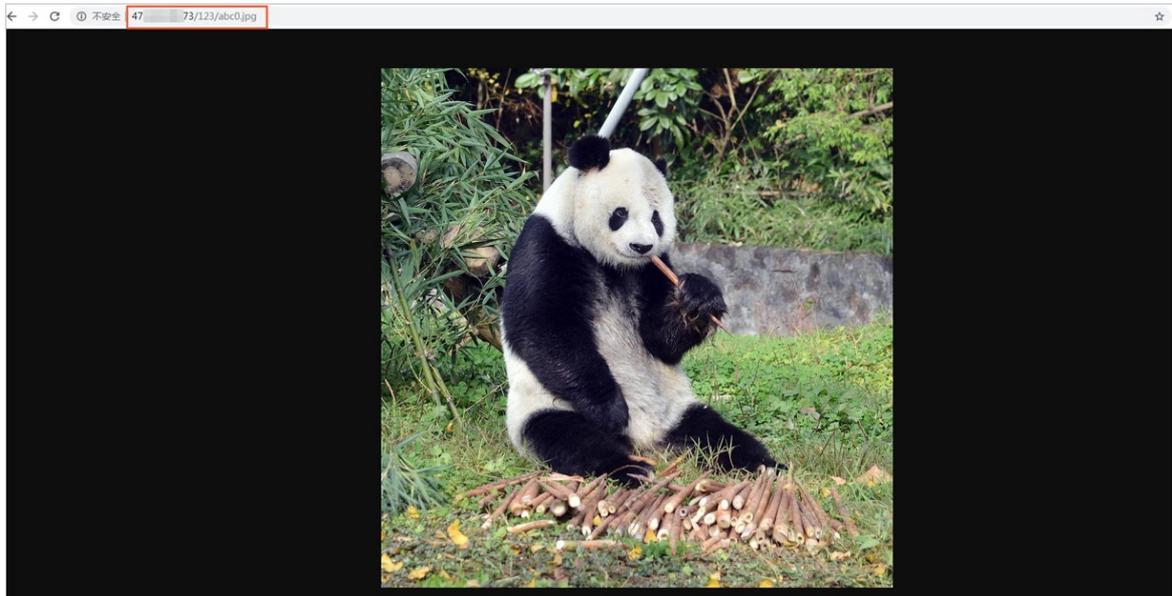
- A demo environment is used in this topic as an example. For data security reasons, we recommend that you configure the https context.
- You can configure a reverse proxy for only one bucket if you use this configuration method.

4. Go to the directory in which the NGINX executable file is located. Double-click `nginx.exe` to start NGINX.

5. Enable TCP port 80 of the ECS instance.

By default, NGINX uses TCP port 80. Therefore, you must enable TCP port 80 when you configure a security group for the ECS instance. For more information, see [Add a security group rule](#).

6. Add the object path to the public IP address of the ECS instance to access OSS resources.



## References

- [Use an ECS instance that runs Ubuntu to configure a reverse proxy for access to OSS](#)
- [Use an ECS instance that runs CentOS to configure a reverse proxy for access to OSS](#)

## 2.4. Check data transmission integrity by using CRC-64

An error may occur when data is transferred between the client and server. OSS can return the CRC-64 value of objects uploaded through any of the methods provided. The client can compare the CRC-64 value with the value calculated on the local machine to verify data integrity.

### Context

OSS calculates the CRC-64 value for newly uploaded objects and stores the result as metadata of the object. OSS then adds the `x-oss-hash-crc64ecma` header to the returned response header, which indicates its CRC-64 value. This CRC-64 value is calculated based on [Standard ECMA-182](#).

If the object exists in OSS before CRC-64 goes online, OSS does not calculate its CRC-64 value. Therefore, its CRC-64 value is not returned when the object is obtained.

### Operation instructions

- The `PutObject`, `AppendObject`, `PostObject`, and `MultipartUploadPart` operations return the corresponding CRC-64 value. The client can obtain the CRC-64 value returned by the server after the upload is complete and can check it against the value calculated on the local machine.
- When the `MultipartComplete` operation is called, the CRC-64 value of the entire object is returned if each part has a CRC-64 value. However, if a part is uploaded before the CRC-64 goes online, the CRC-64 value is returned.
- The `GetObject`, `HeadObject`, and `GetObjectMeta` operations return the corresponding CRC-64 value (if any). After the `GetObject` operation is complete, the client can obtain the CRC-64 value returned by the server and check it against the value calculated on the local machine.

 **Note** The GET request that includes the Range header returns the CRC-64 value of the entire object.

- The newly generated object or part may not have the CRC-64 value after copy related operations such as CopyObject and UploadPartCopy are complete.

## Examples

The following code provides an example on how to use Python to use CRC-64 values to verify data transmission integrity:

### 1. Calculate the CRC-64 value.

```
import oss2
import crcmod
import random
import string

do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693, initCrc=0, xorOut=0xffffffffffffffff, rev=True)

def check_crc64(local_crc64, oss_crc64, msg="check crc64"):
    if local_crc64 != oss_crc64:
        print("{0} check crc64 failed. local:{1}, oss:{2}.".format(msg, local_crc64, oss_crc64))
        return False
    else:
        print("{0} check crc64 ok.".format(msg))
        return True

def random_string(length):
    return ''.join(random.choice(string.ascii_lowercase) for i in range(length))

bucket = oss2.Bucket(oss2.Auth('yourAccessKeyId', 'yourAccessKeySecret'),
                    'https://oss-cn-hangzhou.aliyuncs.com', 'yourBucketName')
```

### 2. Verify CRC-64 values for Put Object.

```
content = random_string(1024)
key = 'normal-key'
result = bucket.put_object(key, content)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(content))
check_crc64(local_crc64, oss_crc64, "put object")
```

### 3. Verify CRC-64 values for Get Object.

```
result = bucket.get_object(key)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(result.resp.read()))
check_crc64(local_crc64, oss_crc64, "get object")
```

### 4. Verify CRC-64 values for UploadPart and Multipart Complete.

```

part_info_list = []
key = "multipart-key"
result = bucket.init_multipart_upload(key)
upload_id = result.upload_id
part_1 = random_string(1024 * 1024)
result = bucket.upload_part(key, upload_id, 1, part_1)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_1))
# Check whether the uploaded part_1 data is complete
check_crc64(local_crc64, oss_crc64, "upload_part object 1")
part_info_list.append(PartInfo(1, result.etag, len(part_1)))
part_2 = random_string(1024 * 1024)
result = bucket.upload_part(key, upload_id, 2, part_2)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_2))
# Check whether the uploaded part_2 data is complete
check_crc64(local_crc64, oss_crc64, "upload_part object 2")
part_info_list.append(PartInfo(2, result.etag, len(part_2)))
result = bucket.complete_multipart_upload(key, upload_id, part_info_list)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_2, do_crc64(part_1)))
# Check whether the final object in OSS is consistent with the local file
check_crc64(local_crc64, oss_crc64, "complete object")

```

## OSS SDK support

The following table describes OSS SDKs, some of which support CRC-64 for downloads and uploads.

SDK	Support for CRC	Example
Java SDK	Yes	<a href="#">CRCSample.java</a>
Python SDK	Yes	<a href="#">object_check.py</a>
PHP SDK	No	None
C# SDK	No	None
C SDK	Yes	<a href="#">oss_crc_sample.c</a>
JavaScript SDK	No	None
Go SDK	Yes	<a href="#">crc_test.go</a>
Ruby SDK	No	None
iOS SDK	Yes	<a href="#">OSSCrc64Tests.m</a>
Android SDK	Yes	<a href="#">CRC64Test.java</a>

# 3. Data lake

## 3.1. Alibaba Cloud ecosystem

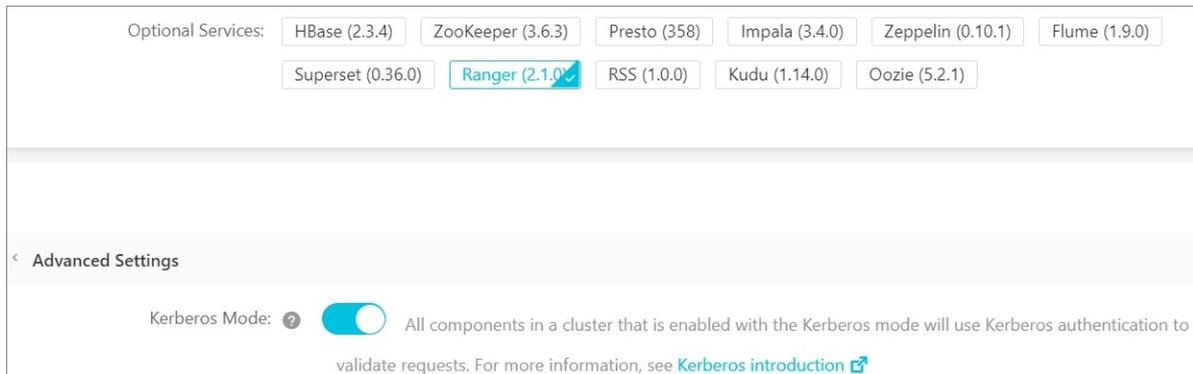
### 3.1.1. Configure Apache Ranger for authentication in EMR clusters

Apache Ranger provides a centralized permission management framework that allows fine-grained access control on multiple components in the Hadoop ecosystem. This topic describes how to integrate the OSS-HDFS service with Ranger and grant permissions to a user to access OSS-HDFS.

#### Prerequisites

- A high-security cluster of E-MapReduce (EMR) V5.6.0 or later is created.

The Ranger service is selected when the cluster is created. For more information about how to create a cluster, see [Create a cluster](#).



- OSS-HDFS is enabled for a bucket and permissions are granted to access OSS-HDFS. For more information about how to enable OSS-HDFS and grant access permissions, see [Enable OSS-HDFS and grant access permissions](#).

#### Integrate OSS-HDFS with Ranger

- 1.
- 2.
3. Deploy client configurations.
  - i. In the left-side navigation pane, choose **Cluster Service > HDFS**.
  - ii. On the HDFS service page, click the **Configure** tab. Then, click **Deploy Client Configuration** in the upper-right corner.
  - iii. In the **Cluster Activities** dialog box, configure Description and click **OK**.
  - iv. In the **Confirm** message, click **OK**.  
You can click **History** in the upper-right corner to view the execution status and progress.
4. Restart Jindofsx Namespace Service.
  - i. In the upper-right corner of the **JindoData** service page, choose **Actions > Restart Jindofsx Namespace Service**.

- ii. In the **Cluster Activities** dialog box, configure **Description** and click **OK**.
- iii. In the **Confirm** message, click **OK**.

5.

6.

7. Create a TGT.

i.

ii.

iii. Create a TGT.

■

- Method 2: Use a keytab file to create a TGT.

The *test.keytab* file generated in [Step 6](#) is stored in the */root/* directory of the *emr-header-1* node. You must run the `cp /root/test.keytab /home/test/` command to copy the file to the */home/test/* directory of the current node. Then, run the following command to create a TGT:

iv.

## Grant access permissions on OSS-HDFS

Perform the following steps to grant the test user access permissions on the *oss://examplebucket/dir/* directory.

1. Access the Ranger web UI. For more information, see [Overview](#).
2. On the Ranger web UI, click **emr-oss**.



3. Grant execute permissions on the *dir/* directory.
  - i. Click **Add New Policy** in the upper-right corner.
  - ii. On the **Edit Policy** page, configure the parameters that are described in the following table.

Parameter	Description
<b>Policy Name</b>	
<b>Path</b>	In this example, this parameter is set to <i>examplebucket/dir</i> .
<b>Select User</b>	The user to whom you want to attach this policy. In this example, this parameter is set to <i>test</i> .
<b>Permissions</b>	The permissions that you want to grant. In this example, this parameter is set to <i>Execute</i> .

- iii. Click **Add**.
4. Access OSS-HDFS.
    - i. Log on to the emr-header-1 node of your cluster in SSH mode. For more information, see [Log on to a cluster](#).
    - ii. Run the following command to switch to the test user:

```
su test
```

- iii. Run the following command to access OSS-HDFS:

```
hadoop fs -ls oss://examplebucket/dir/
```

If you are not granted permissions to access the OSS-HDFS directory, the following information is displayed:

```
org.apache.hadoop.security.AccessControlException: Permission denied: user=test, access=READ_EXECUTE, resourcePath="examplebucket/dir/"
```

## 3.1.2. Use JindoFuse to access the OSS-HDFS service

Alibaba Cloud OSS-HDFS (JindoFS) service uses JindoFuse to provide POSIX support. JindoFuse allows you to mount objects from the JindoFS service to a local file system. This allows you to perform operations on objects from the JindoFS service in the same manner as a local file system.

### Step 1: Configure the client

- Configuration directory

Decompress the installation package that you downloaded. In this example, the package is decompressed in the `/usr/lib/jindosdk-4.0.0` directory.

```
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${JINDOSDK_HOME}/lib/native/
```

- Configuration file

Use the configuration file in the `.ini` format. In this example, the name of the configuration file after compilation is `jindosdk.cfg`. The following code shows the configuration items in the configuration file:

```
[common]
logger.dir = /tmp/fuse-log
[jindodls]
# The endpoint of the bucket for which the OSS-HDFS service is enabled. For example, if t
he bucket for which the OSS-HDFS service is enabled is located in the China (Hangzhou) re
gion, set the endpoint to cn-hangzhou.oss-dls.aliyuncs.com.
fs.oss.endpoint = <your_endpoint>
# The AccessKey ID and AccessKey secret that are used to access the JindoFS service. An A
libaba Cloud account has permissions to call all API operations. If you use the AccessKey
pair of an Alibaba Cloud account, security risks may occur. We recommend that you use a R
AM user to call API operations or perform routine O&M. To create a RAM user, log on to th
e RAM console.
fs.oss.accessKeyId = <your_key_id>
fs.oss.accessKeySecret = <your_key_secret>
```

## Step 2: Mount FUSE

1. Run the following command to create a mount point:

```
mkdir -p <mount-point>
```

2. Run the following command to mount Filesystem in USErspace (FUSE):

```
jindo-fuse <mount_point> -ouri=[<oss_path>]
```

You must set *-ouri* to the dls path that you want to map. The path can be the root directory or a subdirectory of the bucket. After you run the command, a daemon process in the background starts to mount the *<oss\_path>* that you specified to the mount point of the local file system. The mount point is specified by *<mount\_point>*.

For more information about the mount options that you can configure when you mount FUSE, see [Mount options](#).

## Step 3: Access the mount directory

After you mount FUSE to the local path, */mnt/jindodls/* in this example, run the following commands to perform the basic operations of JindoFuse:

- Create a directory

```
mkdir /mnt/oss/dir1
```

- List all directories in */mnt/oss/*

```
ls /mnt/oss/
```

- Write an object

```
echo "hello world" > /mnt/oss/dir1/hello.txt
```

- Read the object

```
cat /mnt/oss/dir1/hello.txt
```

- Delete the directory

```
rm -rf /mnt/oss/dir1/
```

## FAQ

- If you use JindoSDK to call API operations, you can view detailed error messages when errors are reported. If you use JindoFuse, you can view only the preset error messages of the operating system.

```
ls: /mnt/oss/: Input/output error
```

To identify the cause of an error, you must find the *jindosdk.log* file in the path that is specified by the *logger.dir* configuration item of JindoSDK. The following is a common authentication error message that may appear when you use JindoFuse:

```
EMMDD HH:mm:ss jindofs_connectivity.cpp:13] Please check your Endpoint/Bucket/RoleArn.
Failed test connectivity, operation: mkdir, errMsg: [RequestId]: 618B8183343EA53531C62B74
[HostId]: oss-cn-shanghai-internal.aliyuncs.com [ErrorMessage]: [E1010]HTTP/1.1 403 Forbidden ...
```

If the preceding error message appears, check whether the endpoint, the bucket, and the role ARN are properly configured. For more information about how to configure the endpoint, the bucket, and the role ARN, see [Get started with OSS-HDFS](#).

If a program error occurs, contact [technical support](#).

## How do I troubleshoot JindoFuse errors?

- You can uninstall FUSE by using one of the following methods:
  - Manually uninst all FUSE

```
umount <mount_point>
```

- Enable automatic uninst allation

Before you enable automatic uninst allation, you must inst all fuse3.

- Install fuse3

Run the following commands to inst all fuse3:

- CentOS

```
yum install -y fuse3
```

- Debian

```
apt install -y fuse3
```

- Enable automatic uninst allation of FUSE

```
-oauto_unmount
```

## How do I uninst all FUSE?

## Appendix 1: Supported operations

The following table describes the POSIX-based API operations that are supported by JindoFuse.

Operation	Description
-----------	-------------

Operation	Description
getattr()	Queries the attributes of a file. This operation is similar to the <b>ls</b> command.
mkdir()	Creates a directory. This operation is similar to the <b>mkdir</b> command.
rmdir()	Deletes a directory. This operation is similar to the <b>rm -rf</b> command.
unlink()	Deletes a file. This operation is similar to the <b>unlink</b> command.
rename()	Renames a file or a directory. This operation is similar to the <b>mv</b> command.
read()	Reads data in sequence.
pread()	Reads data at random.
write()	Writes data in sequence.
flush()	Flushes data from the memory to the kernel cache.
fsync()	Flushes data from the memory to disks.
release()	Releases a file.
readdir()	Reads a directory.
create()	Creates a file.
open() O_APPEND	Opens a file by using the append mode.
open() O_TRUNC	Opens a file by using the overwrite mode.
ftruncate()	Truncates an opened file.
truncate()	Truncates an unopened file. This operation is similar to the <b>truncate -s</b> command.
chmod()	Modifies the permissions on a file. This operation is similar to the <b>chmod</b> command.
access()	Queries the permissions on a file.
utimes()	Modifies the time when a file is stored and modified.

## Appendix 2: Mount options

The following table describes the options that you can configure to use JindoFuse to mount objects from the JindoFS service to a local file system.

Option	Required	Description	Example
--------	----------	-------------	---------

Option	Required	Description	Example
uri	Yes	The dls path that you want to map. The path can be the root directory of the bucket - <i>ouri=oss://bucket.endpoint/</i> or a subdirectory of the bucket, for example, - <i>ouri=oss://bucket.endpoint/subdir.</i>	<i>-ouri=oss://examplebucket.cn-beijing.oss-dls.aliyuncs.com/</i>
f	No	Starts the FUSE process. By default, a daemon process is used to start the FUSE process in the background. If you use this option, we recommend that you enable terminal logs.	<i>-f</i>
d	No	Enables the debug mode. If you enable the debug mode, the FUSE process starts in the foreground. If you use this option, we recommend that you enable terminal logs.	<i>-d</i>
auto_unmount	No	Automatically unmounts the mount point after the FUSE process exits.	<i>-oauto_unmount</i>
ro	No	Mounts objects from the JindoFS service in read-only mode. After you enable this option, you cannot perform write operations.	<i>-oro</i>
direct_io	No	Allows file read and write without the need for page cache.	<i>-odirect_io</i>
kernel_cache	No	Uses the kernel cache to optimize read performance.	<i>-okernel_cache</i>
auto_cache	No	Enables automatic caching by default. Unlike <code>kernel_cache</code> , if the file size or the time at which the file is modified changes, the cache becomes invalid.	<i>-oauto_cache</i>
entry_timeout	No	The retention period of the file name in the cache if the file is read. Unit: seconds. This option is used to optimize performance. The value of 0 indicates that the file name is not cached. Default value: 0.1.	<i>-oentry_timeout=60</i>

Option	Required	Description	Example
attr_timeout	No	The retention period of the file attributes in the cache. Unit: seconds. This option is used to optimize performance. The value of 0 indicates that the file attributes are not cached. Default value: 0.1.	-oattr_timeout=60
negative_timeout	No	The retention period of the file name in the cache if the file fails to be read. Unit: seconds. This option is used to optimize performance. The value of 0 indicates that the file name is not cached. Default value: 0.1.	-onegative_timeout=0

### Appendix 3: Configuration items

Configuration item	Description
logger.dir	The directory in which the logs are stored. Default value: <i>/tmp/jindodata-log</i> .
logger.sync	The mode in which logs are returned. Default value: <i>false</i> . Valid values: <ul style="list-style-type: none"> <li><i>true</i>: returns logs in synchronous mode.</li> <li><i>false</i>: returns logs in asynchronous mode.</li> </ul>
logger.consolelogger	Specifies whether to display logs. Default value: <i>false</i> . Valid values: <ul style="list-style-type: none"> <li><i>true</i>: displays logs on the terminal.</li> <li><i>false</i>: does not display logs.</li> </ul>
logger.level	Returns logs whose levels are greater than or equal to the value of this configuration item. Valid values: 0 to 6. The following items show the mappings between the values of this configuration item and the log levels: <ul style="list-style-type: none"> <li>0: TRACE</li> <li>1: DEBUG</li> <li>2: INFO</li> <li>3: WARN</li> <li>4: ERROR</li> <li>5: CRITICAL</li> <li>6: OFF</li> </ul>
logger.verbose	Returns Verbose logs whose levels are greater than or equal to the value of this configuration item. Valid values: 0 to 99. Default value: 0. The value of 0 indicates that no Verbose logs are returned.

Configuration item	Description
logger.cleaner.enable	Specifies whether to enable log cleanup. Default value: false. Valid values: <ul style="list-style-type: none"> <li><code>true</code>: enables log cleanup.</li> <li><code>false</code>: disables log cleanup.</li> </ul>
fs.oss.endpoint	The endpoint that is used to access the JindoFS service. Example: <code>cn-hangzhou.oss-dls.aliyuncs.com</code> .
fs.oss.accessKeyId	The AccessKey ID that is used to access the JindoFS service.
fs.oss.accessKeySecret	The AccessKey secret that is used to access the JindoFS service.

### 3.1.3. Use MaxCompute to build a data warehouse based on OSS

This topic describes how to use MaxCompute to build a petabyte-grade data warehouse based on Object Storage Service (OSS). MaxCompute can analyze the large amounts of data stored in OSS in a fast and efficient manner, which allows you to explore data value within several minutes and at low cost.

#### Prerequisites

- OSS is activated. One or more buckets are created.
  - For more information about how to activate OSS, see [Activate OSS](#).
  - For more information about how to create a bucket, see [Create buckets](#).
- You have activated MaxCompute and authorized MaxCompute to access OSS.
  - For more information about how to activate MaxCompute, see [Activate MaxCompute and DataWorks](#).
  - You need to authorize the account used for running MaxCompute jobs to access OSS data. After you log on with an Alibaba Cloud account, go to the [authorization page](#) to authorize the account.

#### Context

Internet finance applications need to store large amounts of financial data exchange objects in OSS every day and perform structured analysis of large text files. MaxCompute provides the OSS external table query function, which allows you to use external tables to load large OSS objects into MaxCompute for analysis. This method can improve the efficiency of the entire process.

#### Operation example: Analyze data collected by IoT

1. Upload data collected by IoT to OSS.

For more information about the procedure, see [Upload objects](#). You can use any data set to perform tests and verify the best practice described in this topic. This example prepares CSV data in OSS. The endpoint is `oss-cn-beijing-internal.aliyuncs.com`. The bucket name is `oss-odps-test`. Data is stored in `/demo/vehicle.csv`.

2. Create a MaxCompute project.

For more information about the procedure, see [Create a project](#).

### 3. Create an external table by using MaxCompute.

For more information about the procedure, see the "Create a table" section in [Create tables](#). An example of the statement is as follows:

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_csv_external
(
    vehicleId int,
    recordId int,
    patientId int,
    calls int,
    locationLatitude double,
    locationLongitude double,
    recordTime string,
    direction string
)
STORED BY 'com.aliyun.odps.CsvStorageHandler'
LOCATION 'oss://oss-cn-beijing-internal.aliyuncs.com/oss-odps-test/Demo/';
```

### 4. Query the external table through MaxCompute.

After creating an external table, you can use it as a normal table. For more information, see [Execute SQL statements and export the result data](#).

Assume that the `/demo/vehicle.csv` object contains the following data:

```
1,1,51,1,46.81006,-92.08174,9/14/2014 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2014 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2014 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2014 0:00,S
1,7,53,1,46.81006,-92.08174,9/14/2014 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2014 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2014 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2014 0:00,N
```

Run the following SQL statement:

```
select recordId, patientId, direction from ambulance_data_csv_external where patientId
> 25;
```

The following output is returned:

recordId	patientId	direction
1	51	S
3	48	NE
4	30	W
5	47	S
7	53	N
8	63	SW
10	31	N

For more information about how to use OSS external tables, see [Overview](#).

## 3.2. Open source ecosystem

### 3.2.1. Use self-managed Hadoop to access OSS-HDFS

OSS-HDFS (JindoFS service) is a cloud native data lake storage service. OSS-HDFS is built on unified metadata management capabilities and is fully compatible with Hadoop Distributed File System (HDFS) API operations. The Portable Operating System Interface (POSIX) is supported in OSS-HDFS. This helps OSS-HDFS handle data lake computing scenarios in the big data and AI fields. This topic describes how to use self-managed Hadoop to access the fully-managed OSS-HDFS service.

#### Prerequisites

OSS-HDFS is enabled for a bucket. To apply for a trial, contact [technical support](#).

#### Background information

You can use OSS-HDFS without the need to modify the Hadoop and Spark applications. You can configure OSS-HDFS with ease to access and manage data in a similar manner in which you manage data in HDFS. In addition, you can take advantage of Object Storage Service (OSS) characteristics such as unlimited storage space, elastic scalability, and high security, reliability, and availability.

Cloud-native data lakes are based on OSS-HDFS. You can use OSS-HDFS to analyze exabytes of data, manage hundreds of millions of objects, and obtain terabytes of throughput. OSS-HDFS provides the flat namespace feature and the hierarchical namespace feature to meet your requirements for big data storage. The hierarchical namespace feature allows you to manage objects in a hierarchical directory structure. OSS-HDFS can automatically convert the storage structure between the flat namespace and the hierarchical namespace by managing object metadata in a unified manner. Hadoop users can access objects in OSS-HDFS without the need to copy and convert the format of the objects. This improves job performance and reduces maintenance costs.

For more information about the application scenarios, characteristics, and features of OSS-HDFS, see [Overview of the OSS-HDFS service](#).

#### Step 1: Create a virtual private cloud (VPC) and include an Elastic Compute Service (ECS) instance in the VPC

1. Create a VPC that allows access to OSS-HDFS by using internal endpoints.

- i. Log on to the [VPC console](#).
  - ii. On the **VPC** page, click **Create VPC**.  
When you create a VPC, make sure that the VPC is located in the same region as the bucket for which you want to enable OSS-HDFS. For more information about how to create a VPC, see [Create a VPC](#).
2. Include an ECS instance in the VPC.
    - i. Click the ID of the VPC. On the page that appears, click the **Resources** tab.
    - ii. In the **Basic Cloud Resources Included** section, click the  icon to the right of Elastic Computing Service (ECS).
    - iii. On the **Instances** page, click **Create Instance**.  
When you create an ECS instance, make sure that the instance is located in the same region as the VPC. For more information about how to create an ECS instance, see [Create an instance](#).

## Step 2: Create a Hadoop runtime environment

1. Install the Java environment.
  - i. To the right of the created ECS instance, click **Connect**.  
For more information about how to connect to an ECS instance, see [Connection methods](#).
  - ii. Check the version of the existing Java Development Kit (JDK).

```
java -version
```

- iii. (Optional) If the JDK version is earlier than 1.8.0, uninstall the JDK. If the JDK version is 1.8.0 or later, skip this step.

```
rpm -qa | grep java | xargs rpm -e --nodeps
```

- iv. Install the JDK package.

```
yum install java-1.8.0-openjdk* -y
```

- v. Configure environment variables.

```
vim /etc/profile
```

- vi. Add environment variables.

If the current path of the JDK does not exist, go to the `/usr/lib/jvm/` path to find the `java-1.8.0-openjdk` file.

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/jre/lib/rt.jar
export PATH=$PATH:$JAVA_HOME/bin
```

2. Enable the SSH service.
  - i. Install the Secure Shell (SSH) service.

```
yum install -y openssh-clients openssh-server
```

- ii. Enable the SSH service.

```
systemctl enable sshd && systemctl start sshd
```

- iii. Generate an SSH key and add the key to the trusted list.

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa  
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
chmod 0600 ~/.ssh/authorized_keys
```

### 3. Install Hadoop.

- i. Download the Hadoop installation package.

```
wget https://mirrors.sonic.net/apache/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.g  
z
```

- ii. Decompress the package.

```
tar xzf hadoop-3.3.1.tar.gz
```

- iii. Move the package to a frequently accessed path.

```
mv hadoop-3.3.1 /usr/local/hadoop
```

- iv. Configure environment variables.

- a. Configure the environment variables of Hadoop.

```
vim /etc/profile  
export HADOOP_HOME=/usr/local/hadoop  
export PATH=$HADOOP_HOME/bin:$PATH
```

- b. Update *HADOOP\_HOME* in the configuration file of Hadoop.

```
cd $HADOOP_HOME  
vim etc/hadoop/hadoop-env.sh
```

- c. Replace *\$(JAVA\_HOME)* with the actual path.

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

- v. (Optional) If the directory does not exist, run the following command to apply the environment variables:

```
cd $HADOOP_HOME/etc/hadoop
```

vi. Update the *core-site.xml* and *hdfs-site.xml* configuration files.

- Update the *core-site.xml* configuration file and add attributes.

```
<configuration>
  <!-- Specify the address of the NameNode in HDFS. -->
  <property>
    <name>fs.defaultFS</name>
    <!-- Replace the value with the hostname or localhost. -->
    <value>hdfs://localhost:9000</value>
  </property>
  <!-- Modify the temporary directory of Hadoop to a custom directory. -->
  <property>
    <name>hadoop.tmp.dir</name>
    <!-- Run the following command to grant the admin user permissions to manage the directory: sudo chown -R admin:admin /opt/module/hadoop-3.3.1-->
    <value>/opt/module/hadoop-3.3.1/data/tmp</value>
  </property>
</configuration>
```

- Update the *hdfs-site.xml* configuration file and add attributes.

```
<configuration>
  <!-- Specify the number of duplicates for HDFS. -->
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

vii. Format the file structure.

```
hdfs namenode -format
```

## viii. Start HDFS.

To start HDFS, you must respectively start the NameNode, DataNode, and secondary NameNode.

## a. Start HDFS.

```
cd /usr/local/hadoop/
sbin/start-dfs.sh
```

## b. Query the progress.

```
jps
```

The following figure shows the result.

```
[[hadoop@02497b177777 hadoop]$ sbin/start-dfs.sh
Starting namenodes on [02497b177777]
02497b177777: Warning: Permanently added '02497b177777' (ECDSA) to the list of known hosts.
Starting datanodes
Starting secondary namenodes [02497b378f76]
[[hadoop@02497b177777 hadoop]$ jps
3239 SecondaryNameNode
3402 Jps
3034 DataNode
2908 NameNode
```

After you complete the preceding procedure, the daemon process is created. You can use a browser to access `http://{ip}:9870` to view the interface of HDFS and associated details.

## 4. Test whether Hadoop is installed.

Run the `hadoop version` command. If Hadoop is installed, a result similar to the following figure is returned.

```
[[hadoop@02497b177777 workspace]$ hadoop version
Hadoop 3.3.1
Source code repository https://github.com/apache/hadoop.git -r a3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled by ubuntu on 2021-06-15T05:13Z
Compiled with protoc 3.7.1
From source with checksum 88a4ddb2299aca054416d6b7f81ca55
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.3.1.jar
```

### Step 3: Switch your on-premise HDFS to OSS-HDFS

1. Enable OSS-HDFS for a bucket when you create the bucket.  
For more information, see [Create buckets](#).
2. Grant permissions to Resource Access Control (RAM) roles.

- i. Authorize the service to manage buckets for which HDFS is enabled.

The first time you use OSS-HDFS, you must perform the following steps to authorize a RAM user to manage buckets in the RAM console.

- a. Create a RAM role named `AliyunOSSDlsDefaultRole`.
  - a. Log on to the [RAM console](#).
  - b. In the left-side navigation pane, choose **Identities > Roles**.
  - c. Click **Create Role**. In the Create Role panel, select **Alibaba Cloud Service** for Select Trusted Entity and click **Next**.
  - d. Set Role Type to **Normal Service Role** and RAM Role Name to `AliyunOSSDlsDefaultRole`. Then, select **OSS** for Select Trusted Service.
  - e. Click **OK**. After the role is created, click **Close**.
- b. Create a custom policy named `AliyunOSSDlsRolePolicy`.
  - a. In the left-side navigation pane, choose **Permissions > Policies**.
  - b. On the Policies page, click **Create Policy**.
  - c. On the **Create Policy** page, click the **JSON** tab. On the JSON tab, enter the following policy content and click **Next Step**. Then, set **Name** to `AliyunOSSDlsRolePolicy`.

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "oss:ListObjects",
      "Resource": [
        "acs:oss:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "oss:*",
      "Resource": [
        "acs:oss:*:*:*/.dlsdata",
        "acs:oss:*:*:*/.dlsdata*"
      ]
    }
  ]
}
```

- d. Click **OK**.
- c. Attach the custom policy to the RAM role.
  - a. In the left-side navigation pane, choose **Identities > Roles**.
  - b. Find the RAM role `AliyunOSSDlsDefaultRole` and click **Input and Attach** in the Actions column.
  - c. In the **Add Permissions** panel, set Type to **Custom Policy** and Policy Name to `AliyunOSSDlsRolePolicy`.
  - d. Click **OK**.

- ii. Authorize the RAM role to access the buckets for which OSS-HDFS is enabled.
  - a. Create a RAM user. For more information, see [Create a RAM user](#).
  - b. Create a custom policy and enter the following policy content:

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "oss:*",
      "Resource": [
        "acs:oss:*:*:*/.dlsdata",
        "acs:oss:*:*:*/.dlsdata*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:GetBucketInfo",
        "oss:PostDataLakeStorageFileOperation"
      ],
      "Resource": "*"
    }
  ],
  "Version": "1"
}

```

For more information about how to create a custom policy, see [Create a custom policy](#).

- c. Attach the custom policy to the RAM user. For more information, see [Grant permissions to a RAM user](#).

If you want to use a service role, such as the E-MapReduce (EMR) service role `AliyunEMRDe`  
`faultRole`, to access the buckets for which OSS-HDFS is enabled, refer to the preceding procedure for granting permissions to a RAM user.

### 3. Download the JindoSDK JAR package.

- i. Change to the destination directory to which you want to download the package.

```
cd /usr/lib/
```

- ii. Download the JindoSDK JAR package.

```
wget https://jindodata-binary.oss-cn-shanghai.aliyuncs.com/release/4.1.0/jindosdk-4.1.0.tar.gz
```

- iii. Decompress the JindoSDK JAR package.

```
tar xzf jindosdk-4.1.0.tar.gz
```

### 4. Configure environment variables.

- i. Change to the destination directory.

```
vim /etc/profile
```

- ii. Decompress the package. In this example, the package is decompressed in the `/usr/lib/jindosdk-4.0.0` directory.

```
export JINDOSDK_HOME=/usr/lib/jindosdk-4.1.0
```

- iii. Configure `HADOOP_CLASSPATH`.

```
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:${JINDOSDK_HOME}/lib/*
```

- iv. Run the following command to apply the environment variables:

```
. /etc/profile
```

5. Configure the implementation class of OSS-HDFS and the AccessKey pair of the bucket.

- i. Configure the JindoSDK DLS implementation class in the `core-site.xml` file of Hadoop.

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

- ii. Configure the AccessKey ID and AccessKey secret of the bucket for which OSS-HDFS is enabled in the `core-site.xml` file of Hadoop in advance.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>xxx</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>xxx</value>
  </property>
</configuration>
```

6. Configure the endpoint of OSS-HDFS.

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in OSS. We recommend that you configure the endpoint in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the endpoint, JindoSDK accesses the corresponding OSS-HDFS operation based on the specified endpoint in the access path.

You can also configure the endpoint of OSS-HDFS in other methods. For more information, see the "Appendix: Other methods for configuring the endpoint of OSS-HDFS" section in [Quick start for the OSS-HDFS service](#).

## Step 4: Access the OSS-HDFS service

- Create a directory

Run the following command to create a directory named `dir/` in a bucket named `examplebucket`:

```
hdfs dfs -mkdir oss://examplebucket.cn-hangzhou.oss-dls.aliyuncs.com/dir/
```

- Upload an object

Run the following command to upload a file named `examplefile.txt` to a bucket named `examplebucket`:

```
hdfs dfs -put /root/workspace/examplefile.txt oss://examplebucket.cn-hangzhou.oss-dls.aliyuncs.com/examplefile.txt
```

- Query a directory

Run the following command to query a directory named `dir/` in a bucket named `examplebucket`:

```
hdfs dfs -ls oss://examplebucket.oss-dls.aliyuncs.com/dir/
```

- Query an object

Run the following command to query an object named `examplefile.txt` in a bucket named `examplebucket`:

```
hdfs dfs -ls oss://examplebucket.oss-dls.aliyuncs.com/examplefile.txt
```

- Query the content of an object

Run the following command to query the content of an object named `examplefile.txt` in a bucket named `examplebucket`:



**Notice** After you run the following command, the content of the queried object is displayed on the screen in plain text. If the content is encoded, use Java API of HDFS to decode the object and read the content of the object.

```
hdfs dfs -cat oss://examplebucket.oss-dls.aliyuncs.com/examplefile.txt
```

- Copy an object or a directory

Run the following command to copy the root directory named `subdir1` to a directory named `subdir2` in a bucket named `examplebucket`. In addition, the position of the `subdir1` root directory, the objects in the `subdir1` root directory, and the structure and content of subdirectories in the `subdir1` root directory remain unchanged.

```
hdfs dfs -cp oss://examplebucket.oss-dls.aliyuncs.com/subdir1/ oss://examplebucket.oss-dls.aliyuncs.com/subdir2/subdir1/
```

- Move an object or a directory

Run the following command to move the root directory named `srcdir` and the objects and subdirectories in the root directory to another root directory named `destdir`:

```
hdfs dfs -mv oss://examplebucket.oss-dls.aliyuncs.com/srcdir/ oss://examplebucket.oss-dls.aliyuncs.com/destdir/
```

- Download an object

Run the following command to download an object named `exampleobject.txt` from a bucket named `examplebucket` to a directory named `/tmp` in the root directory of your computer:

```
hdfs dfs -get oss://examplebucket.oss-dls.aliyuncs.com/exampleobject.txt /tmp/
```

- Delete an object or a directory

Run the following command to delete a directory named `destfolder/` and all objects in the directory from a bucket named `examplebucket`:

```
hdfs dfs -rm oss://examplebucket.oss-dls.aliyuncs.com/destfolder/
```

## 3.2.2. Use HDP 2.6-based Hadoop to read and write OSS data

Hortonworks Data Platform (HDP) is a big data platform released by Hortonworks and consists of open source components such as Hadoop, Hive, and HBase. Hadoop 3.1.1 is included in the HDP 3.0.1 and supports OSS. However, earlier versions of HDP do not support OSS. This topic uses HDP 2.6.1.0 as an example to describe how to configure HDP 2.6 to read and write OSS data.

### Prerequisites

An HDP 2.6.1.0 cluster has been set up. To set up an HDP 2.6.1.0 cluster, use one of the following methods:

- Search for references to use Ambari to set up an HDP 2.6.1.0 cluster.
- If Ambari is not available, manually set up an HDP 2.6.1.0 cluster.

### Procedure

1. Download [HDP 2.6.1.0](#) that supports OSS.

This package includes the support for OSS based on Hadoop in [HDP 2.6.1.0](#). Alibaba Cloud plans to continually release support packages for minor versions of HDP 2.

2. Decompress the downloaded package.

```
[root@hdp-master ~]# tar -xvf hadoop-oss-hdp-2.6.1.0-129.tar
hadoop-oss-hdp-2.6.1.0-129/
hadoop-oss-hdp-2.6.1.0-129/aliyun-java-sdk-ram-3.0.0.jar
hadoop-oss-hdp-2.6.1.0-129/aliyun-java-sdk-core-3.4.0.jar
hadoop-oss-hdp-2.6.1.0-129/aliyun-java-sdk-ecs-4.2.0.jar
hadoop-oss-hdp-2.6.1.0-129/aliyun-java-sdk-sts-3.0.0.jar
hadoop-oss-hdp-2.6.1.0-129/jdom-1.1.jar
hadoop-oss-hdp-2.6.1.0-129/aliyun-sdk-oss-3.4.1.jar
hadoop-oss-hdp-2.6.1.0-129/hadoop-aliyun-2.7.3.2.6.1.0-129.jar
```

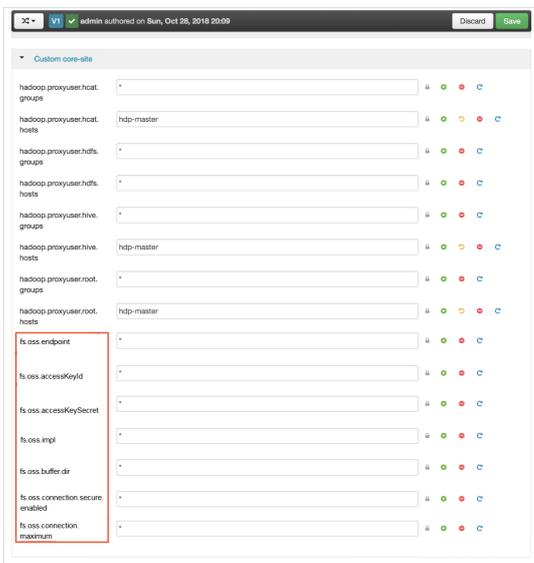
3. Move the `Hadoop-aliyun-2.7.3.2.6.1.0-129.jar` package to the `/${usr}/hdp/current}/hadoop-client/` directory. Move other `jar` packages to the `/${usr}/hdp/current}/hadoop-client/lib/` directory.

Directory structure after the preceding adjustments:

```
[root@hdp-master ~]# ls -lh /usr/hdp/current/hadoop-client/hadoop-aliyun-2.7.3.2.6.1.0-129.jar
-rw-r--r-- 1 root root 64K Oct 28 20:56 /usr/hdp/current/hadoop-client/hadoop-aliyun-2.7.3.2.6.1.0-129.jar
[root@hdp-master ~]# ls -ltrh /usr/hdp/current/hadoop-client/lib
total 27M
.....
drwxr-xr-x 2 root root 4.0K Oct 28 20:10 ranger-hdfs-plugin-impl
drwxr-xr-x 2 root root 4.0K Oct 28 20:10 ranger-yarn-plugin-impl
drwxr-xr-x 2 root root 4.0K Oct 28 20:10 native
-rw-r--r-- 1 root root 114K Oct 28 20:56 aliyun-java-sdk-core-3.4.0.jar
-rw-r--r-- 1 root root 513K Oct 28 20:56 aliyun-sdk-oss-3.4.1.jar
-rw-r--r-- 1 root root 13K Oct 28 20:56 aliyun-java-sdk-sts-3.0.0.jar
-rw-r--r-- 1 root root 211K Oct 28 20:56 aliyun-java-sdk-ram-3.0.0.jar
-rw-r--r-- 1 root root 770K Oct 28 20:56 aliyun-java-sdk-ecs-4.2.0.jar
-rw-r--r-- 1 root root 150K Oct 28 20:56 jdom-1.1.jar
```

**Note** All content in `$$` is environment variables. Modify these environment variables.

4. Perform the preceding operations on all HDP nodes.
5. Use Ambari to add configurations. If your cluster does use Ambari for management, modify `core-site.xml`. This example uses Ambari. The following table describes the configurations you need to add.



Parameter	Configuration method
<b>fs.oss.endpoint</b>	Enter the endpoint used to access the region where the bucket is located. Example: oss-cn-zhangjiakou-internal.aliyuncs.com.
<b>fs.oss.accessKeyId</b>	Enter the AccessKey ID used to access OSS.
<b>fs.oss.accessKeySecret</b>	Enter the AccessKey secret used to access OSS.

Parameter	Configuration method
<code>fs.oss.impl</code>	Enter the class used to implement the OSS file system through Hadoop. Set the value to <code>org.apache.hadoop.fs.aliyun.oss.AliyunOSSFileSystem</code> .
<code>fs.oss.buffer.dir</code>	Enter the temporary file directory. We recommend that you set this parameter to <code>/tmp/oss</code> .
<code>fs.oss.connection.secure.enabled</code>	Specify whether to enable HTTPS. Performance may be affected when HTTPS is enabled. We recommend that you set this parameter to <code>false</code> .
<code>fs.oss.connection.maximum</code>	Enter the maximum number of connections to OSS. We recommend that you set this parameter to <code>2048</code> .

For more information about parameter descriptions, visit [Hadoop-Aliyun module](#).

6. Restart the cluster as prompted by Ambari.
7. Test data reading from and writing to OSS.
  - o Test data reading from OSS

```
hadoop fs -ls oss://{your-bucket-name}/
```

- o Test data writing to OSS

```
hadoop fs -mkdir oss://{your-bucket-name}/hadoop-test
```

If data can be read from and written to OSS, the configurations are successful. Otherwise, check configurations.

8. To run MapReduce jobs, modify the content in the `hdfs://hdp-master:8020/hdp/apps/2.6.1.0-129/mapreduce/mapreduce.tar.gz` package. To run TEZ jobs, modify the content in the `hdfs://hdp-master:8020/hdp/apps/2.6.1.0-129/tez/tez.tar.gz` package. Modify the package based on job types. Add the OSS-compliant package to the preceding package. Run the following commands:

```
[root@hdp-master ~]# sudo su hdfs
[hdfs@hdp-master root]$ cd
[hdfs@hdp-master ~]$ hadoop fs -copyToLocal /hdp/apps/2.6.1.0-129/mapreduce/mapreduce.tar.gz .
[hdfs@hdp-master ~]$ hadoop fs -rm /hdp/apps/2.6.1.0-129/mapreduce/mapreduce.tar.gz
[hdfs@hdp-master ~]$ cp mapreduce.tar.gz mapreduce.tar.gz.bak
[hdfs@hdp-master ~]$ tar zxf mapreduce.tar.gz
[hdfs@hdp-master ~]$ cp /usr/hdp/current/hadoop-client/hadoop-aliyun-2.7.3.2.6.1.0-129.jar hadoop/share/hadoop/tools/lib/
[hdfs@hdp-master ~]$ cp /usr/hdp/current/hadoop-client/lib/aliyun-* hadoop/share/hadoop/tools/lib/
[hdfs@hdp-master ~]$ cp /usr/hdp/current/hadoop-client/lib/jdom-1.1.jar hadoop/share/hadoop/tools/lib/
[hdfs@hdp-master ~]$ tar zcf mapreduce.tar.gz hadoop
[hdfs@hdp-master ~]$ hadoop fs -copyFromLocal mapreduce.tar.gz /hdp/apps/2.6.1.0-129/mapreduce/
```

## Verify the configurations

You can test TeraGen and TeraSort to check whether the configurations take effect.

- Test TeraGen:

```
[hdfs@hdp-master ~]$ hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce
-examples.jar teragen -Dmapred.map.tasks=100 10995116 oss://{bucket-name}/1G-input
18/10/28 21:32:38 INFO client.RMProxy: Connecting to ResourceManager at cdh-master/192.16
8.0.161:8050
18/10/28 21:32:38 INFO client.AHSPProxy: Connecting to Application History server at cdh-m
aster/192.168.0.161:10200
18/10/28 21:32:38 INFO aliyun.oss: [Server]Unable to execute HTTP request: Not Found
[ErrorCode]: NoSuchKey
[RequestId]: 5BD5BA7641FCE369BC1D052C
[HostId]: null
18/10/28 21:32:38 INFO aliyun.oss: [Server]Unable to execute HTTP request: Not Found
[ErrorCode]: NoSuchKey
[RequestId]: 5BD5BA7641FCE369BC1D052F
[HostId]: null
18/10/28 21:32:39 INFO terasort.TeraSort: Generating 10995116 using 100
18/10/28 21:32:39 INFO mapreduce.JobSubmitter: number of splits:100
18/10/28 21:32:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15407289865
31_0005
18/10/28 21:32:39 INFO impl.YarnClientImpl: Submitted application application_15407289865
31_0005
18/10/28 21:32:39 INFO mapreduce.Job: The url to track the job: http://cdh-master:8088/pr
oxy/application_1540728986531_0005/
18/10/28 21:32:39 INFO mapreduce.Job: Running job: job_1540728986531_0005
18/10/28 21:32:49 INFO mapreduce.Job: Job job_1540728986531_0005 running in uber mode : f
alse
18/10/28 21:32:49 INFO mapreduce.Job: map 0% reduce 0%
18/10/28 21:32:55 INFO mapreduce.Job: map 1% reduce 0%
18/10/28 21:32:57 INFO mapreduce.Job: map 2% reduce 0%
18/10/28 21:32:58 INFO mapreduce.Job: map 4% reduce 0%
...
18/10/28 21:34:40 INFO mapreduce.Job: map 99% reduce 0%
18/10/28 21:34:42 INFO mapreduce.Job: map 100% reduce 0%
18/10/28 21:35:15 INFO mapreduce.Job: Job job_1540728986531_0005 completed successfully
18/10/28 21:35:15 INFO mapreduce.Job: Counters: 36
...
```

- Test TeraSort:

```
[hdfs@hdp-master ~]$ hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce
-examples.jar terasort -Dmapred.map.tasks=100 oss://{bucket-name}/1G-input oss://{bucket-
name}/1G-output
18/10/28 21:39:00 INFO terasort.TeraSort: starting
...
18/10/28 21:39:02 INFO mapreduce.JobSubmitter: number of splits:100
18/10/28 21:39:02 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15407289865
31_0006
18/10/28 21:39:02 INFO impl.YarnClientImpl: Submitted application application_15407289865
31_0006
18/10/28 21:39:02 INFO mapreduce.Job: The url to track the job: http://cdh-master:8088/pr
oxy/application_1540728986531_0006/
18/10/28 21:39:02 INFO mapreduce.Job: Running job: job_1540728986531_0006
18/10/28 21:39:09 INFO mapreduce.Job: Job job_1540728986531_0006 running in uber mode : f
alse
18/10/28 21:39:09 INFO mapreduce.Job: map 0% reduce 0%
18/10/28 21:39:17 INFO mapreduce.Job: map 1% reduce 0%
18/10/28 21:39:19 INFO mapreduce.Job: map 2% reduce 0%
18/10/28 21:39:20 INFO mapreduce.Job: map 3% reduce 0%
...
18/10/28 21:42:50 INFO mapreduce.Job: map 100% reduce 75%
18/10/28 21:42:53 INFO mapreduce.Job: map 100% reduce 80%
18/10/28 21:42:56 INFO mapreduce.Job: map 100% reduce 86%
18/10/28 21:42:59 INFO mapreduce.Job: map 100% reduce 92%
18/10/28 21:43:02 INFO mapreduce.Job: map 100% reduce 98%
18/10/28 21:43:05 INFO mapreduce.Job: map 100% reduce 100%
^@18/10/28 21:43:56 INFO mapreduce.Job: Job job_1540728986531_0006 completed successfully
18/10/28 21:43:56 INFO mapreduce.Job: Counters: 54
...
```

If the tests are successful, the configurations take effect.

### 3.2.3. Use JindoSDK with Apache Flink to process data stored in OSS-HDFS

Open source Apache Flink cannot write data to OSS-HDFS (JindoFS service) in streaming mode and cannot write data to storage media by using exactly-once semantics. If you want to write data to OSS-HDFS in streaming mode by using exactly-once semantics in Flink, you must use JindoSDK together with Flink.

#### Prerequisites

Open source Apache Flink 1.10.1 or later is deployed on clusters.

#### Step 1: Deploy JindoSDK

You can run the following command to download the *JAR* file of JindoSDK to the *lib* directory of the root directory on all Flink nodes:

```
jindo-flink-${version}-full.jar
```

The *JAR* file of JindoSDK is contained in the *jindosdk- $\{version\}.tar.gz$*  package. After you decompress the package, the file is stored in the *plugins/flink/* directory. You can access this file in this directory.

## Step 2: Use JindoSDK in Flink jobs

- General configurations

To write data to Object Storage Service (OSS) by using exactly-once semantics, you must perform the following operations:

- i. Enable the checkpoint mechanism of Apache Flink.
  - a. Run the following command to create a `StreamExecutionEnvironment` class:

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
```

- b. Run the following command to enable the checkpoint mechanism:

```
env.enableCheckpointing(<userDefinedCheckpointInterval>, CheckpointingMode.EXACTLY_ONCE);
```

- ii. Use a data source that supports data retransmission, such as Kafka.

- Ease of use

You can use paths that contain the *oss://* prefix to write data to OSS-HDFS without importing additional dependencies. JindoFS can identify paths that contain the *oss://* prefix and allow data to be written to the path.

The following example shows how to write the `DataStream<String>` object `OutputStream` to OSS.

- i. Run the following command to add a sink:

```
String outputPath = "oss://<user-defined-oss-bucket>/<user-defined-oss-dir>"
StreamingFileSink<String> sink = StreamingFileSink.forRowFormat(
    new Path(outputPath),
    new SimpleStringEncoder<String>("UTF-8")
).build();
outputStream.addSink(sink);
```

- ii. Use `env.execute()` to execute Flink jobs.

## Step 3: (Optional) Configure custom parameters

When you submit Flink jobs, you can specify custom parameters to enable or control specific features.

The following example shows how to enable or control a specific feature by configuring `-yD` when you submit Flink jobs in the *yarn-cluster* mode:

```
<flink_home>/bin/flink run -m yarn-cluster -yD key1=value1 -yD key2=value2 ...
```

JindoSDK allows you to enable the entropy injection feature or control the degree of parallelism (DOP) of multipart upload tasks.

- Entropy injection

You can use the entropy injection feature to replace a specific string of the destination path with a random string. This way, written data is distributed to different partitions based on the paths to improve write performance.

To write data to OSS-HDFS, you must complete the following configurations:

```
oss.entropy.key=<user-defined-key>
oss.entropy.length=<user-defined-length>
```

When you write data to an object, the string that is the same as the `<user-defined-key>` string in the write path is replaced with a random string. The length of the random string must be the value of `<user-defined-length>`. The `<user-defined-length>` value must be greater than 0.

- **Multipart upload**

When you write data to OSS-HDFS, the resumable read and write feature can use the efficient multipart upload mechanism to split the file to upload into multiple parts, separately upload the parts, and then combine the parts into a complete object. You can configure the `oss.upload.max.concurrent.uploads` parameter to control the DOP of the data block that you want to upload. If you specify a large value for this parameter, the write performance can be improved. However, more resources are consumed. By default, the value of this parameter is the maximum number of available processors.

## 3.2.4. Use OSS-HDFS as the underlying storage of HBase

HBase is a real-time database that provides high write performance in the Hadoop ecosystem. OSS-HDFS (JindoFS service) is a new storage service released by Alibaba Cloud and is compatible with HDFS interfaces. JindoSDK allows HBase to use OSS-HDFS as the underlying storage and supports the storage of write-ahead logging (WAL) files. This way, the computing and storage resources are separated. OSS-HDFS is more flexible than the local HDFS storage and reduces the O&M cost.

### Step 1: Download and install the JindoSDK JAR package

1. Download the latest version of JindoSDK JAR package. To download the package, visit [GitHub](#).

 **Note** JindoSDK V4.3.0 and later contain Kerberos- and SASL-related dependencies.

2. If Kerberos- and SASL-related dependencies are not contained in your application, you must install the following dependencies on all nodes on which JindoSDK is deployed.

- o **Ubuntu or Debian**

```
sudo apt-get install libkrb5-dev krb5-admin-server krb5-kdc krb5-user libsasl2-dev libbsasl2-modules libsasl2-modules-gssapi-mit
```

- o **Red Hat Enterprise Linux or CentOS**

```
sudo yum install krb5-server krb5-workstation cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-plain
```

- o **macOS**

```
brew install krb5
```

3. Decompress the downloaded installation package.

The following example shows how to decompress the package to the `/usr/lib` path:

```
tar -zxvf jindosdk-4.3.0.tar.gz -C /usr/lib
```

#### 4. Configure `JINDOSDK_HOME` .

```
export JINDOSDK_HOME=/usr/lib/jindosdk-4.3.0
export PATH=$JINDOSDK_HOME/bin:$PATH
```

#### 5. Configure `HADOOP_CLASSPATH` .

 **Notice** Deploy the directory of the installation package and environment variables on all required nodes.

#### 6. Install the decompressed package.

The following example shows how to install the decompressed package to the `HADOOP_CLASSPATH` path.

```
cp ./jindosdk-4.3.0.jar <HADOOP_CLASSPATH>/share/hadoop/hdfs/lib/jindofs-sdk.jar
```

## Step 2: Configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket

### 1. Configure the implementation class of OSS-HDFS in the `core-site.xml` file of HBase.

The following example shows how to configure the implementation class of OSS-HDFS in the file:

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.OSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

### 2. Configure the AccessKey ID and AccessKey secret used to access the bucket for which OSS-HDFS is enabled in the `core-site.xml` file of HBase.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>LTAI5t7h6SgiLSganP2m****</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>KZo149BD9GLPNiDIEmdQ7d****</value>
  </property>
</configuration>
```

## Step 3: Configure the endpoint of OSS-HDFS

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in Object Storage Service (OSS). We recommend that you configure the path that is used to access OSS-HDFS in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the path, JindoSDK accesses the corresponding OSS-HDFS from the endpoint specified in the path.

You can also configure the endpoint of OSS-HDFS by using other methods. In addition, the configuration by using different methods takes effect in a specific order of precedence. For more information, see the "Appendix: Other methods used to configure the endpoint of OSS-HDFS" section in [Get started with OSS-HDFS](#).

## Step 4: Specify a storage path for HBase

You can specify the storage path for HBase and WAL files by changing the value of the `hbase.rootdir` parameter in the `hbase-site` configuration file to the path of objects stored in OSS. The path is in the `oss://bucket.endpoint/hbase-root-dir` format.

 **Notice** To release clusters, you must first disable tables and make sure that all update operations performed on WAL files are synchronized to the storage file named HFile.

## 3.2.5. Use JindoSDK with Hive to process data stored in OSS-HDFS

If you use Hive to build a traditional offline data warehouse that uses HDFS to store data, the data warehouse cannot meet your requirements at a lower cost as the amount of data stored in the warehouse increases. In this case, you can use OSS-HDFS (JindoFS service) as the underlying storage of the Hive data warehouse and use JindoSDK to obtain better read and write performance.

### Step 1: Install JindoSDK on the Hive client or on OSS-HDFS nodes

1. Download the latest version of JindoSDK JAR package. To download the package, visit [GitHub](#).

 **Note** JindoSDK V4.3.0 and later contain Kerberos- and SASL-related dependencies.

2. If Kerberos- and SASL-related dependencies are not contained in your application, you must install the following dependencies on all nodes on which JindoSDK is deployed.

- o Ubuntu or Debian

```
sudo apt-get install libkrb5-dev krb5-admin-server krb5-kdc krb5-user libsasl2-dev libbsasl2-modules libsasl2-modules-gssapi-mit
```

- o Red Hat Enterprise Linux or CentOS

```
sudo yum install krb5-server krb5-workstation cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-plain
```

- o macOS

```
brew install krb5
```

3. Install the downloaded JindoSDK JAR package in the path specified by classpath.

You can run the following command to install this package:

```
cp jindosdk-4.3.0/lib/*.jar $HIVE_HOME/lib/
```

## Step 2: Configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket

1. Configure the implementation class of OSS-HDFS in the *core-site.xml* file of Hive.

The following example shows how to configure the implementation class of OSS-HDFS in the file:

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.OSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

2. Configure the AccessKey ID and AccessKey secret used to access the bucket for which OSS-HDFS is enabled in the *core-site.xml* file of Hive.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>LTAI5t7h6SgiLSganP2m****</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>KZo149BD9GLPNiDIEmdQ7d****</value>
  </property>
</configuration>
```

## Step 3: Configure the endpoint of OSS-HDFS

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in Object Storage Service (OSS). We recommend that you configure the path that is used to access OSS-HDFS in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the path, JindoSDK accesses the corresponding OSS-HDFS from the endpoint specified in the path.

You can also configure the endpoint of OSS-HDFS by using other methods. In addition, the configuration by using different methods takes effect in a specific order of precedence. For more information, see the "Appendix: Other methods used to configure the endpoint of OSS-HDFS" section in [Get started with OSS-HDFS](#).

After you complete the configuration, you must restart Hive for the configuration to take effect.

## Step 4: Use OSS-HDFS to store data

When you create a database or a table, you can use one of the following methods to specify an OSS-HDFS storage path that is used to store the data of the database or table:

- Method 1: Specify the OSS-HDFS storage path in a sample command
  - Specify the OSS-HDFS storage path when you create a database

```
CREATE DATABASE db_on_oss1 LOCATION 'oss://bucket_name.endpoint_name/path/to/db1';
```

- Specify the OSS-HDFS storage path when you create a table

```
CREATE TABLE db2.table_on_oss ... LOCATION 'oss://bucket_name.endpoint_name/path/to/db2/tablepath';
```

- Method 2: Specify the OSS-HDFS storage path in a configuration file

You can set the value of `hive.metastore.warehouse.dir` to the OSS-HDFS storage path in the `hive-site.xml` configuration file of Hive Metastore, and then restart Hive Metastore. By default, databases and tables that you create are stored in the specified OSS-HDFS storage path.

The following example shows how to set `hive.metastore.warehouse.dir` to the OSS-HDFS storage path in the configuration file:

```
<configuration>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>oss://bucket_name.endpoint_name/path/to/warehouse</value>
  </property>
</configuration>
```

## Step 5: Add partitions to an existing table

You can add partitions to an existing table to store the table in smaller units. You can specify query conditions based on partitions. This way, only partitions that meet the specified conditions are scanned and the query performance is improved.

- Syntax

```
ALTER TABLE <table_name> ADD [IF NOT EXISTS] PARTITION <pt_spec> [PARTITION <pt_spec> PARTITION <pt_spec>...] LOCATION 'location';
```

The following table describes the parameters that the previous syntax contains.

Parameter	Required	Description
table_name	Yes	The name of the table to which you want to add partitions.
IF NOT EXISTS	No	Specifies that a partition is added to the table only when no partitions that have the same name as the partition exist. If the IF NOT EXISTS parameter is not specified, and a partition with the same name as the partition to add already exists, the partition fails to be added to the table and an error is returned.

Parameter	Required	Description
pt_spec	Yes	The partitions that you want to add. Specify the partitions in the <code>(partition_col1 = partition_col_value1, partition_col2 = partition_col_value2, ...)</code> format. In this format, <code>partition_col</code> indicates the names of partition key columns, and <code>partition_col_value</code> indicates their values. The names of partition key columns are case-insensitive, but their values are case-sensitive.
location	Yes	The Object Storage Service (OSS) path that is used to store data in a partition.

- Example

The following example shows how to add a partition to a table named `sale_detail` to store the sale records in the China (Hangzhou) region in December 2013 and specify the OSS path that is used to store data in the partition:

```
ALTER TABLE sale_detail ADD IF NOT EXISTS PARTITION (sale_date='201312', region='hangzhou') LOCATION 'oss://examplebucket.cn-hangzhou.oss-dls.aliyuncs.com/path/2013/';
```

## 3.2.6. Use JindoSDK with Impala to query data stored in OSS-HDFS

JindoSDK is a simple and easy-to-use Object Storage Service (OSS) client that is developed for the Hadoop and Spark ecosystems. The client implements a highly-optimized Hadoop file system based on OSS. You can use JindoSDK with Impala to query data stored in OSS-HDFS (JindoFS service) with better query performance than Hadoop OSS clients.

### Prerequisites

Hadoop is installed. For more information about how to install Hadoop, see "Step 2: Create a Hadoop runtime environment" in [Use self-managed Hadoop to access OSS-HDFS](#).

### Step 1: Install JindoSDK on all Impala nodes

1. Download the latest version of JindoSDK JAR package. To download the package, visit [GitHub](#).

 **Note** JindoSDK V4.3.0 and later contain Kerberos- and SASL-related dependencies.

2. If Kerberos- and SASL-related dependencies are not contained in your application, you must install the following dependencies on all nodes on which JindoSDK is deployed.

- o Ubuntu or Debian

```
sudo apt-get install libkrb5-dev krb5-admin-server krb5-kdc krb5-user libsasl2-dev libbsasl2-modules libsasl2-modules-gssapi-mit
```

- o Red Hat Enterprise Linux or CentOS

```
sudo yum install krb5-server krb5-workstation cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-plain
```

- o macOS

```
brew install krb5
```

3. Install the downloaded jindoSDK JAR package in the path specified by classpath.

```
cp jindosdk-4.3.0/lib/*.jar $IMPALA_HOME/lib/
```

## Step 2: Configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket

1. Configure the implementation class of OSS-HDFS in the *core-site.xml* file of Impala.

The following example shows how to configure the implementation class of OSS-HDFS in the file:

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.OSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

2. Configure the AccessKey ID and AccessKey secret used to access the bucket for which OSS-HDFS is enabled in the *core-site.xml* file of Impala.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>LTAI5t7h6SgiLSganP2m****</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>KZo149BD9GLPNiDIEmdQ7d****</value>
  </property>
</configuration>
```

## Step 3: Configure the endpoint of OSS-HDFS

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in Object Storage Service (OSS). We recommend that you configure the path that is used to access OSS-HDFS in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the path, jindoSDK accesses the corresponding OSS-HDFS from the endpoint specified in the path.

You can also configure the endpoint of OSS-HDFS by using other methods. In addition, the configuration by using different methods takes effect in a specific order of precedence. For more information, see the "Appendix: Other methods used to configure the endpoint of OSS-HDFS" section in [Get started with OSS-HDFS](#).

## Step 4: Use Impala to access OSS

### 1. Create a table.

```
CREATE EXTERNAL TABLE customer_demographics (
  `cd_demo_sk` INT,
  `cd_gender` STRING,
  `cd_marital_status` STRING,
  `cd_education_status` STRING,
  `cd_purchase_estimate` INT,
  `cd_credit_rating` STRING,
  `cd_dep_count` INT,
  `cd_dep_employed_count` INT,
  `cd_dep_college_count` INT)
STORED AS PARQUET
LOCATION 'oss://bucket.endpoint/dir';
```

### 2. Query data in the table.

```
select * from customer_demographics;
```

## Step 5: (Optional) Optimize the performance of JindoSDK

### 3.2.7. Use JindoSDK with Spark to process data stored in OSS-HDFS

JindoSDK is a simple and easy-to-use Object Storage Service (OSS) client that is developed for the Hadoop and Spark ecosystems. The client implements a highly-optimized Hadoop file system based on OSS. You can use JindoSDK with Spark to query data stored in OSS-HDFS (JindoFS service) with better query performance than Hadoop OSS clients.

#### Prerequisites

Hadoop is installed. For more information about how to install Hadoop, see "Step 2: Create a Hadoop runtime environment" in [Use self-managed Hadoop to access OSS-HDFS](#).

#### Step 1: Install JindoSDK on all Spark nodes

1. Download the latest version of JindoSDK JAR package. To download the package, visit [GitHub](#).

 **Note** JindoSDK V4.3.0 and later contain Kerberos- and SASL-related dependencies.

2. If Kerberos- and SASL-related dependencies are not contained in your application, you must install the following dependencies on all nodes on which JindoSDK is deployed.

- o Ubuntu or Debian

```
sudo apt-get install libkrb5-dev krb5-admin-server krb5-kdc krb5-user libsasl2-dev li
bsasl2-modules libsasl2-modules-gssapi-mit
```

- o Red Hat Enterprise Linux or CentOS

```
sudo yum install krb5-server krb5-workstation cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-plain
```

- o macOS

```
brew install krb5
```

3. Install the downloaded jindoSDK JAR package in the path specified by classpath.

```
cp jindosdk-4.3.0/lib/*.jar $SPARK_HOME/jars/
```

## Step 2: Configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket

- Configure the implementation class in the *core-site.xml* file
  - i. Configure the implementation class of OSS-HDFS in the *core-site.xml* configuration file of Spark.

The following example shows how to configure the implementation class of OSS-HDFS in the configuration file:

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.OSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

- ii. Configure the AccessKey ID and AccessKey secret used to access the bucket for which OSS-HDFS is enabled in the *core-site.xml* configuration file of Spark.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>LTAI5t7h6SgiLSganP2m****</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>KZo149BD9GLPNiDIEmdQ7d****</value>
  </property>
</configuration>
```

- Configure the implementation class and AccessKey pair when you submit Spark jobs

The following code provides an example on how to configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket when you submit Spark jobs:

```
spark-submit --conf spark.hadoop.fs.AbstractFileSystem.oss.impl=com.aliyun.jindodata.oss.OSS --conf spark.hadoop.fs.oss.impl=com.aliyun.jindodata.oss.JindoOssFileSystem --conf spark.hadoop.fs.oss.accessKeyId=LTAI5t7h6SgiLSganP2m**** --conf spark.hadoop.fs.oss.accessKeySecret=KZo149BD9GLPNiDIEmdQ7d****
```

## Step 3: Configure the endpoint of OSS-HDFS

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in Object Storage Service (OSS). We recommend that you configure the path that is used to access OSS-HDFS in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the path, JindoSDK accesses the corresponding OSS-HDFS from the endpoint specified in the path.

You can also configure the endpoint of OSS-HDFS by using other methods. In addition, the configuration by using different methods takes effect in a specific order of precedence. For more information, see the "Appendix: Other methods used to configure the endpoint of OSS-HDFS" section in [Get started with OSS-HDFS](#).

## Step 4: Use Spark to access OSS

1. Create a table.

```
create table test_oss (c1 string) location "oss://examplebucket.cn-hangzhou.oss-dls.aliyuncs.com/dir/";
```

2. Insert data into the table.

```
insert into table test_oss values ("testdata");
```

3. Query data in the table.

```
select * from test_oss;
```

## Step 5: (Optional) Optimize the performance of JindoSDK

### 3.2.8. Use JindoSDK with Presto to query data stored in OSS-HDFS

Presto is an open source distributed SQL query engine for running interactive analytic queries. This topic describes how to use JindoSDK with Presto to query data stored in OSS-HDFS (JindoFS service).

#### Step 1: Install JindoSDK on all Presto nodes

1. Download the latest version of JindoSDK JAR package. To download the package, visit [GitHub](#).

 **Note** JindoSDK V4.3.0 and later contain Kerberos- and SASL-related dependencies.

2. If Kerberos- and SASL-related dependencies are not contained in your application, you must install the following dependencies on all nodes on which JindoSDK is deployed.

- o Ubuntu or Debian

```
sudo apt-get install libkrb5-dev krb5-admin-server krb5-kdc krb5-user libsasl2-dev libsasl2-modules libsasl2-modules-gssapi-mit
```

- o Red Hat Enterprise Linux or CentOS

```
sudo yum install krb5-server krb5-workstation cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-plain
```

- o macOS

```
brew install krb5
```

3. Install the downloaded JindoSDK JAR package in the path specified by classpath.

```
cp jindosdk-4.3.0/lib/*.jar $PRESTO_HOME/plugin/hive-hadoop2/
```

## Step 2: Configure the implementation class of OSS-HDFS and the AccessKey pair used to access a bucket

1. Configure the implementation class of JindoSDK for OSS-HDFS in the Hadoop configuration file named *core-site.xml* on all Presto nodes.

The following example shows how to configure the implementation class of JindoSDK for OSS-HDFS in the configuration file:

```
<configuration>
  <property>
    <name>fs.AbstractFileSystem.oss.impl</name>
    <value>com.aliyun.jindodata.oss.OSS</value>
  </property>
  <property>
    <name>fs.oss.impl</name>
    <value>com.aliyun.jindodata.oss.JindoOssFileSystem</value>
  </property>
</configuration>
```

2. Configure the AccessKey ID and AccessKey secret that are used to access the bucket for which OSS-HDFS is enabled in the Hadoop configuration file named *core-site.xml* on all Presto nodes.

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>LTAI5t7h6SgiLSganP2m****</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>KZo149BD9GLPNiDIEmdQ7d****</value>
  </property>
</configuration>
```

## Step 3: Configure the endpoint of OSS-HDFS

You must configure the endpoint of OSS-HDFS when you use this service to access buckets in Object Storage Service (OSS). We recommend that you configure the path that is used to access OSS-HDFS in the `oss://<Bucket>.<Endpoint>/<Object>` format. Example: `oss://examplebucket.cn-shanghai.oss-dls.aliyuncs.com/exampleobject.txt`. After you configure the path, JindoSDK accesses the corresponding OSS-HDFS from the endpoint specified in the path.

You can also configure the endpoint of OSS-HDFS by using other methods. In addition, the configuration by using different methods takes effect in a specific order of precedence. For more information, see the "Appendix: Other methods used to configure the endpoint of OSS-HDFS" section in [Get started with OSS-HDFS](#).

After you complete the configuration, you must restart Presto for the configuration to take effect.

## Step 4: Query data stored in OSS-HDFS

In the following example, HiveCatalog is used. You can use Presto to create a schema for Object Storage Service (OSS) and execute SQL statements to query data stored in OSS-HDFS. You must install and deploy JindoSDK in the Hive service because Presto depends on Hive Metastore. For more information, see [Use JindoSDK with Hive to process data stored in OSS-HDFS](#).

1. Log on to the Presto console.

```
presto --server <presto_server_address>:<presto_server_port> --catalog hive
```

2. Create a schema for OSS.

```
create schema testDB with (location='oss://<Bucket>.<Endpoint>/<schema_dir>');
use testDB;
```

3. Create a table.

```
create table tbl (key int, val int);
```

4. Insert data into the table.

```
insert into tbl values (1,666);
```

5. Query the table.

```
select * from tbl;
```

## 3.2.9. Use Kite SDK with Apache Sqoop to read and write data stored in OSS-HDFS

Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases. Apache Sqoop cannot directly read or write data stored in OSS-HDFS. You can use a third-party Kite SDK to convert URIs that are used by Apache Sqoop to access databases and therefore access data stored in OSS-HDFS.

### Prerequisites

- Apache Sqoop V1.4.7 or later is deployed on clusters. To download Apache Sqoop, visit [Apache Sqoop](#).
- You can access OSS-HDFS in the Hadoop environment on which Apache Sqoop depends. For more information, see [Get started with OSS-HDFS](#).

### Procedure

1. Install the JindoSDK JAR package.
  - i. Download the following JindoSDK JAR package: [kite-data-oss-3.4.0.jar](#).

- ii. Install the downloaded JindoSDK JAR package in the path specified by classpath.

```
cp ./kite-data-oss-3.4.0.jar <SQOOP_HOME>/lib/kite-data-oss-3.4.0.jar
```

2. Grant the read and write permissions on the package to a specific user or user group.

```
sudo chmod 755 kite-data-oss-3.4.0.jar
```

3. Import data from Object Storage Service (OSS) to the MySQL database.

```
sqoop import --connect <dburi>/<dbname> --username <username> --password <password> --table <tablename> --target-dir <oss-dir> --temporary-rootdir <oss-tmpdir> --check-column <col> --incremental <mode> --last-value <value> -as <format> -m <count>
```

The following table describes the parameters that you can configure when you import data from OSS to the MySQL database.

Parameter	Required	Description
dburi	Yes	The URI that is used to access the database. Example: <code>jdbc:mysql://192.168.xxx.xxx:3306/</code> .
dbname	Yes	The name of the database.
username	Yes	The username that is used to log on to the database.
password	Yes	The password that is used to log on to the database.
tablename	Yes	The name of the MySQL table.
oss-dir	Yes	The OSS-HDFS path from which you want to read data or to which you want to write data. Example: <code>oss://examplebucket.cn-hangzhou.oss-dls.aliyuncs.com/dir/</code> .
oss-tmpdir	No	The OSS directory to which data is temporarily written. You must specify this parameter if the mode parameter is set to append.  If you set the mode parameter to append, Apache Sqoop first imports data to a temporary directory, and then renames and stores the files in the destination directory. If the destination directory exists in OSS-HDFS, Apache Sqoop does not import data into the directory or overwrite data in the directory.
col	No	The column that is used to check incremental import.
mode	No	The incremental import mode. Valid values: append and lastmodified.  <ul style="list-style-type: none"> <li>○ append: Data is incrementally imported based on incremental columns.</li> <li>○ lastmodified: Data is incrementally imported based on time columns.</li> </ul>

---

Parameter	Required	Description
-----------	----------	-------------

value	No	The maximum value of the column used to check for the previous incremental import task.
format	No	The format in which you want to store an object. Default value: <code>textfile</code> . Valid values: <i>avrodatafile</i> , <i>sequencefile</i> , <i>textfile</i> , and <i>parquetfile</i> .
count	No	The number of MapReduce tasks.

# 4. Content distributing and data processing

## 4.1. Create HLS streams based on OSS

Object Storage Service (OSS) allows you to use Real-Time Messaging Protocol (RTMP) to ingest video and audio streams to OSS buckets and store the streams in the HTTP Live Streaming (HLS) format. OSS also provides various authentication and authorization methods to achieve fine-grained access control on audio and video data stored in buckets.

### Prerequisites

A bucket is created. For more information, see [Create buckets](#).

### Basic operations

OSS allows you to use RTMP to upload H.264-encoded video streams and Advanced Audio Coding (AAC)-encoded audio streams to OSS buckets. You can obtain uploaded audio and video data by accessing the PlayURL of a LiveChannel.

- Upload audio and video data

- i. Call the [PutLiveChannel](#) operation to create a LiveChannel.

After the LiveChannel is created, a PublishURL and a PlayURL are returned. The PublishURL is used to ingest streams by using RTMP, and the PlayURL is used to obtain audio and video data.

- ii. Use PublishURL to ingest video and audio streams

OSS stores uploaded audio and video data as HLS streams that include an M3U8 index object and multiple video objects in the TS format. For more information, see [RTMP-based stream ingest](#).

- Obtain audio and video data

To obtain audio and video data that is uploaded by using a LiveChannel, you can access the PlayURL of the LiveChannel by using a browser to access the M3U8 index object.

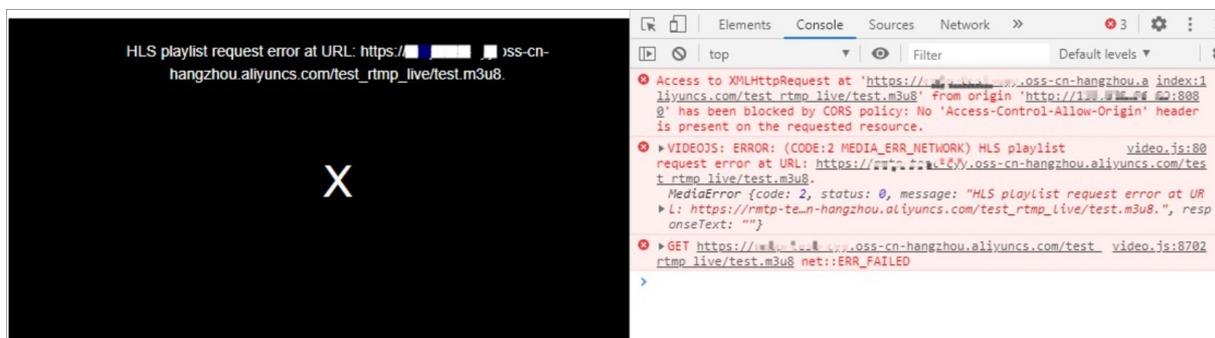
You can use Android and iOS mobile devices and some browsers on PCs, such as Microsoft Edge and Safari, to access the M3U8 index object to play video objects. If you use other browsers such as Google Chrome, you must embed JavaScript scripts such as [Video.js](#) in the browser to play video objects.

If you upload audio and video data to a public read/write bucket, all users can read and write your data in the bucket, which may lead to data leaks and incur additional traffic fees. By default, the access control list (ACL) of a bucket is private, and the bucket denies all cross-origin requests. We recommend that you use one or more of the methods described in the following sections to protect data security based on your actual scenario.

### CORS

If you access an audio or video object embedded in a third-party website rather than directly access the audio and video object stored in OSS by using a browser, the audio or video object may fail to be played because cross-origin requests are denied by the bucket that stores the audio or video object. The request is denied because the web server of the third-party website and the bucket that stores the audio or video object are in different origins, which does not meet the conditions of the same-origin policy.

For example, the address of a web server is `http://192.168.xx.xx:8080`. You use a browser to access the address and then access a website in which a video object stored in an OSS bucket is embedded. In this case, the browser sends a request to OSS to access the video object stored in the bucket. However, the browser identifies that the endpoint of the bucket and the address of the web server, which is `http://192.168.xx.xx:8080`, are in different origins. Then, the browser sends a request to confirm whether the bucket allows cross-origin requests. By default, cross-origin resource sharing (CORS) is disabled for OSS buckets, and all cross-origin requests are denied. Therefore, the video object embedded in the website cannot be played. The following figure shows how the cross-origin request is denied.



You can enable CORS for the bucket that stores audio or video objects to allow users to play the objects on third-party websites.

1. Log on to the [OSS console](#).
2. Click Buckets. On the Buckets page, click the name of the bucket for which you want to enable CORS.
3. Choose **Access Control > Cross-Origin Resource Sharing (CORS)**. In the **Cross-Origin Resource Sharing (CORS)** section, click **Configure**.
4. Click **Create Rule**.
5. In the **Create Rule** dialog box, configure parameters.

In this example, set **Sources** to `http://192.168.xx.xx:8080`. For the configurations of other parameters, see [Configure CORS](#).

- o If the source is a specific domain name, enter the complete domain name. For example, enter `www.example.com` and do not shorten the domain name to `example.com`.
- o If the source is a complete IP address, enter the complete IP address that includes the protocol type and port number. For example, enter `http://xx.xx.xx.xx:80` and do not shorten the IP address to `xx.xx.xx.xx`.

A browser takes tens of seconds to a few minutes to cache the CORS configurations. To allow the CORS configurations to take effect immediately, you can clear your browser cache and then refresh the page.

## Hotlink protection

CORS can prevent your audio or video resources from being embedded in third-party websites. However, CORS configurations cannot prevent your audio or video resources stored in OSS buckets from being directly accessed. In this case, you can configure hotlink protection and specify a Referer whitelist for your buckets to prevent your audio or video resources from being accessed by unauthorized users.

By default, requests in which the Referer field is empty are allowed for OSS buckets. You can use a browser to access the PlayURL of a LiveChannel to view a video object stored in a bucket. To prevent your audio or video resources from being accessed by unauthorized users, you can configure hotlink protection for the bucket to deny requests in which the Referer field is empty and add trusted domain names or IP addresses to the Referer whitelist. In this case, requests from third-party domain names or IP addresses that are not included in the Referer whitelist are denied and a 403 Forbidden error is returned.

- 1.
2. Click **Buckets**. On the **Buckets** page, click the name of the bucket for which you want to configure hotlink protection.
3. Choose **Access Control > Hotlink Protection**.
4. In the **Hotlink Protection** section, click **Configure**.
  - o In the **Referer Whitelist** field, enter domain names or IP addresses. Example: *\*.aliyun.com*.  
You can configure different Referer fields based on your actual scenario. For more information about hotlink protection, see [Configure hotlink protection for a bucket](#).
  - o Turn off **Allow Empty Referer** to deny requests in which the Referer field is empty.

#### Note

- If you turn off **Allow Empty Referer** and configure a Referer whitelist, only requests that include Referer fields specified in the whitelist can access your resources in the bucket.
- If you turn off **Allow Empty Referer** and do not configure a Referer whitelist, the hotlink protection configuration does not take effect. All requests can access your resources.

5. Click **Save**.

## Signature mechanism for private buckets

For data security, the ACL of a bucket is private by default. Therefore, when you send a request to a private bucket to read data from or write data to the bucket, you must add a signature in the request to declare your operation permissions to OSS.

When you want to ingest streams to a private bucket, you must sign the PublishURL to upload audio or video objects to the bucket. For more information, see [RTMP ingest URLs and signatures](#).

The following code provides an example on how to use OSS SDK for Python to obtain the signed PublishURL to ingest streams to a bucket:

```

import os
import oss2
access_key_id = "your_access_key_id"
access_key_secret = "your_access_key_secret"
bucket_name = "your_bucket_name"
endpoint = "your_endpoint"
# Create a bucket.
bucket = oss2.Bucket(oss2.Auth(access_key_id, access_key_secret), endpoint, bucket_name)
# Create and configure a LiveChannel.
# The index object includes three TS objects, and each of the TS object has a duration of 5
seconds. We recommend that you set the value to 5 seconds in this example. The actual durat
ion depends on the key frame of the video object.
channel_name = "your_channel_name"
playlist_name = "your_playlist_name.m3u8"
frag_count_config = 3
frag_duration_config = 5
create_result = bucket.create_live_channel(
    channel_name,
    oss2.models.LiveChannelInfo(
        status = 'enabled',
        description = 'your description here',
        target = oss2.models.LiveChannelInfoTarget(
            playlist_name = playlist_name,
            frag_count = frag_count_config,
            frag_duration = frag_duration_config)))
# Obtain the signed URL used to ingest streams by using RTMP.
# The value of expires in the example indicates the length of time in seconds before the ex
piration of the stream to ingest.
# After you obtain signed_url, you can use data ingestion tools to ingest streams to OSS. O
SS checks the value of expires only when a stream is connected to OSS. A stream that is con
nected to OSS is not suspended even if the duration of the stream exceeds the value specifi
ed by expires.
signed_rtmp_url = bucket.sign_rtmp_url(channel_name, playlist_name, expires=3600)
print(signed_rtmp_url)

```

When you access an object in a private bucket by using the object URL, a signature must be added to the URL. When you access an HLS stream, a request is sent first to dynamically access the M3U8 index object, and then multiple requests are sent to download the latest TS objects based on the content of the index object. A signature must be added to the URL in each request.

```

#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:4
#EXT-X-TARGETDURATION:6
#EXTINF:6.006,
1597993856193.ts?Expires=1598158090&OSSAccessKeyId=LTAI4G6Fg...&Signature=Zrp47sBZsSXsDQOD4...
#EXTINF:6.006,
1597993917986.ts?Expires=1598158090&OSSAccessKeyId=LTAI4G6Fg...&Signature=zy8rjLPa%2FInH1Tco4...
#EXTINF:5.995,
1597993980419.ts?Expires=1598158090&OSSAccessKeyId=LTAI4G6Fg...&Signature=i7qHqynVoKoxceswf...

```

To simplify the signature process, OSS provides a dynamic signature method. You can add the `x-oss-process=hls/sign` header to the URL in the request that is sent to access the M3U8 index object. OSS signs all URLs used to play the TS objects in the returned playlist in the same way as that is specified by the header.

The following code provides an example on how to use OSS SDK for Python to dynamically add signatures to requests when you access audio or video objects:

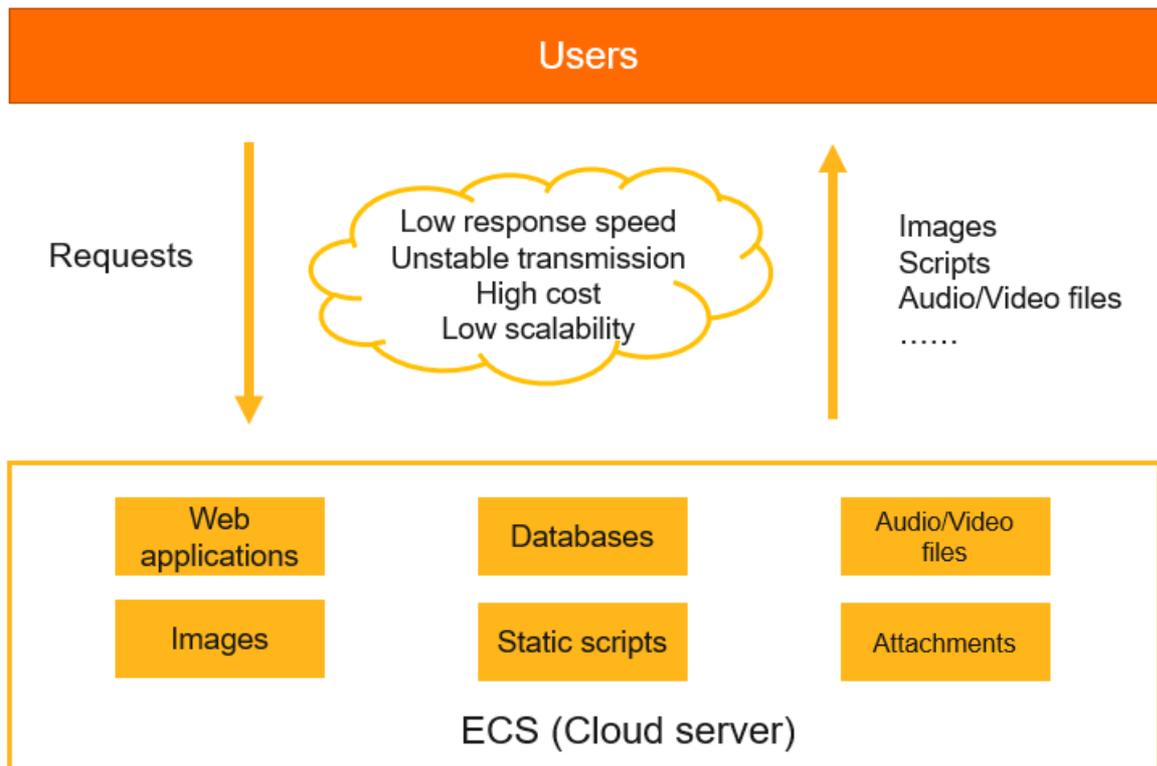
```
# Obtain the signed URL used to view the stream.
your_object_name = "test_rtmp_live/test.m3u8"
style = "hls/sign"
# By default, OSS identifies the forward slashes (/) in the full path of the object as escape
characters when the signed URL is generated. Therefore, you cannot directly use the signed
URL.
# Set the slash_safe parameter to True so that OSS does not identify the forward slashes (/
) in the full path of the object as escape characters. Then, you can directly use the gener
ated signed URL.
signed_download_url = bucket.sign_url('GET', your_object_name, 3600, params={'x-oss-process
': style}, slash_safe=True)
print(signed_download_url)
```

## 4.2. Use CDN to accelerate access to OSS

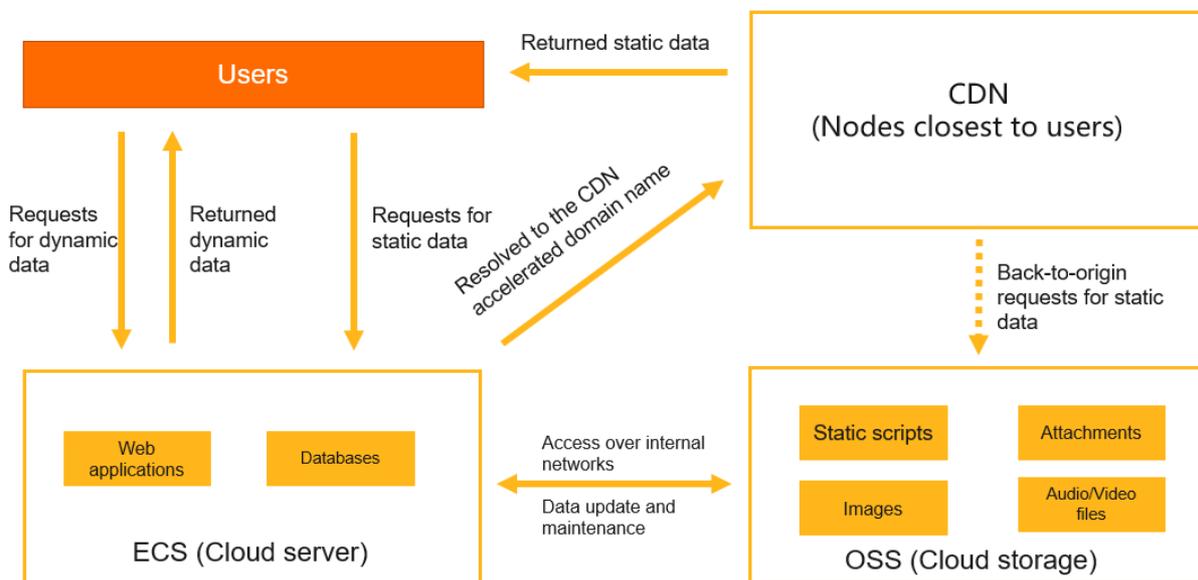
When you access Object Storage Service (OSS) resources, the access speed depends on the region in which the buckets are located and is limited by the downstream bandwidth of OSS. Alibaba Cloud Content Delivery Network (CDN) provides higher bandwidth. If you use CDN to access OSS resources, CDN caches the resources on CDN nodes closest to your region and distributes the resources to you from the nodes. This way, you can access OSS resources more quickly at lower costs. This topic describes how to use CDN to accelerate access to OSS.

### Background information

In traditional website architectures, dynamic and static resources are not separated. Therefore, the performance of a traditional website is bottlenecked when access to the website grows. The following figure shows a traditional website architecture.



You can use an architecture in which dynamic and static resources are separated to resolve the performance problem when the website is accessed by a large number of users. The following figure shows an architecture in which dynamic and static resources are separated.



This architecture separates and distributes resources in the following methods:

- Stores dynamic resources such as web applications and databases on Elastic Compute Service (ECS) instances.
- Stores static resources such as images, video and audio files, and static scripts in OSS buckets.
- Uses OSS buckets as the origins of CDN and distributes objects cached on the node closest to the region in which the user is located to accelerate data access.

This architecture has the following benefits:

- The workload for web servers is reduced.

OSS resources are cached on and distributed from CDN nodes closest to the regions in which users are located. This way, data access is accelerated because the transmission distance is minimized.

- The storage of a large amount of data is supported.

The capacity of OSS buckets can be elastically expanded. You do not need to upgrade your storage architecture.

- Storage fees and traffic fees are reduced.

In this architecture, you are charged the storage fees for OSS buckets, downstream traffic fees for CDN, and a small amount of back-to-origin traffic fees. The storage fees for OSS buckets are 50% cheaper than those for the same capacity of cloud disks. The unit price of CDN traffic is only about 30% to 40% of the unit price of OSS outbound traffic over the Internet.

## Prerequisites

- An OSS bucket is created. Resources are uploaded to the bucket. For more information, see [Upload objects](#).
- Alibaba Cloud CDN is activated. For more information, see [Activate Alibaba Cloud CDN](#).

## Procedure

In the following steps, `example.com` is used as the domain name and `oss.example.com` is used as the accelerated domain name. You can specify an actual domain name as the accelerated domain name, such as a primary domain, second-level domain, or wildcard domain name.

1. Add a domain name.
  - i. Log on to the [CDN console](#). Click **Domain Names**.
  - ii. Click **Add Domain Name**. On the page that appears, configure the following parameters:
    - **Domain Name to Accelerate**: Enter the domain name that you want to specify as the accelerated domain name. In this example, enter `oss.example.com`.
    - **Business Type**: Select **Image and Small File**.
    - **Acceleration Region**: Select **Mainland China Only**.
    - **Origin Information**: Select **Add Origin Server**. On the dialog box that appears, select **OSS Domain** for Type, and then select the domain name of the bucket for which you want to accelerate access from the drop-down list. Keep the default settings of other parameters. Click **OK**.
  - iii. Click **Next**. Then, click **Return**.
  - iv. Wait until the status of the domain name becomes **Running**. Copy the CNAME of the domain name, which is `oss.example.com.w.kunluncan.com` in this example.
2. Resolve the domain name.
  - i. Log on to the [Domains console](#). Click **Resolve** in the Actions column that corresponds to the domain name `example.com`.

- ii. On the **Add Record** page, configure the following parameters:
  - **Record Type**: Select **CNAME** from the drop-down list.
  - **Host**: Enter **oss**.
  - **Value**: Enter the copied CNAME **oss.example.com.w.kunluncan.com**.
  - **Other parameters**: Keep the default settings.
- iii. Click **OK**. Wait a few minutes, and then run the ping command to check whether the accelerated domain name takes effect. If a similar result showed in the following figure is returned, the accelerated domain name takes effect.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\>ping

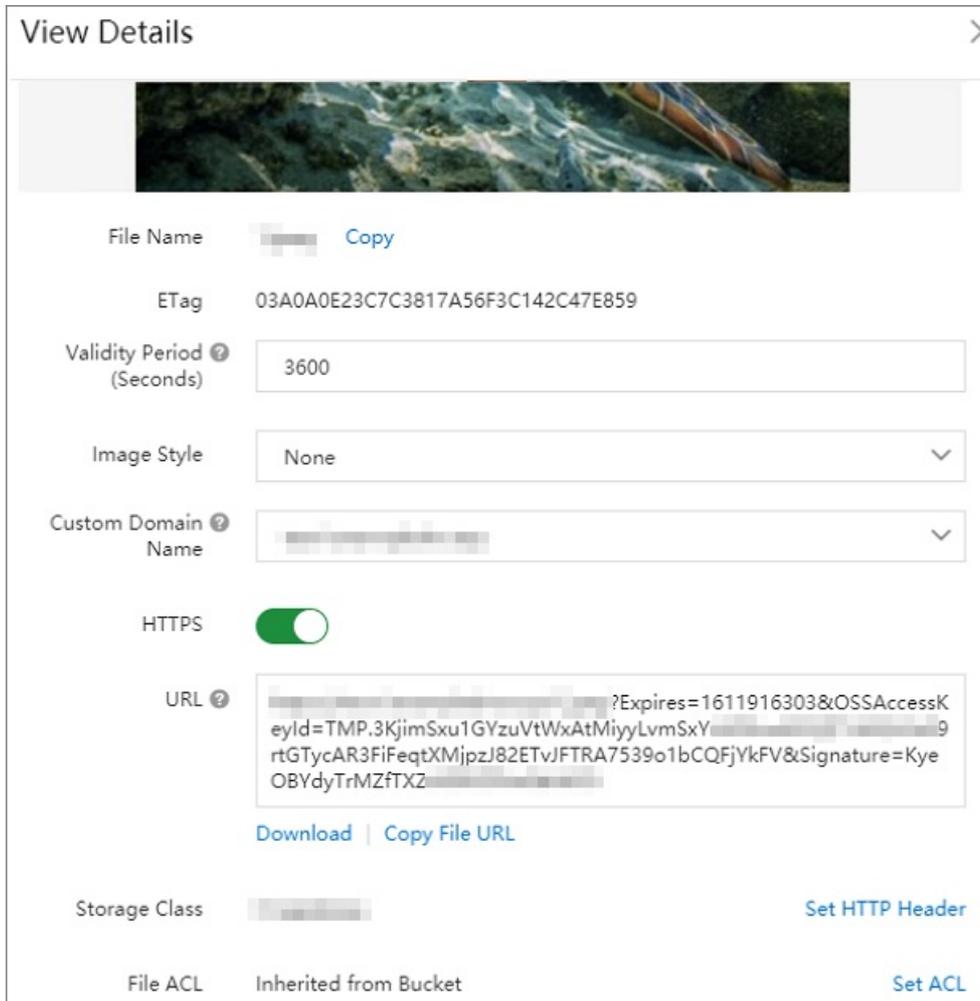
Pinging w.kunlungn.com [1 42] with 32 bytes of data:
Reply from 1 42: bytes=32 time=2ms TTL=106

Ping statistics for 1 42:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

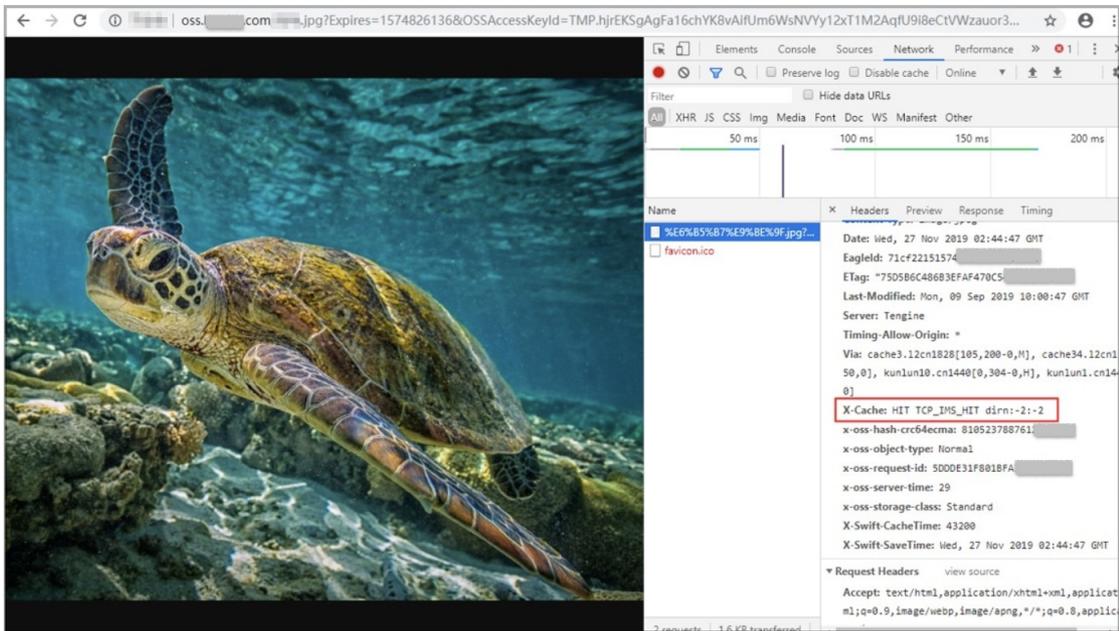
C:\Users\>
```

3. Enable auto CDN cache update
  - i. Log on to the **OSS console**. In the left-side navigation pane, click **Bucket s**. On the **Buckets** page, click the name of the bucket for which you want to accelerate access.
  - ii. Choose **Transmission > Domain Names**.
  - iii. Turn on **Auto CDN Cache Update** for the accelerated domain name that you add.
4. View the URL of an object.
  - i. Log on to the **OSS console**. In the left-side navigation pane, click **Bucket s**. On the **Buckets** page, click the name of the bucket in which the object you want to view is stored.
  - ii. Click **Files**. On the page that appears, click **View Details** in the **Actions** column that corresponds to the object that you want to view.

- iii. In the **View Details** panel, select the accelerated domain name from the **Custom Domain Name** drop-down list. In this example, select **oss.example.com**. As shown in the following figure, the URL of the object starts with the accelerated domain name.



- iv. Access the object by using the URL and use the developer tools of your browser to view the details. The following figure shows that the accelerated domain name takes effect and the object is cached on CDN.



5. Remove the validity period of the URL.
  - i. In the **View Details** panel, click **Set ACL**.
  - ii. Select **Public Read**. Click **OK**.
6. (Optional) Configure HTTPS certificates used to access the object.
  - i. In the **View Details** panel of the object, turn on **HTTPS**.
  - ii. Log on to the **CDN console**. Click **Domain Names**. On the page that appears, click the accelerated domain name that you add.
  - iii. In the left-side navigation pane, click **HTTPS Settings**. In the **SSL Certificate** section, click **Modify**.
  - iv. After you complete the settings, you can access the object by using HTTPS. For more information, see **Configure HTTPS certificates**.

## Purchase resource plans

To further reduce the cost, click **OSS Resource Plan** and **CDN Resource Plan** to purchase resource plans.

# 5. Data backup and recovery

## 5.1. Back up buckets

Alibaba Cloud offers multiple backup methods for data in OSS to suit different scenarios. This topic describes methods used to back up OSS data on the cloud.

### Back up data by using scheduled backup

You can create a backup plan in Hybrid Backup Recovery (HBR) to back up your OSS data. This allows you to recover your data when the data is lost due to an unintended modification or deletion. You can also use HBR to store historical OSS data for an extended period of time at low cost. For more information, see [Configure scheduled backup](#).

### Back up data by using CRR

Cross-region replication (CRR) enables the automatic and asynchronous (near real-time) replication of objects across buckets in different OSS regions. Operations such as the creation, overwriting, and deletion of objects can be synchronized from a source bucket to a destination bucket. For more information, see [Configure CRR](#).

### Back up data by using Data Online Migration

Alibaba Cloud Data Online Migration is used as a data channel between different storage services. You can migrate data from third-party data stores to OSS or between OSS buckets by using Data Online Migration. For more information, see [Background information](#).

### Back up data by using the ossimport tool

ossimport is a tool used to migrate data to OSS. You can deploy ossimport on the local server or ECS instance in the cloud to migrate data stored locally or in other cloud storage systems to OSS. For more information, see [Architectures and configurations](#).

# 6. Cost management

## 6.1. Configure lifecycle rules to manage object versions

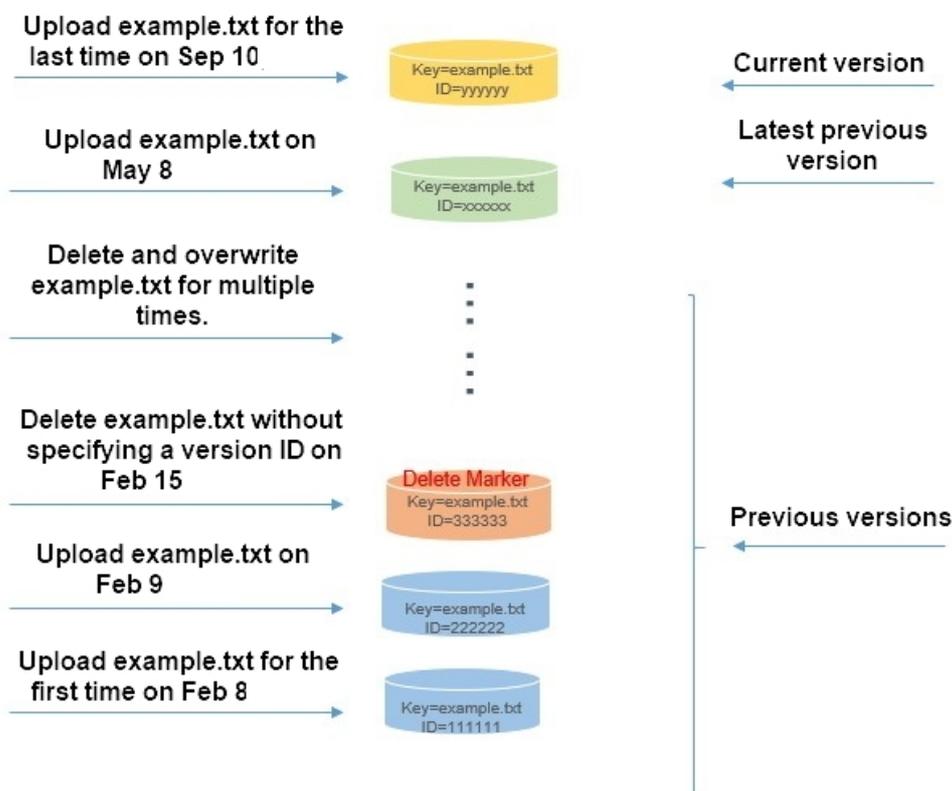
When versioning is enabled for a bucket, data that is overwritten or deleted in the bucket is saved as a previous version. To reduce storage costs and improve bucket performance, you can configure lifecycle rules to delete expired delete markers and previous versions that you no longer use.

### Prerequisites

Versioning is enabled for the bucket. For more information, see [Enable versioning](#).

### Scenarios

Assume that a user uploaded an object named `example.txt` to a versioned bucket named `examplebucket` on February 8. Within the same year, the user overwrote `example.txt` and deleted the object without specifying a version ID multiple times. Each time when the object is overwritten or deleted, Object Storage Service (OSS) saves the current version of the object as a previous version and specifies a random string as the globally unique ID. In the following figure, simple strings but not actual version IDs are used for ease of viewing.



To meet business requirements, the user wants to manage the versions of `example.txt` to achieve the following goals:

- Retain only the versions that are generated on May 8 and September 10.
- Recover the latest previous version generated on May 8 to the current version.

## Usage notes

When you configure lifecycle rules to manage object versions, take note of the following items:

- Expiration policy for the current version of an object
  - In a versioned bucket, if the expiration policy specified in a lifecycle rule is implemented on the current version of an object, OSS adds a delete marker to the object and stores the current version as a previous version. The delete marker becomes the current version of the object.
  - In a versioning-suspended bucket, if the expiration policy specified in a lifecycle rule is implemented on the current version of an object, OSS adds a delete marker whose version ID is null to the object as the new current version. If the object has an existing version whose version ID is null, the version is overwritten by the delete marker because version IDs are globally unique.

- Expiration policy for a previous version of an object

In a bucket for which versioning is enabled or suspended, if the expiration policy specified in a lifecycle is implemented on a previous version of an object, the previous version is permanently deleted and cannot be recovered.

For more information about lifecycle rules, see [Lifecycle rules based on the last modified time](#).

## Procedure

1. Retain specified object versions

Example: The current date is September 10. In this case, the user can perform the following steps to configure a lifecycle rule to retain only the versions of objects uploaded on May 8 and September 10.

- i. Log on to the [OSS console](#).
- ii. In the left-side navigation pane, click **Buckets**. On the Buckets page, click `examplebucket`.
- iii. In the left-side navigation pane, choose **Basic Settings > Lifecycle**. In the **Lifecycle** section, click **Configure**.
- iv. On the page that appears, click **Create Rule**. In the Create Rule panel, configure the parameters described in the following table.

### Create Rule

**i** File deletion and storage class conversion are **irreversible**. Set lifecycle rules as needed.

After the settings of a rule are saved, this rule will be immediately applied to all files and parts of which the validity period has ended or those that were last updated before the specified date. Exercise caution when you configure this rule.

---

**Basic Settings**

Status: Enabled Disabled

Applied To: Files with Specified Prefix Whole Bucket

Tagging ?

---

**Current Version**

File Lifecycle: Validity Period (Days) Expiration Date Clean Up Delete Marker ?  
Disabled

---

**Previous Versions**

File Lifecycle: Validity Period (Days) Disabled

Transit to IA Storage Class:  60

Transit to Archive Storage Class:  180

Transit to Cold Archive Storage Class:  200

Delete:  90  
Files are deleted 90 days after they are last modified.

---

**Delete Parts**

Part Lifecycle: Validity Period (Days) Expiration Date Disabled

Delete:  90  
Parts are deleted 90 days after they are generated.

---

OK Cancel

Section	Parameter	Configuration method
	<b>Status</b>	Select <b>Enabled</b> .

Basic Settings Section	Parameter	Configuration method
	<b>Applied To</b>	Select <b>Whole Bucket</b> .
Current Version	<b>File Lifecycle</b>	Select <b>Clean Up Delete Marker</b> .
Previous Versions	<b>File Lifecycle</b>	Select <b>Validity Period (Days)</b> .
	<b>Delete</b>	Set the value to 90. An object expires 90 days after it is stored as a previous version and is deleted the day after it expires.
Delete Parts	<b>Part Lifecycle</b>	Select <b>Validity Period (Days)</b> .
	<b>Delete Parts</b>	Set the value to 90. Parts that are generated in <b>multipart upload</b> tasks expire 90 days after they are generated and are deleted the day after they expire.

v. Click **OK**.

## 2. Recover a specified object version

To recover the latest previous version that is generated for example.txt on May 8 to the current version, perform the following steps:

- i. On the Overview page of examplebucket, click **Files**.
- ii. Find the previous version of example.txt that you want to recover.
- iii. Click **Restore** in the Actions column corresponding to the previous version that you want to recover.

# 7. Migrate data to OSS

## 7.1. Migrate data between OSS

### 7.1.1. Overview

You can migrate data from a bucket to another bucket owned by the same Alibaba Cloud account. You can also migrate data between Object Storage Service (OSS) buckets owned by different Alibaba Cloud accounts.

Scenarios:

- Data migration between buckets owned by the same Alibaba Cloud account. The buckets may be located in different regions. For more information, see [Use CRR or SRR to migrate OSS data owned by the same Alibaba Cloud account](#).
- Data migration between buckets owned by different Alibaba Cloud accounts. The buckets may be located in different regions. For more information, see [Use Data Online Migration to migrate data between OSS buckets](#).

### 7.1.2. Use CRR or SRR to migrate OSS data owned by the same Alibaba Cloud account

You can use cross-region replication (CRR) to migrate data from a bucket located in Region A to a bucket located in Region B. The two buckets must be owned by the same Alibaba Cloud account. To migrate data in a bucket located in Region A to another bucket located in the same region, you can use the same-region replication (SRR) feature.

#### Usage notes

- Data migration starts three to five minutes after the CRR or SRR rule is configured.
- CRR and SRR replicate data asynchronously (near real-time). Based on the amount of data, it may take several minutes to several hours to migrate the data.
- When historical data is migrated, objects in the source bucket may overwrite objects in the destination bucket if these objects have the same names. To avoid data loss, we recommend that you enable versioning for the source and destination buckets. For more information about how to enable versioning for a bucket, see [Configure versioning](#).
- If you do not want to migrate incremental data in the source bucket after the data in the source bucket is migrated, you can disable the CRR or SRR rule. After the CRR or SRR rule is disabled, the migrated data is stored in the destination bucket, and the incremental data in the source bucket is not replicated to the destination bucket.

For more information about CRR, see [CRR](#). For more information about SRR, see [SRR](#).

#### Data migration between buckets in different regions

Assume that you want to migrate data from the source Bucket A in the China (Beijing) region to the destination Bucket B in the China (Hangzhou) region. Perform the following steps:

1. Log on to the [OSS console](#).
2. In the left-side navigation pane, click **Buckets**. On the **Buckets** page, click **Bucket A**.

3. In the left-side navigation pane, choose **Redundancy for Fault Tolerance > Cross-Region Replication**.
4. In the **Cross-Region Replication** section, click **Configure**.
5. Click **Cross-Region Replication**. In the **Cross-Region Replication** panel, configure the parameters described in the following table.

Parameter	Description and Example
Source Region	The region of Bucket A, which is the China (Beijing) region.
Source Bucket	The name of Bucket A.
Destination Region	Select the China (Hangzhou) region.
Destination Bucket	Select Bucket B.
Applied To	Select <b>All Files in Source Bucket</b> .
Operations	Select <b>Add/Change</b> .
Replicate Historical Data	Select <b>Yes</b> .

6. Click **OK**.  
Then, the **Cross-Region Replication** page displays the data migration progress.

## Data migration between buckets in the same region

Assume that you want to migrate data from the source Bucket C in the China (Beijing) region to the destination Bucket D in the same region. Perform the following steps:

1. Log on to the [OSS console](#).
2. In the left-side navigation pane, click **Buckets**. On the **Buckets** page, click Bucket C.
3. In the left-side navigation pane, choose **Redundancy for Fault Tolerance > Same-Region Replication**.
4. In the **Same-Region Replication** section, click **Configure**.
5. Click **Same-Region Replication**.
6. In the **Same-Region Replication** panel, configure the parameters described in the following table.

Parameter	Description and Example
Source Region	The region of Bucket C, which is the China (Beijing) region.
Source Bucket	The name of Bucket C.
Destination Bucket	Select Bucket D.
Applied To	Select <b>All Files in Source Bucket</b> .

Parameter	Description and Example
Operations	Select <b>Add/Change</b> .
Replicate Historical Data	Select <b>Yes</b> .

7. Click **OK**.

Then, the **Same-Region Replication** page displays the data migration progress.

## References

For more information about how to migrate data in buckets owned by different Alibaba Cloud accounts, see [Use Data Online Migration to migrate data between OSS buckets](#).

### 7.1.3. Use Data Online Migration to migrate data between OSS buckets

You can use Alibaba Cloud Data Online Migration to migrate data from Bucket A owned by Alibaba Cloud Account A to Bucket B owned by Alibaba Cloud Account B. The buckets can be located in the same region or in different regions.

#### Prerequisites

- A Resource Access Management (RAM) user is created.

Create RAM User A by using Alibaba Cloud Account A and create RAM User B by using Alibaba Cloud Account B. For more information, see [Create a RAM user](#).

- An AccessKey pair is created.

Create an AccessKey pair for RAM User A and RAM User B respectively and save the AccessKey pair information. For more information, see [Create an AccessKey pair for a RAM user](#).

- The RAM users are granted related permissions.

Grant the `AliyunOSSFullAccess` and `AliyunMGWFullAccess` permissions to RAM User A and RAM User B. For more information, see [Grant permissions to a RAM user](#).

#### Migrate data across Alibaba Cloud accounts and regions

Assume that you need to migrate data in Bucket A owned by Alibaba Cloud Account A in the China (Shanghai) region to Bucket B owned by Alibaba Cloud Account B in the China (Hangzhou) region by using the public endpoints of the buckets. Perform the following steps:

 **Notice** When you migrate data across different Alibaba Cloud accounts and regions, you can use only the public endpoints of the source and destination buckets.

1. Create a source data address.
  - i. Log on to the [Alibaba Cloud Data Transport console](#).
  - ii. In the left-side navigation pane, choose **Data Online Migration > Data Address**, and then click **Create Data Address**.

- iii. In the **Create Data Address** panel, configure parameters. The following table describes the parameters.

Parameter	Description and example
<b>Data Type</b>	Select <b>OSS</b> .
<b>Data Region</b>	The region in which the source bucket is located. Select <b>China (Shanghai)</b> .
<b>Data Name</b>	Enter <i>migrationtask1</i> .
<b>OSS Endpoint</b>	Select <b>https://oss-cn-shanghai.aliyuncs.com</b> . For more information about Object Storage Service (OSS) endpoints, see <a href="#">Regions and endpoints</a> .
<b>AccessKey Id</b>	Enter the AccessKey ID of the RAM User A.
<b>AccessKey Secret</b>	Enter the AccessKey secret of the RAM User A.
<b>OSS Bucket</b>	Select Bucket A.

2. Create a destination data address.

- i. In the left-side navigation pane, choose **Data Online Migration > Data Address**, and then click **Create Data Address**.
- ii. In the **Create Data Address** panel, configure parameters. The following table describes the parameters.

Parameter	Description and example
<b>Data Type</b>	Select <b>OSS</b> .
<b>Data Region</b>	The region in which the destination bucket is located. Select <b>China (Hangzhou)</b> .
<b>Data Name</b>	Enter <i>migrationtask2</i> .
<b>OSS Endpoint</b>	Select <b>https://oss-cn-hangzhou.aliyuncs.com</b> .
<b>AccessKey Id</b>	Enter the AccessKey ID of the RAM User B.
<b>AccessKey Secret</b>	Enter the AccessKey secret of the RAM User B.
<b>OSS Bucket</b>	Select Bucket B.

3. Create a migration job.

- i. In the left-side navigation pane, choose **Data Online Migration > Migration Jobs**. On the page that appears, click **Create Job**.
- ii. In the **Create Job** panel, read the terms of the migration service and select **I understand the above terms and conditions, and apply for opening data migration service**. Then, click **Next**.
- iii. In the **Fee Reminder** dialog box, click **Yes, Go Ahead**.

- iv. In the **Create Job** panel, configure parameters. The following table describes the parameters. Retain the default value for other parameters, and then click **Next**.

Parameter	Description and example
<b>Job Name</b>	Enter <i>Task2</i> .
<b>Source Data Address</b>	Select the created source data address <code>[oss]migrationtask1</code> .
<b>Destination Data Address</b>	Select the created destination data address <code>[oss]migrationtask2</code> .
<b>Migration Type</b>	Select <b>Full</b> .

- v. On the **Performance** tab, navigate to the **Data Prediction** section and configure the **Data Size** and **File Count** parameters.
- vi. On the **Performance** tab, navigate to the **Flow Control** section, specify the **Time Range** and **Max Flow(MB/s)** parameters, and then click **Add**.
- vii. Click **Create**.

## Migrate data across Alibaba Cloud accounts within the same region

Assume that you want to migrate data from Bucket A owned by Alibaba Cloud account A in the China (Shanghai) region to Bucket B owned by Alibaba Cloud account B within the same region by using the internal endpoints of the buckets. Perform the following steps:

 **Note** When you migrate data across different Alibaba Cloud accounts within the same region, we recommend that you use the internal endpoints of the source and destination buckets. If you use the public endpoints of the buckets, you are charged for the large amount of outbound traffic generated over the Internet.

1. Create a source data address.
  - i. Log on to the [Alibaba Cloud Data Transport console](#).
  - ii. In the left-side navigation pane, choose **Data Online Migration > Data Address**, and then click **Create Data Address**.

- iii. In the **Create Data Address** panel, configure parameters. The following table describes the parameters.

Parameter	Description and example
<b>Data Type</b>	Select <b>OSS</b> .
<b>Data Region</b>	The region in which the source bucket is located. Select <b>China (Shanghai)</b> .
<b>Data Name</b>	Enter <i>migrationtask1</i> .
<b>OSS Endpoint</b>	Select <a href="https://oss-cn-shanghai-internal.aliyuncs.com">https://oss-cn-shanghai-internal.aliyuncs.com</a> . For more information about OSS endpoints, see <a href="#">Regions and endpoints</a> .
<b>AccessKey Id</b>	Enter the AccessKey ID of RAM User A.
<b>AccessKey Secret</b>	Enter the AccessKey secret of RAM User A.
<b>OSS Bucket</b>	Select Bucket A.

2. Create a destination data address.

- i. In the left-side navigation pane, choose **Data Online Migration > Data Address**, and then click **Create Data Address**.
- ii. In the **Create Data Address** panel, configure parameters. The following table describes the parameters.

Parameter	Description and example
<b>Data Type</b>	Select <b>OSS</b> .
<b>Data Region</b>	The region in which the destination bucket is located. Select <b>China (Shanghai)</b> .
<b>Data Name</b>	Enter <i>migrationtask2</i> .
<b>OSS Endpoint</b>	Select <a href="https://oss-cn-shanghai-internal.aliyuncs.com">https://oss-cn-shanghai-internal.aliyuncs.com</a> .
<b>AccessKey Id</b>	Enter the AccessKey ID of RAM User B.
<b>AccessKey Secret</b>	Enter the AccessKey secret of RAM User B.
<b>OSS Bucket</b>	Select Bucket B.

3. Create a migration job.

- i. In the left-side navigation pane, choose **Data Online Migration > Migration Jobs**. On the page that appears, click **Create Job**.
- ii. In the **Create Job** panel, read the terms of the migration service and select **I understand the above terms and conditions, and apply for opening data migration service**. Then, click **Next**.
- iii. In the **Fee Reminder** dialog box, click **Yes, Go Ahead**.

- iv. In the **Create Job** panel, configure parameters. The following table describes the parameters. Retain the default value for other parameters, and then click **Next**.

Parameter	Description and example
<b>Job Name</b>	Enter <i>Task2</i> .
<b>Source Data Address</b>	Select the created source data address <code>[oss]migrationtask1</code> .
<b>Destination Data Address</b>	Select the created destination data address <code>[oss]migrationtask2</code> .
<b>Migration Type</b>	Select <b>Full</b> .

- v. On the **Performance** tab, navigate to the **Data Prediction** section and configure the **Data Size** and **File Count** parameters.
- vi. On the **Performance** tab, navigate to the **Flow Control** section, specify the **Time Range** and **Max Flow(MB/s)** parameters, and then click **Add**.
- vii. Click **Create**.

## References

- Migrate specified data

The preceding scenario assumes that you migrated all the data in a bucket. You can also migrate part of the data in a bucket. For example, to migrate objects whose names contain a specific prefix, specify the prefix when you create data addresses.

- Migrate incremental data

If the source data is changed after full migration of the source bucket, you can specify **Migration Interval** and **Migration Times** to perform incremental data migration. This way, the incremental data in the source bucket is migrated from the source data address to the destination data address.

- Select the operation performed on objects that have the same name

For objects that have the same name in the source data address and the destination data address, you can specify **File Overwrite Method** to directly overwrite the destination data or skip the objects. You can also specify **File Overwrite Method** to determine whether to overwrite the destination data or skip the migration of the objects based on the object metadata such as the last modified time, object sizes, and object Content-Type.

For more information about how to meet more requirements when you migrate data between buckets in OSS, see [Migrate data](#).

## Other scenarios for data migration across accounts

- For more information about how to migrate data within an Alibaba Cloud account, see [Use CRR or SRR to migrate OSS data owned by the same Alibaba Cloud account](#).

## Migrate data within an account

# 7.2. Migrate data sources from a third party to OSS

You can use Alibaba Cloud Data Online Migration to migrate data from a third-party storage service to OSS or between OSS buckets.

To use Data Online Migration, you need only to log on to the Data Transport console, specify information about the source and destination buckets, and create a migration job. After you start a migration job, you can perform management tasks for the job in the console. For example, you can view the migration progress and bandwidth throttling of the job. Additionally, you can use the console to generate a migration report to view the list of migrated files and the list of files that failed to be migrated. For more information about how to migrate data for each data source, see [Data Online Migration](#).

## 7.3. Seamlessly migrate data from Amazon S3 to Alibaba Cloud OSS

Object Storage Service (OSS) is compatible with the Amazon Simple Storage Service (Amazon S3) API to allow you seamlessly migrate data from S3 to OSS.

### Usage notes

- Limits

You can create buckets and upload objects by using S3 SDKs because OSS is compatible with the S3 protocol. For more information about the performance metrics and limits of OSS, such as bandwidth and queries per second (QPS), see [Limits](#).

- Configure the client

After data is migrated to OSS, you can still use S3 API operations to access OSS. You need only to configure your S3 client application by performing the following operations:

- i. Obtain the AccessKey ID and AccessKey secret of your Alibaba Cloud account or those of a Resource Access Management (RAM) user. Specify the AccessKey ID and AccessKey secret in your client and SDK that you use.
- ii. Set the endpoint for the client connection to an OSS endpoint. For more information about OSS endpoints, see [Regions and endpoints](#).

### Migration tutorials

You can use [Alibaba Cloud Data Online Migration](#) to migrate data from AWS S3 to Alibaba Cloud OSS. For more information, see [Background information](#).

### Use S3 API operations to access OSS after migration

When you use S3 API operations to access OSS after data is migrated from S3 to OSS, take note of the following items:

- Path style and virtual hosted style

[Virtual hosted style](#) supports access to OSS by adding the bucket name to the host header. For security reasons, OSS supports only virtual hosted style access. Therefore, you must configure your application client after the migration from S3 to OSS. By default, some S3 tools use path style access, which also requires proper configurations. Otherwise, OSS may report errors and prohibit access.

- Definitions of ACLs

Definitions of access control lists (ACLs) in OSS are not the same as those of S3. You can adjust the configuration of ACLs after the migration. The following table describes the differences between OSS and S3.

Level	AWS S3 permission	AWS S3	Alibaba Cloud OSS
Bucket	READ	The permission to list objects in a bucket.	If no object permissions are configured for an object in a bucket, only read operations can be performed on the object.
	WRITE	The permissions to write or overwrite objects in a bucket.	<ul style="list-style-type: none"> <li>◦ If the object you want to write does not exist in the specified bucket, the object is created in the bucket.</li> <li>◦ If the object you want to write exists in the specified bucket and no permissions are configured for the existing object, the existing object can be overwritten.</li> <li>◦ You can use <code>InitiateMultipartUpload</code> to upload objects.</li> </ul>
	READ_ACP	The permission to read the ACL of a bucket.	Only the bucket owner and authorized RAM users have permissions to read the ACL of a bucket.
	WRITE_ACP	The permission to configure the ACL of a bucket.	Only the bucket owner and authorized RAM users have permissions to configure the ACL of a bucket.
Object	READ	The permission to read an object.	An object can be read.
	WRITE	N/A	An object can be overwritten.
	READ_ACP	The permission to read the ACL of an object.	Only the bucket owner and authorized RAM users have permissions to read the ACL of an object.
	WRITE_ACP	The permission to configure the ACL of an object.	Only the bucket owner and authorized RAM users have permissions to configure the ACL of an object.

 **Notice** OSS supports only three ACL modes in S3: private, public read, and public read/write.

- Storage classes

OSS supports the following storage classes: Standard, Infrequent Access (IA), and Archive. Standard corresponds to `STANDARD`, IA corresponds to `STANDARD_IA`, and Archive corresponds to `GLACIER` in Amazon S3. You can convert the storage class of OSS objects based on your requirements.

To read an Archive object in OSS, you must use the Restore request to restore it. OSS ignores the lifetime configured for objects in the S3 API. By default, the restored state lasts for one day and can be extended up to seven days. Then, the object enters the frozen state again.

- ETag
  - If objects are uploaded by using the PUT method, the ETag of an OSS object and that of an Amazon S3 object differ in case sensitivity. The ETag is in uppercase for an OSS object but in lowercase for an S3 object. If your client uses ETag to validate content, configure your client to ignore the case sensitivity to avoid errors.
  - If objects are uploaded by using the multipart upload method, OSS calculates ETag values in a way that is different from S3.

## Compatible S3 API operations

The following API operations in S3 are compatible with OSS.

Operation	API
Bucket operation	<ul style="list-style-type: none"> <li>• PutBucket</li> <li>• DeleteBucket</li> <li>• GetBucket</li> <li>• GetBucketACL</li> <li>• GetBucketLifecycle</li> <li>• GetBucketLocation</li> <li>• GetBucketLogging</li> <li>• HeadBucket</li> <li>• PutBucketACL</li> <li>• PutBucketLifecycle</li> <li>• PutBucketLogging</li> </ul>
Object operation	<ul style="list-style-type: none"> <li>• DeleteObject</li> <li>• DeleteObjects</li> <li>• GetObject</li> <li>• GetObjectACL</li> <li>• HeadObject</li> <li>• PostObject</li> <li>• PutObject</li> <li>• PutObjectCopy</li> <li>• PutObjectACL</li> </ul>
Multipart operation	<ul style="list-style-type: none"> <li>• InitiateMultipartUpload</li> <li>• AbortMultipartUpload</li> <li>• CompleteMultipartUpload</li> <li>• ListParts</li> <li>• UploadPart</li> <li>• UploadPartCopy</li> </ul>

## 7.4. Use ossimport to migrate data

You can use `ossimport` to migrate data from local storage, third-party storage, or Object Storage Service (OSS) buckets in a region to OSS buckets in different regions. This topic describes how to use `ossimport` to migrate data from a third-party storage service to OSS.

### Context

Assume that a user has 500 TB of data stored in the Guangzhou region of Tencent Cloud Object Storage (COS). The user wants to use `ossimport` to migrate the data to a bucket in the China (Hangzhou) region of OSS within a week. During the migration process, business operations must continue to run normally.

`ossimport` can be deployed in standalone mode or distributed mode:

- The standalone mode is applicable when you migrate data volumes smaller than 30 TB.
- The distributed mode is suitable to migrate data volumes larger than 30 TB.

To migrate large amounts of data, deploy `ossimport` in distributed mode.

 **Note** You can also use data online migration to easily migrate data. For more information, see [Background information](#) about Data Online Migration.

### Preparations

- Activate OSS. Create a bucket in the China (Hangzhou) region.
  - For more information about how to activate OSS, see [Activate OSS](#).
  - For more information about how to create a bucket, see [Create buckets](#).
- Configure a Resource Access Management (RAM) user and grant OSS access permissions to the RAM user.

Create a RAM user in the RAM console. Authorize the RAM user to access OSS, and then save the AccessKey ID and the AccessKey secret. For more information, see [Create a Resource Access Management \(RAM\) user and grant required permissions to the RAM user](#).

- (Optional) Purchase an Elastic Compute Service (ECS) instance.

The ECS instance and OSS instance are located in the same region, which is China (Hangzhou). For more information about ECS instances, see [General-purpose instance families](#). We recommend that you purchase a pay-as-you-go instance if you want to release the ECS instance after the data is migrated.

 **Note** If you want to deploy `ossimport` to a small number of machines, you can deploy them locally. If you want to deploy `ossimport` to a large number of machines, we recommend that you deploy them on an ECS instance. An ECS instance is used in the example to show how to perform a migration task.

The number of required ECS instances is calculated based on the formula: Number of required ECS instances =  $X/Y/(Z/100)$ . In the formula, X indicates the amount of data to be migrated. Y indicates the required duration in days. Z indicates the migration speed in Mbit/s (about Z/100 TB of data to be migrated each day). If the migration speed of an ECS instance reaches 200 Mbit/s (about 2 TB of data is migrated each day), you need to purchase 36 ECS instances ( $500/7/2$ ) in the preceding example).

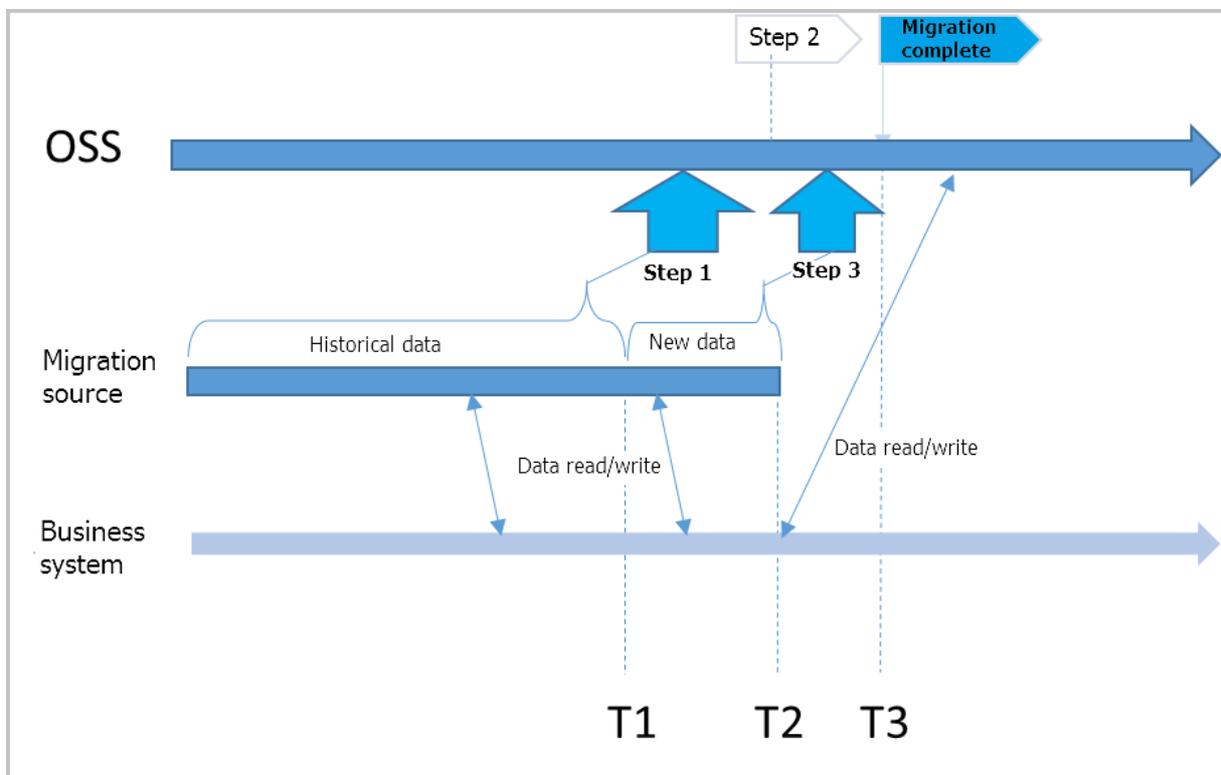
- Configure ossimport properly.

To meet the large-scale migration requirements in this example, you must deploy ossimport in distributed mode on ECS. For more information about the configuration and definition of distributed deployment, such as `conf/job.cfg`, `conf/sys.properties`, and concurrency control, see [Architectures and configurations](#). For more information about operations on distributed deployment such as downloading ossimport and troubleshooting common errors during configurations, see [Distributed deployment](#).

## Migration solutions

The following process describes how to migrate data from a third-party storage service to OSS in distributed mode.

**Note** After you deploy ossimport in the distributed environment on the ECS instance, ossimport downloads data from the Guangzhou region of Tencent Cloud Object Storage (COS) to the ECS instance located in the China (Hangzhou) region. We recommend that you use the Internet when you migrate data. To use ossimport to upload data from the ECS instance to the OSS instance within the China (Hangzhou) region, we recommend that you use the internal network.



Costs involved in the migration process include the fees incurred when the source and destination buckets are accessed, outbound traffic fees for the source bucket, ECS instance fees, data storage fees. If more than 1 TB of data is to migrate, the storage cost and the migration period increase proportionally. Compared with the data transfer and storage fees, fewer fees are incurred when you use ECS. If more ECS instances are used, the migration period is shortened.

## Implementation

1. Migrate all data last modified before T1.

For more information, see the [Running](#) section in Distributed deployment.

 **Notice** T1 is a Unix timestamp that indicates the number of milliseconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC. You can run the `date +%s` command to obtain the value.

2. Configure a mirroring-based back-to-origin rule.

The origin keeps generating new data during the migration. To ensure business continuity and a seamless switchover, you need to configure a back-to-origin rule. When objects that are requested by users do not exist in OSS, OSS retrieves these objects from the origin and returns the objects to the users. For more information, see [Overview](#).

3. Switch the read/write operations on the business system to OSS. At this time, the business system records the time at T2.
4. Open the `job.cfg` configuration file. Specify `importSince=T1`. Reinitiate the migration task to migrate incremental data last modified between T1 and T2.

 **Note**

- After Step 4 is complete, all read and write operations on your business system are switched to OSS. Data stored in the third-party storage service is only a copy of historical data, which can be retained or deleted.
- `ossimport` only migrates and verifies data, but does not delete data.

## References

For more information about `ossimport`, see the following topics:

[Distributed deployment](#)

[Architectures and configurations](#)

[Troubleshooting](#)

# 7.5. Migrate data from HDFS to OSS

This topic describes how to use Jindo DistCp to migrate data from Hadoop Distributed File System (HDFS) to Object Storage Service (OSS). Jindo DistCp is a data copy tool provided by Alibaba Cloud.

## Context

HDFS is used as the underlying storage for a large amount of data in traditional big data architectures. The DistCp tool provided by Hadoop is used to migrate or copy data in HDFS. However, this tool cannot take advantage of the features of OSS, which results in low efficiency and poor data consistency. In addition, DistCp provides only simple features, which cannot meet user requirements.

Jindo DistCp is used to copy files in a distributed file system, and you can use it to copy files within or between large-scale clusters. Jindo DistCp uses MapReduce to distribute files, handle errors, and restore data. Lists of files and directories are used as the input of the map and reduce tasks. Each task copies some files and directories in the input list. Jindo DistCp allows you to copy data between HDFS DataNodes, between HDFS and OSS, and between OSS buckets. It also provides various custom parameters and policies for data copying.

Compared with Hadoop DistCp, Jindo DistCp has the following advantages in data migration from HDFS to OSS:

- High efficiency. The data migration speed of Jindo DistCp is 1.59 times faster than that of Hadoop DistCp.
- Rich basic features. Jindo DistCp provides multiple copy methods and various scenario-based optimization policies.
- Deep integration with OSS. Jindo DistCp takes advantage of OSS features so that you can perform various operations on files, including file compression and the storage class conversion to Archive.
- File copying without changing file names. This ensures data consistency.
- High compatibility. Jindo DistCp is applicable to various scenarios and can be used to replace Hadoop DistCp. Jindo DistCp supports Hadoop 2.7.x and Hadoop 3.x.

## Prerequisites

- If you use a self-managed Elastic Compute Service (ECS) cluster, a Hadoop 2.7.x or Hadoop 3.x environment is available and MapReduce jobs can be run in the environment.
- If you use Alibaba Cloud E-MapReduce (EMR):
  - For EMR V3.28.0 or Bigfoot 2.7.0 or later, you can run Shell commands to use Jindo DistCp. For more information, see [Use Jindo DistCp](#).
  - For EMR V3.28.0 or Bigfoot 2.7.0 or earlier, you may encounter compatibility issues. In that case, submit a [ticket](#) for technical support.

## Step 1: Download the JAR package of Jindo DistCp

- [JAR package of Jindo DistCp for Hadoop 2.7.x](#)
- [JAR package of Jindo DistCp for Hadoop 3.x](#)

## Step 2: Configure the AccessKey pair used to access OSS

You can configure the AccessKey pair by using one of the following methods:

- Configure the AccessKey pair by using the related command.

For example, copy the directory in HDFS to a specified path in OSS and configure `--ossKey`, `--ossSecret`, and `--ossEndPoint` in the following command:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming/examplefile --dest oss://examplebucket/example_file --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com
```

- Configure the AccessKey pair by using a configuration file.

Configure `--ossKey`, `--ossSecret`, and `--ossEndPoint` in the `core-site.xml` file of Hadoop. The following code provides an example on how to configure the AccessKey pair in the configuration file:

```
<configuration>
  <property>
    <name>fs.oss.accessKeyId</name>
    <value>xxx</value>
  </property>
  <property>
    <name>fs.oss.accessKeySecret</name>
    <value>xxx</value>
  </property>
  <property>
    <name>fs.oss.endpoint</name>
    <!-- If you access OSS from an ECS instance, we recommend that you use an internal endpoint of OSS in the oss-cn-xxx-internal.aliyuncs.com format. -->
    <value>oss-cn-xxx.aliyuncs.com</value>
  </property>
</configuration>
```

- Use the password-free feature of JindoFS SDK.

Use the password-free feature of JindoFS SDK so that you do not need to store your AccessKey pairs in plaintext. This improves data security. For more information, see [Use the password-free feature of JindoFS SDK](#).

### Step 3: Migrate or copy data

Jindo DistCp V3.7.3 is used in this example. Replace the version number with your actual version number.

- Migrate or copy full data.

The following command provides an example on how to migrate or copy full data from the `/data/incoming` directory in HDFS to the `oss://examplebucket/incoming/` path in OSS:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --parallelism 10
```

The following table describes the parameters and options in the command.

Parameter/Option	Description	Example
<code>--src</code>	The source path of the data to migrate or copy in HDFS.	<code>/data/incoming</code>
<code>--dest</code>	The destination path of the data to migrate or copy in OSS.	<code>oss://examplebucket/incoming</code>
<code>--ossKey</code>	The AccessKey ID used to access OSS. For information about how to obtain an AccessKey ID, see <a href="#">Obtain an AccessKey pair</a> .	<code>LTAI5t7h6SgiLSganP2m****</code>

Parameter/Option	Description	Example
<code>--ossSecret</code>	The AccessKey secret used to access OSS. For information about how to obtain an AccessKey secret, see <a href="#">Obtain an AccessKey pair</a> .	KZo149BD9GLPNiDIEmdQ7dyNKG****
<code>--ossEndPoint</code>	The endpoint of the region in which the bucket is located. For more information about the regions and endpoints of OSS, see <a href="#">Regions and endpoints</a> .   <b>Notice</b> If you access OSS from an ECS instance, we recommend that you use an internal endpoint of OSS in the <code>oss-cn-xxx-internal.aliyuncs.com</code> format.	oss-cn-hangzhou.aliyuncs.com
<code>--parallelism</code>	The number of data migration or data copying tasks that can be concurrently run based on the amount of resources in your cluster.	10

- Migrate or copy incremental data

If you want to migrate or copy only the incremental data from the source path after full data migration or copying, you can use the `--update` option.

The following command provides an example on how to migrate or copy only incremental data from the `/data/incoming` directory in HDFS to the `oss://examplebucket/incoming` path in OSS:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --update --parallelism 10
```

By default, checksum is enabled when you use the `--update` option. This way, Jindo DistCp compares file names, file sizes, and the checksums of files in the source path and the destination path. If data inconsistency is detected in the preceding items, incremental data migration or copying is automatically started.

To disable Jindo DistCp from comparing the checksums of files in the source path and the destination path, add the `--disableChecksum` option to the command. Example:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --update --disableChecksum --parallelism 10
```

## Appendix 1: Parameters and options supported by Jindo DistCp

Jindo DistCp provides a variety of parameters and options. You can run the following command to obtain information about the parameters and options:

```
hadoop jar jindo-distcp-3.7.3.jar --help
```

The following table describes the parameters and options.

Parameter/Option	Description	Example
--src	The source path of files to copy.	--src oss://exampleBucket/sourceDir
--dest	The destination path of files to copy.	--dest oss://exampleBucket/destDir
--parallelism	The number of copying tasks that can be concurrently run. You can set this parameter based on the amount of resources in your cluster.	--parallelism 10
--policy	The storage class of the files after they are copied to OSS. Valid values: <ul style="list-style-type: none"><li>• <i>ia</i>: Infrequent Access (IA)</li><li>• <i>archive</i>: Archive</li><li>• <i>ColdArchive</i>: Cold Archive</li></ul>	--policy archive
--srcPattern	Specifies that a regular expression is used to filter files to copy. You must use the full path in the regular expression.	--srcPattern .*\.log
--deleteOnSuccess	Specifies that the files to copy from the source path are deleted after they are copied to the destination path.	--deleteOnSuccess

Parameter/Option	Description	Example
<code>--outputCodec</code>	<p>The compression method for the files to copy. Compression codecs available in the current version are <code>gzip</code>, <code>gz</code>, <code>lzo</code>, <code>lzop</code>, and <code>snappy</code>. Jindo DistCp also supports following keywords: <code>none</code> and <code>keep</code>. Default value: <code>keep</code>.</p> <ul style="list-style-type: none"> <li><code>none</code>: indicates that the files are saved uncompressed. If the files have been compressed, Jindo DistCp decompresses them.</li> <li><code>keep</code>: indicates that the files remain compressed. This is the default keyword.</li> </ul> <div style="background-color: #e0f2f7; padding: 5px; margin-top: 10px;"> <p> <b>Note</b> If you want to use the Lempel-Ziv-Oberhumer (LZO) compression algorithm in an open source Hadoop cluster, you must install the native library of <code>gplcompression</code> and the <code>Hadoop-LZO</code> package. Otherwise, we recommend that you use other compression methods.</p> </div>	<code>--outputCodec gzip</code>
<code>--srcPrefixesFile</code>	The list of files to copy. The files in the list are prefixed with the path specified by the <code>src</code> parameter.	<code>--srcPrefixesFile file:///opt/folders.txt</code>
<code>--outputManifest</code>	Specifies that a <code>gzip</code> file is generated in the directory specified by the <code>dest</code> parameter to record the information about files that are copied.	<code>--outputManifest=manifest-2020-04-17.gz</code>
<code>--requirePreviousManifest</code>	Specifies whether this copy operation reads files that are copied by previous copy operations. Valid values: <ul style="list-style-type: none"> <li><code>false</code>: indicates that the copy operation does not read files that are copied and directly copies full data.</li> <li><code>true</code>: indicates that the copy operation reads files that are copied and copies only incremental data.</li> </ul>	<code>--requirePreviousManifest=false</code>
<code>--previousManifest</code>	Specifies that the copy operation reads the path of the files that are copied and copies only incremental data.	<code>--previousManifest=oss://exampleBucket/manifest-2020-04-16.gz</code>

Parameter/Option	Description	Example
--copyFromManifest	Specifies that the files recorded in the manifest file are copied. Usually, --copyFromManifest is used with --previousManifest.	--previousManifest oss://exampleBucket/manifest-2020-04-16.gz --copyFromManifest
--groupBy	Specifies that regular expressions are used to group files that match specific conditions.	--groupBy='.*/[a-z]+.*.txt'
--targetSize	The threshold for the file size after the files are grouped. Unit: MB.	--targetSize=10
--enableBalancePlan	This option is used when files to copy have little difference in size. For example, the files are all larger than 10 GB or all smaller than 10 KB.	--enableBalancePlan
--enableDynamicPlan	This option is used when files to copy have a significant difference in size. For example, files larger than 10 GB and files smaller than 10 KB need to be copied in a task.	--enableDynamicPlan
--enableTransaction	This option is used to guarantee the data consistency between jobs. By default, only the data consistency between tasks is guaranteed.	--enableTransaction
--diff	This option is used to check whether all files are copied and generate a list of files that fail to be copied.	--diff
--ossKey	The AccessKey ID used to access OSS.	--ossKey LTAI5t7h6SgiLSganP2m****
--ossSecret	The AccessKey secret used to access OSS.	--ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG****
--ossEndPoint	The endpoint of the region in which the bucket is located.	--ossEndPoint oss-cn-hangzhou.aliyuncs.com
--cleanUpPending	This option is used to clean up files that are incompletely copied to OSS. It takes some time to clean up the files.	--cleanUpPending
--queue	The name of a YARN queue.	--queue examplequeue1
--bandwidth	The bandwidth for the data copying task in MB.	--bandwidth 6
--disableChecksum	This option is used to disable checksum verification.	--disableChecksum

Parameter/Option	Description	Example
--enableCMS	This option is used to enable the alerting feature of CloudMonitor.	--enableCMS
--update	Specifies that only incremental data is migrated to the destination path. Incremental data refers to data that is added to the source path after the last full data migration.	--update
--filters	Specifies the path of a file. Data in each line in the file contains a regular expression, which corresponds to files that do not need to be copied or compared.	--filters /path/to/filterfile.txt
--tmp	Specifies a directory to store temporary files when you use Jindo DistCp.	--tmp /data
--overwrite	Specifies that objects in the source path overwrite objects with the same names in the mapped destination path.	--overwrite
--ignore	Specifies that exceptions are ignored during data migration to ensure uninterrupted migration. Errors are reported in the form of Jindo DistCp counters. If the --enableCMS option is used, you receive notifications in a specified form.	--ignore

## Appendix 2: Sample scenarios

You can use the following methods to verify data integrity:

- Method 1: Use Jindo DistCp counters

Parameters included in the information of DistCp counters, such as BYTES\_EXPECTED and FILES\_EXPECTED, can be used to verify data integrity.

```
Example
JindoDistcpCounter
  BYTES_COPIED=10000
  BYTES_EXPECTED=10000
  FILES_COPIED=11
  FILES_EXPECTED=11
  ...
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
```

The following table describes parameters that may be included in the counters in the preceding example.

Parameter	Description
BYTES_COPIED	The number of bytes that have been copied.
BYTES_EXPECTED	The number of bytes to be copied.
FILES_COPIED	The number of files that have been copied.
FILES_EXPECTED	The number of files to be copied.
FILES_SKIPPED	The number of files that are skipped when only incremental data is copied.
BYTES_SKIPPED	The number of bytes that are skipped when only incremental data is copied.
COPY_FAILED	The number of files that fail to be copied. The alerting feature is triggered when the value is not 0.
BYTES_FAILED	The number of bytes that fail to be copied.
DIFF_FILES	The number of files that are different in the source path and the destination path. The alerting feature is triggered when the value is not 0.
DIFF_FAILED	The number of files that are not properly compared. The number is added to the DIFF_FILE value.
SRC_MISS	The number of files that do not exist in the source path. The number is added to the DIFF_FILES value.
DST_MISS	The number of files that do not exist in the destination path. The number is added to the DIFF_FILES value.
LENGTH_DIFF	The number of files that have identical names but different sizes in the source path and the destination path. The number is added to the DIFF_FILES value.
CHECKSUM_DIFF	The number of files that fail to pass the checksum verification. The number is added to the COPY_FAILED value.
SAME_FILES	The number of files that are identical in the source path and the destination path.

- Method 2: Use the `--diff` option

You can use the `--diff` option to compare the names and sizes of files in the source path and the destination path. Example:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --diff
```

1. You can use the `--diff` option to check whether all files in the specified path in HDFS are migrated to OSS.

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --diff
```

If the following command output is displayed, all files have been migrated. Otherwise, a manifest file is generated in the current working directory.

```
INFO distcp.JindoDistCp: Jindo DistCp job exit with 0
```

2. You can use the `--copyFromManifest` and `--previousManifest` options to migrate the files listed in the manifest file.

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --previousManifest=file:///opt/manifest-2020-04-17.gz --copyFromManifest --parallelism 20
```

In this case, `file:///opt/manifest-2020-04-17.gz` specified by `--previousManifest` is the local path in which the command is run.

You can use the `--policy` option to specify the storage class of files to migrate to OSS. The IA storage class is specified in the following example:

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --policy ia --parallelism 10
```

To set the storage class to Archive for files to migrate to OSS, replace `--policy ia` with `--policy archive`. To set the storage class to Cold Archive for files to migrate to OSS, replace `--policy ia` with `--policy coldArchive`. Cold Archive is supported only by some regions. For more information, see [Cold Archive](#).

- Files to migrate include a large number of small files and a small number of extra large files.

For example, files in the source path in HDFS include 500,000 files of 100 KB and 10 files of 5 TB. In this case, you can use the `--enableDynamicPlan` option to speed up data transmission.

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --enableDynamicPlan --parallelism 10
```

- Files to migrate have little difference in size.

For example, files in the source path in HDFS include 100 files of 200 KB. In this case, you can use the `--enableBalancePlan` option to speed up data transmission.

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming --ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZo149BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint oss-cn-hangzhou.aliyuncs.com --enableBalancePlan --parallelism 10
```

 **Note** `--enableDynamicPlan` and `--enableBalancePlan` cannot be used together.

You can use the `--deleteOnSuccess` option to delete specific data from the source path after the data is migrated or copied to the destination path.

```
hadoop jar jindo-distcp-3.7.3.jar --src /data/incoming --dest oss://examplebucket/incoming
--ossKey LTAI5t7h6SgiLSganP2m**** --ossSecret KZol49BD9GLPNiDIEmdQ7dyNKG**** --ossEndPoint
oss-cn-hangzhou.aliyuncs.com --deleteOnSuccess --parallelism 10
```

**Scenario 1: How do I verify data integrity after data is transmitted to OSS by using Jindo DistCp?**

**Scenario 2: Which parameters can I configure to resume data migration from HDFS to OSS in the case of a migration failure?**

**Scenario 3: Which parameters can I configure to specify the storage class of files to migrate to OSS as IA, Archive, or Cold Archive?**

**Scenario 4: I have a clear understanding of data distribution (such as the proportion of large files to small files) in the source path. Which parameters can I configure to speed up data transmission?**

**Scenario 5: After data is migrated or copied, which parameters can I configure to delete specific data from the source path?**

# 8. Security

## 8.1. Reduce the risks of unauthorized access caused by AccessKey pair leaks

If the AccessKey pairs of individual or enterprises users are leaked, users that are not supposed to have access to Object Storage Service (OSS) resources may be able to manage the resources and threaten data security. To address this issue, OSS provides a wide array of best practices for security for you to implement data security and protection.

 **Notice** The following best practices are a set of general rules, not a compressive set of security solutions. These best practices are for reference only. We recommend that you remain conscious about data security and take the necessary precautions and preventive measures.

### Block public access

Unless your business requires everyone, including anonymous visitors, be able to read data from and write data to your OSS resources (including buckets and objects), do not set the access control list (ACL) of the buckets or the objects to public read or public read/write. Description of public read/write and public read:

- **Public read/write:** Anyone, including anonymous users, can perform read and write operations on objects in the bucket.

 **Warning** All Internet users can access objects in the bucket and write data to the bucket. If you set the ACL to public read/write, the objects may be accessed unpredictably, which can lead to spiraling costs. If a user uploads prohibited data or information, you may be held liable. Therefore, we recommend that you do not set the File ACL parameter to public read/write except in special cases.

- **Public read:** Only the bucket owner can perform write operations on the objects in the bucket. Other users, including anonymous users, can perform only read operations on the objects in the bucket.

 **Warning** All Internet users can access objects in the bucket. This may result in unexpected access to the data in your bucket and out-of-control costs. Exercise caution when you set your bucket ACL to public read.

To reduce security risks caused by public access, we recommend that you set the ACL of a bucket or an object to private. After you set the ACL to private, only the bucket owner can read and write the bucket and the objects in the bucket.

You can use multiple methods to set the ACL of a bucket or an object to private. For more information, see [Configure the ACL feature for a bucket](#) and [Configure ACL for objects](#).

### Do not include plaintext AccessKey pairs in code or store encrypted AccessKey pairs locally

Plaintext AccessKey pairs in code may be leaked with the code. AccessKey pairs locally encrypted and stored are also not secure because encrypted and decrypted content is stored in the memory and the data in the memory can be stored in another device. Mobile apps and applications on PCs are prone to these risks. To obtain decrypted data, attackers need only to use technologies such as injection, API hooking, and dynamic debugging.

On the server, you can use managed secret plug-ins for Alibaba Cloud SDKs to avoid plaintext AccessKey pairs in code. This way, you can prevent AccessKey pairs from being leaked with source code or compiled code. For more information about managed secret plug-ins for Alibaba Cloud SDKs, see [Managed secret plug-ins for Alibaba Cloud SDKs](#).

 **Notice** This method does not apply to clients. Do not embed AccessKey pairs in clients.

## Access OSS by using a RAM user

An Alibaba Cloud account has permissions to call all API operations. If you use the AccessKey pair of an Alibaba Cloud account, security risks may occur. We recommend that you create and use a RAM user to call API operations or perform routine O&M.

You can create RAM users and grant the RAM users different permissions to control their access to your resources. RAM allows you to keep your Alibaba Cloud account and password strictly confidential in scenarios where multiple users in your enterprise need to collaboratively manage cloud resources. It also allows you to grant the users the minimum required permissions, which ensures high security. For more information about how to create a RAM user, see [Create a RAM user](#).

After you create a RAM user, you can use a RAM policy to grant permissions to the RAM user. This way, you can manage your users, such as employees, systems, and applications, and control the resources that can be accessed by these users. For example, you can use the following RAM policy to prevent specified RAM users from accessing a bucket named examplebucket and objects or directories in examplebucket.

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "oss:*",
      "Resource": [
        "acs:oss:*:*:examplebucket",
        "acs:oss:*:*:examplebucket/*"
      ]
    }
  ]
}
```

You can also use RAM policies to prevent RAM users from deleting a directory in a bucket, or authorize RAM users to only read resources in a bucket. For more information about RAM policy examples, see [Common examples of RAM policies](#).

## Enable MFA

MFA is an easy-to-use and effective authentication method. When MFA is enabled, a dynamic authentication code generated by an MFA device is required in addition to your username and password when you attempt to log on to the Alibaba Cloud Management Console. This way, unauthorized access can be blocked to ensure security of your account in the event of password leaks.

You can choose to enable MFA for an Alibaba Cloud account. For more information, see [Enable an MFA device for an Alibaba Cloud account](#). You can choose to enable MFA for a RAM user. For more information, see [Enable an MFA device for a RAM user](#).

## Access OSS by using a temporary access credential provided by STS

You can use Security Token Service (STS) to generate a temporary credential to allow a user to access your OSS resources within the specified period. This way, you do not need to share your AccessKey pair or risk the security of your OSS resources.

For more information about how to use STS to grant temporary permissions to access OSS, see [Use a temporary credential provided by STS to access OSS](#) in OSS Development Guide.

## Configure bucket policies

You can configure bucket policies to authorize other users to access specified OSS resources for a bucket. For example, you can configure bucket policies to authorize other accounts to access or manage all resources or part of resources in your bucket. You can also configure bucket policies to grant different permissions to different RAM users of the same account.

When you configure bucket policies, follow the principle of least privilege (PoLP) to minimize security risks.

- Do not authorize users to access all resources in your bucket

To avoid excessive permissions and illegal access, we recommend that you do not specify all resources in your bucket and you grant permissions only on required resource paths.

- Do not allow anonymous access

Anonymous accounts can be used to access OSS if an endpoint and a bucket name are provided. However, endpoints can be enumerated, and bucket names can be obtained from the URLs of objects they are authorized to access. Therefore, anonymous access increases security risks.

- Specify Action

When you configure a bucket policy in the OSS console, Action that provides four authorization operations is only a convenient method for users to configure policies, and the specified action may not meet your business requirements. We recommend that you grant only required permissions to users by using Advanced Settings. For example, read-only permissions include `oss:ListObjects` and `oss:GetObject`. In most scenarios, only the `oss:GetObject` permission is required if you want to download an object.

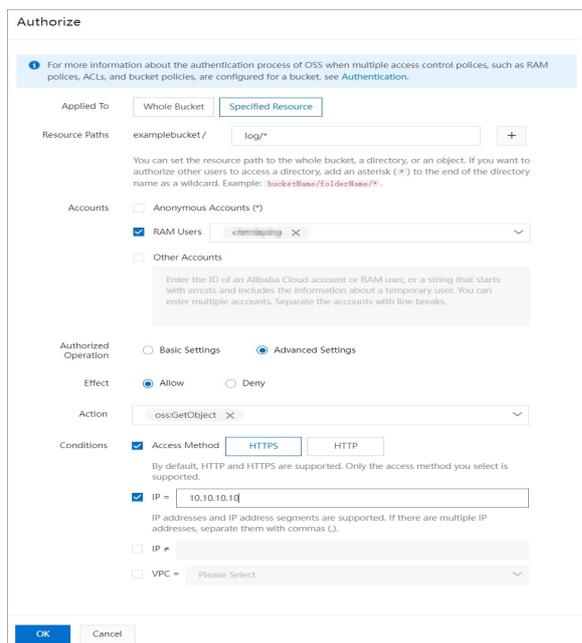
- Enable HTTPS for access

You can enable HTTPS to resolve issues such as man-in-the-middle attacks and domain hijacking. In addition, if you use Google Chrome to view an HTTPS website, HTTP resources of the website cannot be loaded by default. Make sure that you enable HTTPS, which is the most cost-effective way to resolve various risks.

- Specify source IP addresses

If the IP addresses used to access OSS resources are specified and can be enumerated, we recommend that you configure IP addresses.

For example, you can use a bucket policy to authorize the Test RAM user to download all objects in the log directory in examplebucket by using OSS SDKs or command-line tool ossutil.



## 8.2. Reduce the risks of unexpectedly high fees caused by malicious access traffic

When your Object Storage Service (OSS) buckets suffer attacks or fraudulent traffic, sudden spikes in traffic occur. This causes your storage fees to be unexpectedly higher than your regular storage costs. To address this issue, we recommend that you implement the following security best practices for data security and protection.

**Notice** The following best practices are a set of general rules, not a comprehensive set of security solutions. These best practices are for reference only and may not be applicable to your business scenarios. We recommend that you remain aware of data security threats and take the necessary precautions and preventive measures.

### Block public access

Do not set the access control list (ACL) of the buckets or the objects to public read or public read/write unless your business requires all users, including anonymous users, to be able to read data from and write data to your OSS resources. Description of public read/write and public read:

- Public read/write: All users, including anonymous users, can perform read and write operations on objects in the bucket.

**Warning** All users can access objects in the bucket and write data to the bucket. If you set the ACL to public read/write, unexpected access to objects can cause unexpectedly high fees. Therefore, we recommend that you do not set your bucket ACL to public read/write except in special cases.

- **Public read:** Only the bucket owner can perform write operations on the objects in the bucket. Other users, including anonymous users, can perform only read operations on the objects in the bucket.

**Warning** All Internet users can access objects in the bucket. This may result in unexpected access to the data in your bucket and unexpectedly high costs. Exercise caution when you set your bucket ACL to public read.

To reduce security risks caused by public access, we recommend that you set the ACL of a bucket or an object to private. After you set the ACL to private, only the bucket owner can read and write the bucket and the objects in the bucket.

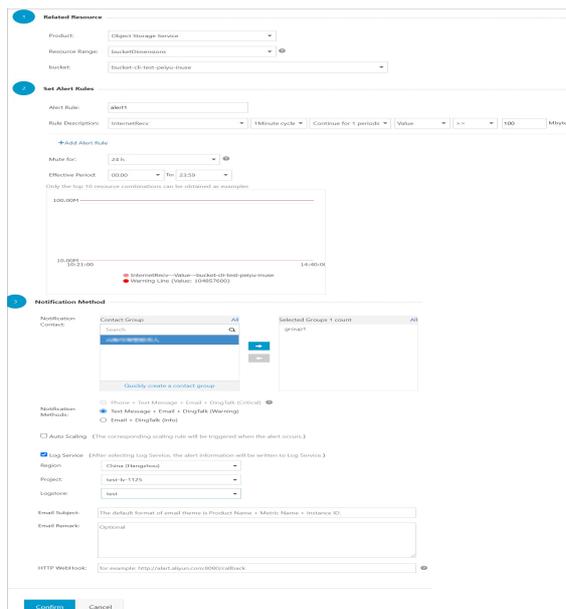
You can use multiple methods to set the ACL of a bucket or an object to private. For more information, see [Configure the ACL feature for a bucket](#) and [Configure ACL for objects](#).

## Configure alert rules in the Cloud Monitor console

You can create alert rules to monitor the usage and the status of cloud service resources. When an alert rule is triggered, CloudMonitor sends an alert notification to you. This way, you can be informed of exceptions and handle them efficiently.

For example, you can configure an alert rule for a bucket to send you a notification by using Short Message Service (SMS), email, and DingTalk and write the alert information to the Logstore specified by using Log Service when the Internet inbound traffic, Internet outbound traffic, CDN inbound traffic, or CDN outbound traffic is equal to or greater than 100 MB.

The following figure shows how to configure an alert rule that is triggered when Internet inbound traffic is equal to or greater than 100 MB.



You can configure alert rules for a bucket, and you can also configure alert rules for all OSS resources owned by your Alibaba Cloud account. For more information about how to configure alert rules, see [Create an alert rule](#).

## Configure hotlink protection

You can configure a Referrer whitelist for a bucket to prevent your resources in the bucket from unauthorized access.

OSS determines the source from which a request is sent based on the Referrer header field in the request. When a browser sends a request to the web server, the Referrer field is contained in the request to indicate the source from which the request is sent. OSS determines whether to allow or deny the request based on the Referrer field contained in the request and the Referrer whitelist configured for the specified bucket. If the Referrer field in the request matches the Referrer whitelist, the request is allowed. Otherwise, the request is denied.

For example, in the following figure, the Referrer whitelist is set to `https://www.example.com`, and empty Referrer is not allowed.



After you configure the Referrer, assume that User A adds an image object named `exampleobject.png` to the `https://www.example.com` website. When a user accesses the image on the website, the browser sends a request in which the value of the Referrer field is `https://www.example.com`. OSS allows the request because the Referrer field in the request is included in the Referrer whitelist.

User B adds the URL of the image object to the `https://example.org` website without authorization. When a user accesses the image on the website, the browser sends a request in which the value of the Referrer field is `https://example.org`. OSS denies the request because the Referrer field in the request is not included in the Referrer whitelist.

For more information about hotlink protection, see [Hotlink protection](#) in Developer Guide.

## Configure CORS

Cross-origin resource sharing (CORS) is a standard cross-origin solution provided by HTML5 to allow web application servers to control cross-origin access. This way, the security of data transmission across origins is ensured. Browsers check cross-origin requests based on the same-origin policy to keep the website content secure. When a request is sent from Website A by using JavaScript to access Website B of another origin, the browser rejects the request. In this case, you can configure CORS rules to allow cross-origin requests.

OSS allows you to configure CORS rules to allow or deny cross-origin requests based on your requirements. For example, the following figure shows the CORS rule configuration if you want to allow only requests whose source is `www.aliyun.com` and whose cross-origin request method is `GET`:

**Create Rule**

Sources \*  
www.allyun.com

You can set multiple sources. Each line can contain one source and up to one wildcard (\*).

Allowed Methods \*  
 GET  POST  PUT  DELETE  HEAD

Allowed Headers  
You can set multiple allowed headers. Each line can contain one header and up to one wildcard (\*).

Exposed Headers  
You can set multiple exposed headers. Each line can contain one header and cannot contain wildcards (\*).

Cache Timeout (Seconds)  
0

Vary: Origin   
Configure whether to return the Vary: Origin header. If both CORS and non-CORS requests are sent to OSS, select this option to avoid errors. If Vary: Origin is selected, access through the browser or CDN back-to-origin requests may increase. [Learn more.](#)

OK Cancel

For more information about CORS, see [Configure CORS](#).

## Do not name objects by using sequential prefixes

When you upload a large number of objects and name them by using sequential prefixes such as timestamps and letters, dates, numeric IDs that can be traversed, attackers can figure out the name rules and create a script to obtain all the objects. This causes data leakage. Therefore, we recommend that you add hexadecimal hash prefixes to your objects or name the objects by reversing the order of digits that indicate seconds in object names. This can reduce the risk that the object names are traversed. For more information, see [OSS performance and scalability best practices](#).

## 8.3. Reduce the risks of data loss caused by accidental operations

Accidental deletions of data are common and can affect business operations. To address this issue, select one of the preventive methods described in this topic.

 **Notice** The following best practices are a set of general rules, not a comprehensive set of security solutions. These best practices are for reference only because they may be unsuitable for your environment. We recommend that you remain conscious about data security and take the necessary precautions and preventive measures.

### Enable versioning for a bucket

Object Storage Service (OSS) allows you to configure versioning for a bucket to protect objects stored in the bucket. After you enable versioning for a bucket, data that is overwritten or deleted in the bucket is saved as a previous version. Versioning allows you to recover a previous version of an object to protect the object from being accidentally overwritten or deleted.

When versioning is enabled for a bucket, OSS specifies a unique ID for each version of an object stored in the bucket. Perform the following steps in the OSS console:

- Enable versioning when you create a bucket

- i. Log on to the [OSS console](#).
  - ii. In the left-side navigation pane, click **Buckets**. On the Buckets page, click **Create Bucket**.
  - iii. In the **Create Bucket** panel, configure parameters.  
Set **Versioning** to **Enable**. For more information about how to configure other parameters, see [Create buckets](#).
  - iv. Click **OK**.
- Enable versioning for an existing bucket
    - i. In the left-side navigation pane, click **Buckets**. On the Buckets page, click the name of the bucket for which you want to enable versioning.
    - ii. Choose **Redundancy for Fault Tolerance > Versioning**.
    - iii. Click **Configure**. Set Versioning to **Enabled**.
    - iv. Click **Save**.

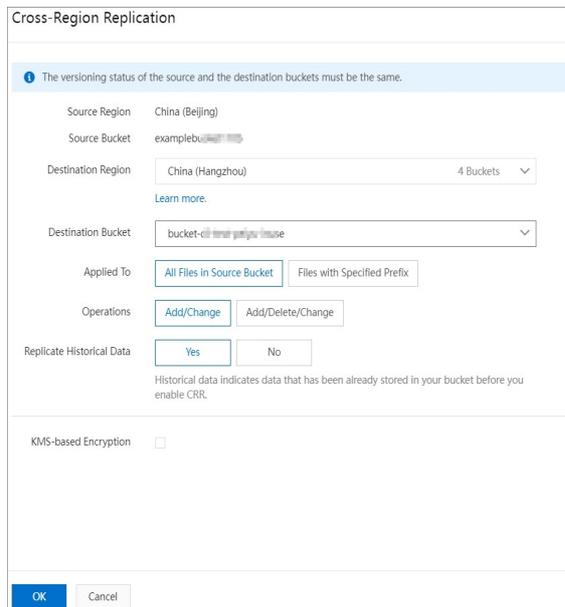
For more information about versioning, see [Overview](#).

## Enable CRR for a bucket

Cross-region replication (CRR) enables the automatic and asynchronous (near real-time) replication of objects across buckets in different OSS regions. Operations such as create and overwrite objects can be synchronized from a source bucket to a destination bucket.

CRR can meet your requirements on cross-region disaster recovery and data replication. You can set Operations of a CRR rule to **Add-Change** to back up data that is accidentally deleted.

For example, after you set the parameters of a CRR rule based on the following figure, all data in the source bucket named srcbucket is synchronized to the destination bucket named destbucket. This way, an object accidentally deleted in the srcbucket bucket can be found in the destbucket bucket.



For more information about CRR, see [CRR](#).

## Configure scheduled backup for a bucket

You can use the scheduled backup feature provided by OSS to back up objects in a bucket to Hybrid Backup Recovery (HBR) on a regular basis. If an object is accidentally lost, you can recover the object from HBR.

For example, after you configure a backup schedule for a bucket named `examplebucket` based on the configurations in the following figure, HBR backs up all objects in the bucket at the start time and permanently saves the backups in the destination backup vault based on the retention policy. For more information about how to configure scheduled backup, see [Configure scheduled backup](#).

After HBR completes a full backup, you can create an OSS restore job in the HBR console to recover an object accidentally deleted in the `examplebucket` bucket to the version before the object was deleted. For more information about how to create an OSS restore job, see [Create an OSS restore job](#).

## 8.4. Sensitive data protection

This topic describes how to integrate Alibaba Cloud Object Storage Service (OSS) with Sensitive Data Discovery and Protection (SDDP) to identify, classify, and protect sensitive data.

### Prerequisites

- SDDP is activated.  
For more information, see [Quick start](#).
- OSS is activated.  
For more information, see [Activate OSS](#).

### Context

Sensitive data is stored in a variety of forms across different storage systems, and can include high value data such as personal data, passwords, keys, and sensitive images. How to identify, locate, and protect sensitive data is essential. OSS provides a number of options to secure data, such as fine-grained [access control](#) and [data encryption](#). OSS also provides data protection mechanisms such as [ZRS](#), [cross-region replication](#), and [versioning](#), as well as monitoring and audit capabilities such as [Logging](#) and [Real-time log query](#). You can also integrate OSS with SDDP to better identify, classify, and protect sensitive data.

After you authorize SDDP to scan your OSS buckets, SDDP identifies sensitive data from your large amounts of data, classifies and displays sensitive data by risk level, and tracks how sensitive data is used. In addition, SDDP protects and audits sensitive data based on predefined security rules so that you can obtain the security status of your data assets in OSS buckets at any time. For more information, see [What is DSC?](#).

 **Note** After you authorize SDDP to scan your OSS buckets, SDDP scans all objects stored in your OSS buckets at the first scan and charges you for a full scan. If you add new objects to or modify objects in your OSS buckets after the first scan, SDDP will only scan the new or modified objects to minimize charges. For more information about billing methods, see [Subscription](#).

## Scenarios

- Sensitive data identification

Enterprises have large amounts of data, but they cannot accurately identify whether their data contains sensitive information or where the sensitive data is located. You can integrate OSS with SDDP to scan and classify data stored in OSS by using the built-in rules for algorithms of SDDP or by using custom rules that meet your industry requirements. You can also make further protection arrangements based on scan results. For example, OSS provides access control and encryption features to protect data.

- Data masking

If you share data for analysis or use without first masking it, sensitive data may be leaked. Built-in and custom masking algorithms are available when OSS is integrated with SDDP. You can use these algorithms to mask sensitive data in the production environment before data is transferred to other environments such as the development and testing environments. This ensures that the sensitive data remains secure while being usable in other environments.

- Anomaly detection and audit

SDDP uses an intelligent model to analyze and audit access to sensitive data in OSS. If a risk is detected, SDDP sends an alert to your data security team. This helps you improve risk prediction and prevention capabilities.

## Benefits

- Visual

- SDDP displays sensitive data detection results on a graphical user interface (GUI), which allows you to clearly view the security status of your data.
- SDDP monitors data access and provides audit logs for you to trace anomalous activities, which reduces security risks for your data.
- SDDP increases the overall security transparency of your data assets and enhances data governance.
- SDDP reduces the cost of maintaining data security and provides fundamental data for you to formulate security rules that are suitable for your enterprise.

- Intelligent

- SDDP uses big data and machine learning technologies as well as intelligent algorithms to detect and monitor sensitive data and high-risk activities such as anomalous data access and potential data leaks. Additionally, SDDP provides suggestions to resolve detected issues.

- SDDP allows you to customize the rules to detect sensitive data so that you can ensure that sensitive data is detected and protected more accurately and efficiently.
- SDDP integrates complex data formats and content to a unified data risk model and presents data in a standard manner for you to protect your key data assets.
- Cloud-native
  - SDDP takes advantage of cloud services and supports multiple cloud data sources.
  - Compared with traditional sensitive data protection software, SDDP provides a more robust service architecture and higher availability at lower costs and features higher system security.

## Procedure

- 1.
- 2.
3. On the **OSS** tab, click **Unauthorized**.
4. Select the required OSS bucket and click **Batch Operation**.

You can also click **Authorization** in the Open protection column corresponding to the required bucket to authorize a bucket.
5. In the **Batch processing for selected assets** dialog box, configure the following parameters:
  - **Identify permissions**: specifies whether to grant SDDP the sensitive data detection permission on the selected data assets.
  - **Audit permissions**: specifies whether to grant SDDP the audit permission on the selected data assets.
  - **Desensitization permissions**: specifies whether to grant SDDP the sensitive data de-identification permission on the selected data assets.
  - **Sensitive data sampling**: the number of samples to be collected from the selected data assets. Valid values: 0, 5, and 10.
  - **Audit log archiving**: the number of days for which audit logs are retained for the selected data assets. Unit: days. Valid values: 30, 90, and 180.

 **Note** You do not need to activate Log Service to archive audit logs generated by SDDP.

6. Click **Ok**.

After the authorization is complete, scans authorized OSS buckets for sensitive data. When accesses an OSS bucket for the first time, DSC automatically scans all the data in the OSS bucket, and you are charged for the full scan. For more information, see the "How long does it take to scan data in my data asset after I authorize DSC to access the data asset?" section of the [Sensitive data scan and detection](#) topic.

In the list of authorized data assets, you can modify the authorization configuration for a data asset or cancel the authorization for a data asset. After you cancel the authorization for an OSS bucket, no longer scans the OSS bucket.

 **Note** scans only authorized OSS buckets and analyzes risks of sensitive data detected in these OSS buckets.

7. Add a security policy.

After sensitive data is scanned, you can start to take measures to harden security, such as configuring [server-side encryption](#) and [access control](#) based on the scan results.

# 9. IaC automation

## 9.1. Terraform

### 9.1.1. Overview

Terraform is an open source automatic resource orchestration tool that supports multiple cloud service providers. Alibaba Cloud is the third largest cloud service provider. The Alibaba Cloud provider supports at least 90 resources and data sources across more than 20 services and products. An increasing number of developers contribute to the Alibaba Cloud Terraform ecosystem. For more information, visit [terraform-alicloud-provider](#).

HashiCorp Terraform is an automatic IT infrastructure orchestration tool that can use code to manage and maintain IT resources. The easy-to-use command line interface (CLI) of Terraform allows you to deploy configuration files on Alibaba Cloud or any other supported cloud and control the versions of the configuration files. The CLI provides code for infrastructure resources such as VMs, storage accounts, and network interfaces defined in the configuration files that describe the cloud resource topology. Terraform is a highly scalable tool that supports new infrastructures from providers. You can use Terraform to create, modify, or delete cloud resources, such as OSS objects, ECS instances, VPCs, ApsaraDB RDS instances, and SLB instances.

### Features of the OSS Terraform module

OSS Terraform Module provides bucket and object management features. Example:

- Bucket management features:
  - Create a bucket.
  - Configure an ACL for a bucket.
  - Configure Cross-Origin Resource Sharing (CORS) for a bucket.
  - Configure logging for a bucket.
  - Configure static website hosting for a bucket.
  - Configure hot link protection for a bucket.
  - Configure the lifecycle rules of a bucket.
- Object management features:
  - Upload an object.
  - Configure server-end encryption for an object.
  - Configure an ACL for an object.
  - Configure object metadata.

### References

- For more information about how to install and use Terraform, see [Use Terraform to manage OSS](#).
- For more information about the download address of the OSS Terraform module, visit [terraform-alicloud-modules](#).
- For more information about the OSS Terraform module, visit [alicloud\\_oss\\_bucket](#).

## 9.1.2. Use Terraform to manage OSS

This topic describes how to install, configure, and use Terraform to manage Object Storage Service (OSS).

### Install and configure Terraform

Before you use Terraform, perform the following steps to install and configure Terraform:

1. Download the software package applicable to your operating system from [Download Terraform](#).  
In this topic, Terraform is installed and configured in a Linux operating system as an example.
2. Decompress the package to `/usr/local/bin`.  
If you decompress the executable file to another path, you must add the path to global variables.
3. Run Terraform to verify the path configuration. If a list of available Terraform options is displayed, Terraform is installed.

```
[root@test bin]#terraform
Usage: terraform [-version] [-help] <command> [args]
```

4. Create a Resource Access Management (RAM) user and grant permissions to the user.
  - i. Log on to the [RAM console](#).
  - ii. Create a RAM user named `Terraform`. Then, create an AccessKey pair for the RAM user.  
For more information, see [Create a RAM user](#).
  - iii. Grant permissions to the RAM user.  
You can grant relevant management permissions to the `Terraform` RAM user. For more information, see [Grant permissions to a RAM user](#).

 **Notice** For security reasons, do not use the AccessKey pair of your Alibaba Cloud account to configure Terraform.

5. Create a test directory.

You must create a separate working directory for each Terraform project. In this example, create a test directory named `terraform-test`.

```
[root@test bin]#mkdir terraform-test
```

6. Go to the `Terraform-test` directory.

```
[root@test bin]#cd terraform-test
[root@test terraform-test]#
```

7. Create configuration files.

Terraform reads all `*.tf` and `*.tfvars` files in the directory when Terraform is run. You can write configuration information to different files based on the actual scenario. Frequently used configuration files:

```

provider.tf          -- used to configure providers
terraform.tfvars     -- used to configure the variables required to configure providers
variable.tf         -- used to configure universal variables
resource.tf         -- used to define resources
data.tf             -- used to define package files
output.tf           -- used to configure the output

```

For example, when you create the *provider.tf* file, you can configure your authentication information based on the following format:

```

[root@test terraform-test]# vim provider.tf
provider "alicloud" {
  region      = "cn-beijing"
  access_key  = "LTA*****NO2"
  secret_key  = "MOk8x0*****wwff"
}

```

For more information about configurations, see [alicloud\\_oss\\_bucket](#).

## 8. Initialize your working directory.

```

[root@test terraform-test]#terraform init
Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "alicloud" (1.25.0)...
The following providers do not have any version constraints in configuration,
so the latest version was installed.
To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.
* provider.alicloud: version = "~> 1.25"
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

```

 **Notice** After you create a working directory and configuration file for a Terraform project, you must initialize the working directory.

You can use Terraform after you complete the preceding operations.

## Use Terraform to manage OSS

After Terraform is installed, you can run Terraform commands to manage OSS. The following examples provide a description of some common commands of Terraform:

- **terraform plan:** You can run this command to view the operations to be performed before a configuration file is executed.

Assume that you add a configuration file named *test.tf* that is used to create a bucket.

```
[root@test terraform-test]#vim test.tf
resource "alicloud_oss_bucket" "bucket-acl"{
  bucket = "figo-chen-2020"
  acl = "private"
}
```

You can run the **terraform plan** command to view the operations to be performed based on the *test.tf* configuration file.

```
[root@test terraform-test]# terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

-----
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
Terraform will perform the following actions:
+ alicloud_oss_bucket.bucket-acl
   id:                <computed>
   acl:                "private"
   bucket:             "figo-chen-2020"
   creation_date:     <computed>
   extranet_endpoint: <computed>
   intranet_endpoint: <computed>
   location:          <computed>
   logging_isenable:  "true"
   owner:              <computed>
   referer_config.#: <computed>
   storage_class:     <computed>
Plan: 1 to add, 0 to change, 0 to destroy.

-----
Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

- **terraform apply:** You can run this command to execute a configuration file in the working directory.

For example, to create a bucket named *figo-chen-2020*, you must add a configuration file named *test.tf*.

```
[root@test terraform-test]#vim test.tf
resource "alicloud_oss_bucket" "bucket-acl"{
  bucket = "figo-chen-2020"
  acl = "private"
}
```

Then, run the **terraform apply** command to execute the configuration file.

```
[root@test terraform-test]#terraform apply
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
Terraform will perform the following actions:
+ alicloud_oss_bucket.bucket-acl
  id:                <computed>
  acl:                "private"
  bucket:            "figo-chen-2020"
  creation_date:     <computed>
  extranet_endpoint: <computed>
  intranet_endpoint: <computed>
  location:          <computed>
  logging_isenable:  "true"
  owner:             <computed>
  referer_config.#: <computed>
  storage_class:     <computed>
Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Enter a value: yes
alicloud_oss_bucket.bucket-acl: Creating...
  acl:                "" => "private"
  bucket:            "" => "figo-chen-2020"
  creation_date:     "" => "<computed>"
  extranet_endpoint: "" => "<computed>"
  intranet_endpoint: "" => "<computed>"
  location:          "" => "<computed>"
  logging_isenable:  "" => "true"
  owner:             "" => "<computed>"
  referer_config.#:  "" => "<computed>"
  storage_class:     "" => "<computed>"
alicloud_oss_bucket.bucket-acl: Creation complete after 1s (ID: figo-chen-2020)
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

 **Note** After you execute the configuration file, a new bucket is created if the *figo-chen-2020* bucket does not exist. If the *figo-chen-2020* bucket exists and contains no data, the existing bucket is overwritten by the new bucket.

- **terraform destroy**: You can run this command to delete an empty bucket created by Terraform.
- **terraform import** : If a bucket is not created by using Terraform, you can run this command to import an existing bucket.

Before you run this command, run the following command to create a file named *main.tf* and write information about the existing bucket to the file:

```
[root@test terraform-test]#vim main.tf
resource "alicloud_oss_bucket" "bucket" {
  bucket = "test-hangzhou-2025"
  acl = "private"
}
```

Run the following command to import the *test-hangzhou-2025* bucket:

```
terraform import alicloud_oss_bucket.bucket test-hangzhou-2025
```

## References

- For more information about how to configure buckets by using Terraform, visit [alicloud\\_oss\\_bucket](#).
- For more information about how to configure objects by using Terraform, visit [alicloud\\_oss\\_bucket\\_object](#).

# 10. Others

## 10.1. Use Function Compute to download multiple objects as a package

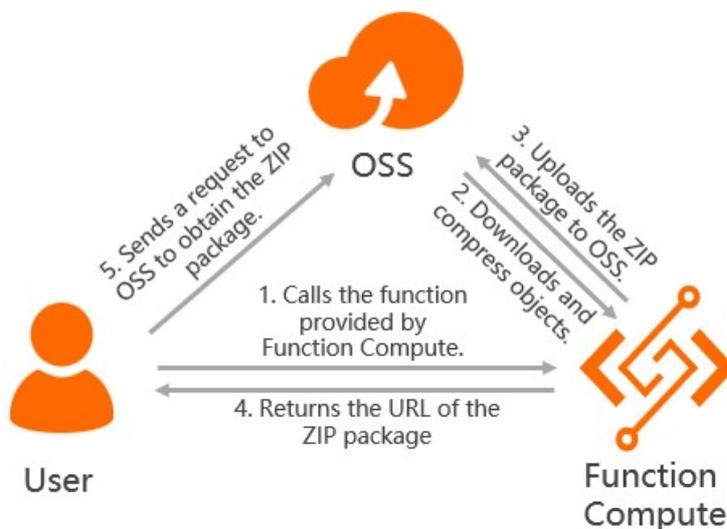
This topic describes how to use Function Compute to download multiple objects as a package to an on-premises device.

### Prerequisites

- Function Compute is activated.  
You can activate Function Compute on the [Function Compute](#) page.
- Function Compute is granted permissions to access Object Storage Service (OSS).  
For more information, see [Grant Function Compute permissions to access other Alibaba Cloud services](#).
- Multiple objects are uploaded to the specified directory of a bucket. In this example, multiple objects are uploaded to the `dir` directory of the `examplebucket` bucket in the China (Hangzhou) region. For more information, see [Simple upload](#).
- A RAM role is created for the service to which the application belongs in Function Compute, and the Alibaba Cloud Resource Name (ARN) of the RAM role is recorded. For more information, see [Create a RAM role for a trusted Alibaba Cloud account](#).

### How Function Compute works

When you download multiple objects from OSS at a time, the download speed may be affected if the sizes of the objects are small and the number of the objects is large. To resolve this issue, use Function Compute to compress the objects into a package. Then, download the package to your on-premises device and extract the package. The following figure shows how to use Function Compute to compress objects into a package and download the package.



1. A user invokes a function and specifies a bucket and the objects to be compressed.

2. Function Compute obtains the specified objects from OSS and generates a ZIP package that has a random name.
3. Function Compute uploads the ZIP package to OSS.
4. Function Compute returns the URL that is used to download the ZIP package to the user.
5. The user uses the returned URL to download the ZIP package from OSS.

## Usage notes

- The disk space used to run the function is limited. Streaming download and streaming upload are used to transfer the ZIP package. Only a small amount of data is cached in the memory.
- To accelerate the transfer of the ZIP package, Function Compute uploads the ZIP package to OSS while Function Compute generates the ZIP package.
- When the ZIP package is uploaded to OSS, multipart upload is used to upload parts in multiple threads in parallel.
- Function Compute takes up to 10 minutes to compress objects into a package. Test data: Function Compute takes approximately 63s to upload 57 objects whose total size is 1.06 GB.

## Procedure

The following example uses Function Compute to download multiple objects in the `dir/` directory of the `examplebucket` bucket in the China (Hangzhou) region to your on-premises device.

1. Use Serverless Devs to deploy applications.
  - i. Install Serverless Devs.

```
curl -o- -L http://cli.so/install.sh | bash
```

Check whether Serverless Devs is installed.

```
s -v
```

The following code is returned if Serverless Devs is installed:

```
@serverless-devs/s: 2.1.1, s-home: /root/.s, linux-x64, node-v14.19.3
```

- ii. Configure Serverless Devs.

```
s config add --AccessKeyID LTAI5t7h6SgiLSganP2m**** --AccessKeySecret KZo149BD9GLP  
NiDIEmdQ7dyNKG**** --access fc-access
```

The following table describes the parameters in Serverless Devs.

Parameter	Description
<code>--AccessKeyID</code>	The AccessKey ID used to access Function Compute.
<code>--AccessKeySecret</code>	The AccessKey secret used to access Function Compute.
<code>--Access</code>	The custom key alias.

## iii. Deploy the application template start-zip-oss.

## a. Initialize a project.

```
s init start-zip-oss -d start-zip-oss
```

## b. Refer to the following example values to complete configurations for start-zip-oss.

Item	Example value
The ID of the region where the application is to be deployed	cn-hangzhou
The service to which the application belongs in Function Compute	start-zip-oss
The function name of the application	zip-oss-func
The ARN of the RAM role that is created for the service to which the application belongs in Function Compute	acs:ram::137918634953****:role/test-role

## c. Start the project.

```
cd start-zip-oss
```

## d. Deploy the project.

```
s deploy -y
```

## e. Record the returned system\_url.

```
fc-zip-oss-service:
  region:  cn-hangzhou
  service:
    name: start-zip-oss
  function:
    name:      zip-oss-func
    runtime:  python2.7
    handler:  main.main_handler
    memorySize: 3072
    timeout:  1800
  url:
    system_url: https://zip-oss-func-start-zip-oss-ayouye****.cn-hangzhou.fcapp
.run
  triggers:
    -
      type: http
      name: http-test
```

## 2. Generate the event.json file and specify the bucket name and the directory where the objects to be downloaded are stored.

In this example, Linux is used.

```
cat > event.json <<EOF
{
  "bucket": "examplebucket",
  "source-dir": "dir/"
}
EOF
```

### 3. Trigger the function.

```
curl -v -L -o /test/oss.zip -d @./event.json https://zip-oss-func-zip-oss-ayouye****.cn-hangzhou.fcapp.run
```

- o `/test/oss.zip`: the local path used to store objects in the `dir/` directory.
- o `https://zip-oss-func-zip-oss-ayouye****.cn-hangzhou.fcapp.run`: `system_url` returned when the project is deployed.

 **Note** If you want to change the anonymous non-authenticated HTTP function in the preceding example to the authenticated HTTP function, you can use one of the following OSS SDKs:

- o [OSS SDK for Java](#)
- o [OSS SDK for Python](#)
- o [OSS SDK for Node.js](#)

## 10.2. Use CloudMonitor to monitor OSS throttling information in real time

If the number of requests sent by users exceeds a threshold that you configure for Object Storage Service (OSS), OSS throttling is triggered. After OSS throttling is triggered, users may fail to send more requests. To address this issue, you need to only complete simple configurations in the CloudMonitor console. This way, you can monitor requests sent to OSS in real time and receive notifications immediately after throttling is triggered.

### Context

OSS provides throttling at the user and bucket levels such as bandwidth throttling and queries per second (QPS) throttling. When the QPS or bandwidth usage of requests sent to OSS exceeds the threshold that you configure for OSS, access to OSS is throttled and the access speed is limited. If bandwidth throttling is triggered, latencies of access to OSS increase. If QPS throttling is triggered, OSS discards specific requests. For more information about bandwidth throttling and QPS throttling, see [Limits](#).

In the CloudMonitor console, you can create an alert rule for OSS throttling events and configure contact groups to receive notifications by text message, email, and DingTalk chatbot if OSS QPS throttling or bandwidth throttling is triggered, or alert thresholds are reached.

### Prerequisites

A contact group is created to receive notifications immediately after throttling is triggered. Contacts are added to the contact group. For more information, see [Create an alert contact or alert contact group](#).

## Create an alert rule

- 1.
2. In the left-side navigation pane, choose **Event Monitoring > System Event**.
3. Click the **Event Alert** tab.
4. Click **Create Alert Rule**.
5. In the **Create / Modify Event Alert** panel, configure the following parameters. Retain default values for other parameters. Then, click **OK**.

Parameter	Description
Alert Rule Name	Set this parameter to <i>rule1</i> .
Product Type	Select <b>Object Storage Service</b> .
Event Type	Select <b>All Events</b> .
Event Level	Select <b>WARN</b> and <b>INFO</b> .
Event Name	Select <b>All Events</b> . For more information about OSS throttling events supported by CloudMonitor, see <a href="#">OSS</a> .
Contact Group	Select <b>Alert Notification</b> for Notification Method. Then, select the created contract group.
Notification Method	Select <b>Warning (Message+Email ID+DingTalk Robot)</b> .

After the alert rule is configured, if OSS throttling is triggered or alert thresholds are reached, CloudMonitor automatically sends notifications to the specified contacts. A notification includes information such as the alert resource, event name, event type, and event details. For more information about alert notifications, see [Alert notification](#).

 **Notice** The system checks whether the monitored items reach the throttling threshold at an interval of 1 minute. If the duration in which the monitored items exceed the throttling threshold is equal to or larger than 30 seconds within an interval, the system sends a notification to the specified contacts. The system checks whether the monitored items reach the alert threshold at an interval of 10 minutes. If the duration in which the monitored items exceed the throttling threshold is equal to or larger than 1 second within an interval, the system sends a notification to the specified contacts.

## Alert notification

If the specified contacts receive an alert notification, refer to the following tables for detailed information about the notification and the event name, cause, impact, and solution.

 **Notice** If you want to view the traffic usage of all buckets that belong to the current user after you receive a user-level alert notification, create an OSS monitoring dashboard in advance. For more information about how to create an OSS monitoring dashboard, see [Create a system preset dashboard](#).

## Event names in alert notifications

 **Note** The alert thresholds in the following table are calculated based on the formula: Alert threshold = Throttling threshold × 0.8.

Event name	Cause	Impact	Solution
BucketIngressBandwidthThresholdExceeded	The total upstream bandwidth consumed by the current bucket exceeds the throttling threshold.	Requests sent to upload objects are throttled, and latencies increase.	Reduce the number of concurrent upload requests.
BucketEgressBandwidthThresholdExceeded	The total downstream bandwidth consumed by the current bucket exceeds the throttling threshold.	Requests sent to download objects are throttled, and latencies increase.	Reduce the number of concurrent download requests.
BucketQpsThresholdExceeded	The total number of requests received by the current bucket per second exceeds the throttling threshold.	OSS does not respond to specific requests and returns HTTP status code 503.	Reduce the number of requests sent per second.
UserIngressBandwidthThresholdExceeded	The total upstream bandwidth consumed by all buckets of the current user exceeds the throttling threshold.	Requests sent to upload objects are throttled, and latencies increase.	Reduce the number of concurrent upload requests.
UserEgressBandwidthThresholdExceeded	The total downstream bandwidth consumed by all buckets of the current user exceeds the throttling threshold.	Requests sent to download objects are throttled, and latencies increase.	Reduce the number of concurrent download requests.
UserQpsThresholdExceeded	The total number of requests received by all buckets of the current user per second exceeds the throttling threshold.	OSS does not respond to specific requests.	Reduce the number of requests sent per second.

Event name	Cause	impact	Solution
BucketImageCpuThresholdExceeded	The number of CPU cores consumed by the current bucket to process Image Processing (IMG) requests exceeds the throttling threshold.	Latencies increase after requests are sent to implement IMG.	Reduce the number of concurrent requests sent to implement IMG.
UserImageCpuThresholdExceeded	The number of CPU cores consumed by all buckets of the current user to process IMG requests exceeds the throttling threshold.	Latencies increase after requests are sent to implement IMG.	Reduce the number of concurrent requests sent to implement IMG.
BucketMirrorIngressBandwidthThresholdExceeded	The bandwidth consumed by the current bucket to send mirroring-based back-to-origin requests exceeds the throttling threshold.	Latencies increase after mirroring-based back-to-origin requests are sent.	Reduce the number of concurrent mirroring-based back-to-origin requests.
BucketMirrorQpsThresholdExceeded	The total number of requests sent by the current bucket per second to implement mirroring-based back-to-origin exceeds the throttling threshold.	OSS does not respond to specific mirroring-based back-to-origin requests.	Reduce the number of mirroring-based back-to-origin requests sent per second.
UserMirrorIngressBandwidthThresholdExceeded	The total traffic consumed by all buckets of the current user to send mirroring-based back-to-origin requests exceeds the throttling threshold.	Latencies increase after upload requests are sent to implement mirroring-based back-to-origin.	Reduce the number of concurrent mirroring-based back-to-origin requests.
UserMirrorQpsThresholdExceeded	The total number of requests sent by all buckets of the current user per second to implement mirroring-based back-to-origin exceeds the throttling threshold.	OSS does not respond to specific mirroring-based back-to-origin requests.	Reduce the number of mirroring-based back-to-origin requests sent per second.
BucketIngressBandwidth	The total upstream bandwidth consumed by the current bucket exceeds the alert threshold.	Latencies increase after upstream requests are sent to the bucket.	Reduce the number of concurrent upstream requests.

Event name	Cause	impact	Solution
BucketEgressBandwidth	The total downstream bandwidth consumed by the current bucket exceeds the alert threshold.	Latencies increase after downstream requests are sent to the bucket.	Reduce the number of concurrent downstream requests.
UserIngressBandwidth	The total upstream bandwidth consumed by all buckets of the current user exceeds the alert threshold.	Latencies increase after upstream requests are sent to the bucket.	Reduce the number of concurrent upstream requests.
UserEgressBandwidth	The total downstream bandwidth consumed by all buckets of the current user exceeds the alert threshold.	Latencies increase after downstream requests are sent to the bucket.	Reduce the number of concurrent downstream requests.

## Detailed information about alert notifications

Parameter	Description	Example
AvgSeverity	The degree of throttling. The greater the value is, the higher the latency becomes. Valid values: 0 to 100.	10
QoSType	The type of the throttling that is triggered. Valid values: <ul style="list-style-type: none"> <li><i>IngressBandwidth</i>: the upstream bandwidth.</li> <li><i>EgressBandWidth</i>: the downstream bandwidth.</li> <li><i>Qps</i>: the number of requests per second.</li> </ul>	<i>IngressBandwidth</i>
TrafficSource	The source of the traffic that triggers the throttling. Valid values: <ul style="list-style-type: none"> <li><i>intranet</i>: requests that are sent over the internal network.</li> <li><i>extranet</i>: requests that are sent over the Internet.</li> <li><i>net_all</i>: requests that are sent over the internal network and the Internet.</li> </ul>	<i>net_all</i>

## 10.3. OSS performance and scalability best practices

If you upload a large number of objects that use sequential prefixes such as timesteps and letters in the object names, multiple object indexes may be stored in a single partition. If an excessive number of requests are sent to query these objects, the response time is increased. To resolve this issue, we recommend that you add random prefixes to the names of objects.

### Background information

OSS partitions objects based on the object names that are encoded by using UTF-8. This way, a large number of objects can be processed and the response time is reduced for the requests. OSS supports only up to 2,000 queries per second (QPS) in the sequential read and write mode. If you use sequential prefixes such as timesteps and letters in object names when you upload a large number of objects, multiple object indexes may be stored in a single partition. For more information about the QPS of a single account, see [Limits](#).

If you use sequential prefixes in object names when you upload a large number of objects and you call operations such as GET, PUT, DELETE, COPY, and HEAD operations more than 2,000 times per second, perform batch operations such as batch delete operations that include more than 2,000 operations per second, or list more than 2,000 objects per second, the following issues occur:

- The partition becomes a hotspot. The I/O capacity is exhausted, or the system automatically limits the request rate.
- OSS partitions the data to balance the distribution of the data among partitions again and reduce the number of hotspots. This process may increase the request processing time.

 **Note** This operation is performed based on the analysis result of system status and processing capability. Objects that use sequential prefixes in object names may be stored in hotspots after the preceding operation is performed.

To resolve these issues, you can change the sequential prefixes in the object names to random prefixes for evenly distribution of object indexes and I/O loads among different partitions.

### Solution

You can use the following methods to change sequential prefixes in object names to random prefixes:

- Specify a hexadecimal hash as the prefix in an object name

If you use dates and customer IDs to generate object names, sequential prefixes that use timesteps are included in object names as shown in the following examples:

```
sample-bucket-01/2017-11-11/customer-1/file1
sample-bucket-01/2017-11-11/customer-2/file2
sample-bucket-01/2017-11-11/customer-3/file3
...
sample-bucket-01/2017-11-12/customer-2/file4
sample-bucket-01/2017-11-12/customer-5/file5
sample-bucket-01/2017-11-12/customer-7/file6
...
```

In this case, you can use the MD5 hash of multiple characters of the customer ID as the object name prefix. If you use the MD5 hash of four characters of the customer ID as the object name prefix, the names of the objects are generated as shown in the following examples:

```
sample-bucket-01/9b11/2017-11-11/customer-1/file1
sample-bucket-01/9fc2/2017-11-11/customer-2/file2
sample-bucket-01/d1b3/2017-11-11/customer-3/file3
...
sample-bucket-01/9fc2/2017-11-12/customer-2/file4
sample-bucket-01/f1ed/2017-11-12/customer-5/file5
sample-bucket-01/0ddc/2017-11-12/customer-7/file6
...
```

If you use the hexadecimal hash of four characters of the customer ID as the object name prefix, each character can be one of the 16 values (0-9, a-f). This way, the total number of combinations of the four characters is 65,536 ( $16^4$ ). In the storage system, the data can be distributed to up to 65,536 partitions. You can perform up to 2,000 operations per second on each partition. You can use the request rate to determine whether the number of buckets in a hash table meets your business requirements.

If you want to view the objects from the sample-bucket-01 bucket whose names contain a specified date, such as 2017-11-11, you need to call an operation to display all objects from sample-bucket-01. This way, you need to call the ListObject operation multiple times to obtain all objects in sample-bucket-01 and display the objects whose names contain the specified date.

- Reverse the order of digits that indicate milliseconds in object names

If you use the UNIX timestamps that are accurate to milliseconds to generate object names, sequential prefixes are included in object names as shown in the following examples:

```
sample-bucket-02/1513160001245.log
sample-bucket-02/1513160001722.log
sample-bucket-02/1513160001836.log
sample-bucket-02/1513160001956.log
...
sample-bucket-02/1513160002153.log
sample-bucket-02/1513160002556.log
sample-bucket-02/1513160002859.log
...
```

In this case, you can reverse the order of the digits in the UNIX timestamp. This way, the object names do not contain sequential prefixes. After you reverse the order of the digits, the object names are displayed as shown in the following examples:

```
sample-bucket-02/5421000613151.log
sample-bucket-02/2271000613151.log
sample-bucket-02/6381000613151.log
sample-bucket-02/6591000613151.log
...
sample-bucket-02/3512000613151.log
sample-bucket-02/6552000613151.log
sample-bucket-02/9582000613151.log
...
```

The first three digits indicate milliseconds and 1,000 values are available. The fourth digit changes at an interval of 1 second. The fifth digit changes at an interval of 10 seconds. The reverse operation increases the randomness of prefixes. This way, requests are evenly distributed among each partition to avoid performance bottleneck issues.