Alibaba Cloud Object Storage Service

Developer Guide

Issue: 20190420

Legal disclaimer

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

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
`Italics`	It is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *`[-all|-t]`*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Usage instructions

If it is your first time using Alibaba Cloud OSS, see the *OSS Quick Start Guide* to quickly get started with OSS.

The following table lists the manuals and guides that help you fully utilize OSS:

Resource	Description
Developer Guide	Describes the core concepts, functions, and provides methods (through console, API, or SDK) of using these functions.
Best Practices	Describes the application scenarios and configuration practices of OSS.
SDK Reference	Describes the SDK development, related parameters, and code samples based on major languages.
API Reference	Describes the RESTful API operations supported by OSS and provides related examples.
Console User Guide	Describes all operations supported by the OSS console.
Image Processing Guide	Describes various functions provided by Image Processing, such as format conversion, cropping, scaling, rotation, watermarks, and style encapsulation.
OSS migration tool	Describes the official migration tool that helps you migrate data from your local or third-party storage service to OSS.

2 Basic concepts

Before you use OSS, we recommend that you have a basic understanding of the following concepts.

Bucket

A bucket is a container for objects stored in OSS. Every object is contained in a bucket. The data model structure of Alibaba Cloud OSS is flat instead of hierarchical.

- All objects (files) are directly related to their corresponding buckets. Therefore, OSS lacks the hierarchical structure of directories and subfolders as in a file system.
- A user can have multiple buckets.
- A bucket name must be globally unique within OSS and cannot be changed once a bucket is created.
- A bucket can contain an unlimited number of objects.

The naming conventions for buckets are as follows:

- The bucket names must contain only lower case letters, numbers, and hyphens (-).
- The bucket names must start and end with a lower-case letter or number.
- The bucket names must be at least 3 bytes and no more than 63 bytes in length.

Object

Objects, also known as files, are the fundamental entities stored in OSS. An object is composed of metadata, data, and key. The key is the unique object name in a bucket. Metadata defines the attributes of an object, such as the time last modified and the object size. You can also specify custom metadata of an object.

The lifecycle of an object starts when it is uploaded, and ends when it is deleted. During the lifecycle, the object content cannot be changed. If you want to modify an object, you must upload a new object with the same name as the existing one to replace it. Therefore, unlike the file system, OSS does not allow users to modify objects directly.

OSS provides the Append Upload function, which allows you to continually append data to the end of an object.

The naming conventions for objects are as follows:

- The object names must use UTF-8 encoding.
- The object names must be at least 1 byte and no more than 1023 bytes.
- The object names cannot start with a backslash (\ ) or a forward slash (/).

 Note:
Object names are case sensitive. Unless otherwise stated, objects and files mentioned in OSS documents are collectively called objects.

Region

A region represents the physical location of an OSS data center. You can choose the region where OSS will store the buckets you create. You may choose a region to optimize latency, minimize costs, or address regulatory requirements. Generally, the closer the user is in proximity to a region, the faster the access speed is. For more information, see *OSS regions and endpoints*.

Regions are configured at bucket level instead of object level. Therefore, all objects contained in a bucket are stored in the same region. A region is specified when a bucket is created, and cannot be changed once it is created.

Endpoint

An endpoint is the domain name used to access the OSS. OSS provides external services through HTTP RESTful APIs. Different regions use different endpoints. For the same region, access through an intranet or through the Internet also uses different endpoints. For example, regarding the Hangzhou region, its Internet endpoint is oss-cn-hangzhou.aliyuncs.com, and its intranet endpoint is oss-cn-hangzhou-internal.aliyuncs.com. For more information, see *OSS regions and endpoints*.

AccessKey

An AccessKey (AK) is composed of an AccessKeyId and an AccessKeySecret. They work in pairs to perform access identity verification. OSS verifies the identity of a request sender by using the AccessKeyId/AccessKeySecret symmetric encryption method. The AccessKeyId is used to identify a user. The AccessKeySecret is used for the user to encrypt the signature and for OSS to verify the signature. The AccessKeySecret must be kept confidential. In OSS, AccessKeys are generated by the following three methods:

- The bucket owner applies for AccessKeys.

- The bucket owner uses RAM to authorize a third party to apply for AccessKeys.
- The bucket owner uses STS to authorize a third party to apply for AccessKeys.

For more information about AccessKeys, see *Access control*.

Strong consistency

In OSS, object operations are atomic, which means operations are either successful or failed without an intermediate state. OSS will never write corrupted or partial data.

Object operations in OSS are strongly consistent. For example, once a user receives an upload (PUT) success response, the object can be read immediately, and the data has already been written in triplicate. Therefore, OSS provides strong consistency for read-after-write. The same is true for the delete operations. Once a user deletes an object, the object becomes nonexistent immediately.

Data redundancy mechanism

OSS uses a data redundancy storage mechanism to store redundant data of each object on multiple devices of different facilities in the same area, ensuring data reliability and availability in case of hardware failure.

- Object operations in OSS are strongly consistent. For example, once a user receives an upload or copy success response, the object can be read immediately, and the redundant data has already been written to multiple devices.
- To ensure complete data transmission, OSS checks whether an error occurs when packets are transmitted between the client and the server by calculating the checksum of the network traffic packets.
- The redundant storage mechanism of OSS can avoid data loss if two storage facilities are damaged at the same time.
  - After data is stored in OSS, OSS checks whether redundant data is lost. If yes, OSS recovers the lost redundant data to ensure data reliability and availability.
  - OSS periodically checks the integrity of data through verification to discover data damage caused by factors such as hardware failure. If data is partially damaged or lost, OSS reconstructs and repairs the damaged data by using redundant data.

Comparison between OSS and file systems

Comparison item	OSS	File system
Data model	OSS is a distributed object storage service that uses a key-value pair format.	The file system is a hierarchical tree structure of directories that contain files.
Data retrieval	Objects are retrieved based on unique object names (keys). Although users can use names like test1/test.jpg , this does not indicate that the object test.jpg is saved in a directory named test1. For OSS, test1 /test.jpg and a.jpg have no essential difference . Similar amounts of resources are consumed during access to objects of different names.	Files are retrieved based on their locations in directories.
Advantage	OSS supports massive concurrent accesses , which means large volumes of unstructur ed data (such as images, videos, and documents) can be stored and retrieved without excessive use of resources.	Folder operations such as renaming, moving, and deleting directories are quite easy, because data does not need to be copied and replaced.
Disadvantage	The stored objects cannot be modified directly. If you want to modify an object, you must upload the new object of the same name to replace the existing one.	System performance depends on the capacity of a single device. The more files and directorie s that are created in the file system, the more resources are consumed, and the lengthier the user process becomes.

As a result, mapping OSS to a file system is not a recommended practice. When you use OSS, we recommend that you make full use of its advantages, including its

massive data processing capabilities to store massive volumes of unstructured data, such as images, videos, and documents.

The mapping between OSS concepts and file system concepts is as follows:

OSS	File system
Object	File
Bucket	Home directory
Region	NA
Endpoint	NA
AccessKey	NA
NA	Multilevel directory
GetService	Retrieving the list of home directories
GetBucket	Retrieving the list of files
PutObject	Writing a file
AppendObject	Appending data to an existing file
GetObject	Reading a file
DeleteObject	Deleting an object
NA	Modifying file content
CopyObject (same target and source)	Modifying file attributes
CopyObject	Copying a file
NA	Renaming a file

3 Endpoint

3.1 Endpoints

Composition rules for domain names

In the network requests for OSS, except those for the GetService API, the domain names are the third-level domain names with specified bucket names.

The domain name contains a bucket name and an endpoint in the format of BucketName.Endpoint. An endpoint is an access domain name. OSS provides external services through HTTP RESTful APIs. Different regions use different domain names. A region has an Internet endpoint and an intranet endpoint. For example, the Internet endpoint of the region China East 1 is oss-cn-hangzhou.aliyuncs.com, and the intranet endpoint of the region China East 1 is oss-cn-hangzhou-internal.aliyuncs.com. For more information, see *Regions and endpoints*.

Access OSS through the Internet

You can always access OSS through the Internet. In the Internet, the inbound traffic (write) is free, and outbound traffic (read) is charged. For more information about outbound traffic charges, see *OSS Pricing*.

You can access OSS through the Internet by using either of the following methods:

- Method 1: Use the URL to access OSS resource. The URL is constructed as follows:

```
< Schema >://< Bucket >.< Internet  Endpoint >/< Object >,  where :
   Schema   is   HTTP   or   HTTPS .
   Bucket   is   your   OSS   storage   space .
   Endpoint   is   the   access   domain   name   for   the   region
   of   a   bucket .  Enter   the   Internet   endpoint   here .
```

```
   Object   is   a   file   uploaded   to   the   OSS .
```

For example, in the region China East 1, the object named myfile/aaa.txt is stored in the bucket abc. The Internet access address of the object is:

```
 abc . oss - cn - hangzhou . aliyuncs . com / myfile / aaa . txt
```

You can directly use the object URL in HTML as follows:

```
< img   src =" https :// abc . oss - cn - hangzhou . aliyuncs . com
 / mypng / aaa . png " />
```

- Method 2: Configure the Internet access domain name through OSS SDKs.

  You must set different endpoints when operating buckets of different regions.

  For example, before configuring buckets in the region China East 1, you must set the endpoint during class instantiation.

```
String   accessKeyI  d  = "< key >";
  String   accessKeyS  ecret  = "< secret >";
  String   endpoint  = " oss - cn - hangzhou . aliyuncs . com ";
  OSSClient   client  =  new   OSSClient ( endpoint ,  accessKeyI
d ,  accessKeyS  ecret );
```

Access OSS through an intranet

Intranet refers to the internal communication networks among Alibaba Cloud products. For example, you access OSS through ECS. In an intranet, the inbound and outbound traffic is free. If the ECS instance and the OSS bucket are in the same region , we recommend that you use an intranet to access OSS.

You can access OSS through an intranet by using either of the following methods:

- Method 1: Use the URL to access OSS resource. The URL is constructed as follows:

```
< Schema >://< Bucket >.< IntranetEn  dpoint >/< Object >,  where :
   Schema   is   HTTP   or   HTTPS .
   Bucket   is   your   OSS   storage   space .
   Endpoint   is   the   access   domain   name   for   the   region
   of   a   bucket .  Enter   the   intranet   endpoint   here .
```

```
   Object   is   a   file   uploaded   to   the   OSS .
```

For example, in the region China East 1, the object named myfile/aaa.txt is stored in the bucket abc. The intranet access address of the object is:

```
abc . oss - cn - hangzhou - internal . aliyuncs . com / myfile /
aaa . txt
```

- Method 2: Configure the intranet access domain name using OSS SDKs on ECS.

  For example, set the intranet endpoint for the Java SDK on ECS as follows:

```
String   accessKeyI  d  = "< key >";
  String   accessKeyS  ecret  = "< secret >";
  String   endpoint  = " oss - cn - hangzhou - internal . aliyuncs
. com ";
  OSSClient   client  =  new   OSSClient ( endpoint ,  accessKeyI
d ,  accessKeyS  ecret );
```

Note:

If you want to use an intranet to access OSS, the ECS instance and the OSS bucket must be in the same region.

For example, you have purchased ECS instances of China North 2 (Beijing), and you have two OSS buckets:

- One buckets is beijingres, and its region is China North 2 (Beijing). The intranet address `beijingres . oss - cn - beijing - internal . aliyuncs . com` can be used by ECS instances to access beijingres resources, and the traffic generated is free.
- The other bucket is qingdaores, and its region is China North 1 (Qingdao). The intranet address `qingdaores . oss - cn - qingdao - internal . aliyuncs . com` cannot be used by ECS instances to access qingdaores resources. The Internet address `qingdaores . oss - cn - qingdao . aliyuncs . com` must be used to access qingdaores resources, and the outbound traffic generated is charged.

3.2 Regions and endpoints

Regions indicate the regions where the data center of OSS is located. Endpoints indicate the domain names used by users to access OSS through Internet. This topic describes the mapping relationship between regions and endpoints.

Regions and OSS endpoints in classic networks

The following table describes the OSS Internet and intranet endpoints of each region in classic networks.

Region name	OSS region	Internet endpoint	Internet endpoint protocol	Intranet endpoint for ECS access	Intranet endpoint protocol
China East 1 (Hangzhou)	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	HTTP and HTTPS	oss-cn-hangzhou-internal.aliyuncs.com	HTTP and HTTPS
China East 2 (Shanghai)	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	HTTP and HTTPS	oss-cn-shanghai-internal.aliyuncs.com	HTTP and HTTPS
China North 1 (Qingdao)	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	HTTP and HTTPS	oss-cn-qingdao-internal.aliyuncs.com	HTTP and HTTPS
China North 2 (Beijing)	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	HTTP and HTTPS	oss-cn-beijing-internal.aliyuncs.com	HTTP and HTTPS
China North 3 (Zhangjiakou)	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	HTTP and HTTPS	oss-cn-zhangjiakou-internal.aliyuncs.com	HTTP and HTTPS

Region name	OSS region	Internet endpoint	Internet endpoint protocol	Intranet endpoint for ECS access	Intranet endpoint protocol
China North 5 (Hohhot)	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	HTTP and HTTPS	oss-cn-huhehaote-internal.aliyuncs.com	HTTP and HTTPS
China South 1 (Shenzhen)	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	HTTP and HTTPS	oss-cn-shenzhen-internal.aliyuncs.com	HTTP and HTTPS
Hong Kong	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	HTTP and HTTPS	oss-cn-hongkong-internal.aliyuncs.com	HTTP and HTTPS
US West 1 (Silicon Valley)	oss-us-west-1	oss-us-west-1.aliyuncs.com	HTTP and HTTPS	oss-us-west-1-internal.aliyuncs.com	HTTP and HTTPS
US East 1 (Virginia)	oss-us-east-1	oss-us-east-1.aliyuncs.com	HTTP and HTTPS	oss-us-east-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-2-internal.aliyuncs.com	HTTP and HTTPS

Region name	OSS region	Internet endpoint	Internet endpoint protocol	Intranet endpoint for ECS access	Intranet endpoint protocol
Asia Pacific SE 3 (Kuala Lumpur)	oss-ap-southeast-3	oss-ap-southeast-3.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-3-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 5 (Jakarta)	oss-ap-southeast-5	oss-ap-southeast-5.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-5-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific NE 1 (Tokyo)	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	HTTP and HTTPS	oss-ap-northeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SOU 1 (Mumbai)	oss-ap-south-1	oss-ap-south-1.aliyuncs.com	HTTP and HTTPS	oss-ap-south-1-internal.aliyuncs.com	HTTP and HTTPS
EU Central 1 (Frankfurt)	oss-eu-central-1	oss-eu-central-1.aliyuncs.com	HTTP and HTTPS	oss-eu-central-1-internal.aliyuncs.com	HTTP and HTTPS
UK (London)	oss-eu-west-1	oss-eu-west-1.aliyuncs.com	HTTP and HTTPS	oss-eu-west-1-internal.aliyuncs.com	HTTP and HTTPS
Middle East 1 (Dubai)	oss-me-east-1	oss-me-east-1.aliyuncs.com	HTTP and HTTPS	oss-me-east-1-internal.aliyuncs.com	HTTP and HTTPS

Note:

- We recommend that you use third-level domain names that are in `bucket name + endpoint` format to share links or bind custom domain names (CNAME). For

example, the third-level domain name for a bucket named oss-sample in China East 2 (Shanghai) is oss-sample.oss-cn-shanghai.aliyuncs.com.

- When using SDKs, use `http://` or `https://` + `endpoint` as the initialization parameter. For example, we recommend that you use `http://oss-cn-shanghai.aliyuncs.com` or `https://oss-cn-shanghai.aliyuncs.com` as the initialization parameter of an endpoint in China East 2 (Shanghai). Do not use a third-level domain name, that is, `http://bucket.oss-cn-shanghai.aliyuncs.com`, as the initialization parameter.
- By default, the Internet address "oss.aliyuncs.com" directs to the Internet endpoint of China East 1 (Hangzhou), and the intranet address "oss-internal.aliyuncs.com" directs to the intranet endpoint of China East 1 (Hangzhou).

Regions and OSS endpoints in VPC networks

ECS instances in VPC networks can use the following endpoints to access OSS.

Region name	OSS region	Endpoint in VPC networks	Protocol
China East 1 (Hangzhou)	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	HTTP and HTTPS
China East 2 (Shanghai)	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	HTTP and HTTPS
China North 1 (Qingdao)	oss-cn-qingdao	oss-cn-qingdao-internal.aliyuncs.com	HTTP and HTTPS
China North 2 (Beijing)	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	HTTP and HTTPS
China North 3 (Zhangjiakou)	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	HTTP and HTTPS
China North 5 (Hohhot)	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	HTTP and HTTPS

Region name	OSS region	Endpoint in VPC networks	Protocol
China South 1 (Shenzhen)	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	HTTP and HTTPS
Hong Kong	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	HTTP and HTTPS
US West 1 (Silicon Valley)	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	HTTP and HTTPS
US East 1 (Virginia)	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	oss-ap-southeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 3 (Kuala Lumpur)	oss-ap-southeast-3	oss-ap-southeast-3-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 5 (Jakarta)	oss-ap-southeast-5	oss-ap-southeast-5-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific NE 1 (Tokyo)	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SOU 1 (Mumbai)	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	HTTP and HTTPS
EU Central 1 (Frankfurt)	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	HTTP and HTTPS
UK (London)	oss-eu-west-1	oss-eu-west-1-internal.aliyuncs.com	HTTP and HTTPS

Region name	OSS region	Endpoint in VPC networks	Protocol
Middle East 1 (Dubai)	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	HTTP and HTTPS

Usage of endpoints

- For the composition rules of OSS endpoints and the methods of accessing OSS through the Internet and intranet, see *Endpoints*.
- If you are an ECS user and need to use an OSS intranet endpoint, see *How do ECS users use OSS intranet addresses?*

4 Storage classes

4.1 Introduction to storage classes

OSS provides three storage classes: Standard, Infrequent Access, and Archive. These storage classes cover various data storage scenarios from frequently accessed (hot) data to infrequently accessed (cold) data.

Standard

OSS Standard storage provides highly reliable, highly available, and high-performance object storage service that supports frequent data access. The high throughput and low latency of OSS make it well suited for storing social networking content such as images, audio, and video. It is also great for storing large unstructured data sets for use in big data analytics.

OSS Standard storage has the following features:

- Designed for 99.999999999% (11 nines) durability.
- Designed for 99.99% availability.
- High-throughput and low-latency access performance.
- Supports HTTPS.
- Supports Image Processing.

Infrequent Access

OSS Infrequent Access storage is suitable for storing long-lived, but less-frequently accessed data (once or twice per month). With a storage unit price lower than the Standard class, it is suitable for longer-term backup of various mobile apps, smart device data, and enterprise data. Objects of the Infrequent Access storage class have a minimum storage duration. Charges apply if you delete objects that have been stored for less than 30 days. Objects of the Infrequent Access storage class have a minimum billable size. Objects smaller than 64 KB are charged as 64 KB. Data retrieval incurs charges.

OSS Infrequent Access storage has the following features:

- Designed for 99.999999999% (11 nines) durability.
- Designed for 99.99% service availability.

- Supports real-time access.
- Supports HTTPS.
- Supports Image Processing.
- Specified minimum storage duration and minimum billable size.

Archive

OSS Archive storage has the lowest price among the three storage classes. It is suitable for storing archival data for a long time (more than half a year recommended), such as medical images, scientific materials, and video footages. The data is infrequently accessed during the storage period and it may take about one minute to restore the data to a readable state. Objects of the Archive storage class have a minimum storage duration. Charges apply if you delete objects that are stored for less than 60 days. Objects of the Archive storage class have a minimum billable size. Objects smaller than 64 KB are charged as 64 KB. Data retrieval incurs charges.

OSS Archive storage has the following features:

- Designed for 99.999999999% (11 nines) durability.
- Designed for 99.99% service availability (restored data).
- It takes one minute to restore the stored data from the frozen state to the readable state.
- Supports HTTPS.
- Supports Image Processing, but data needs to be restored first.
- Specified minimum storage duration and minimum billable size.

Comparison of storage classes

Item	Standard	Infrequent Access	Archive
Data durability	99.999999999%	99.999999999%	99.999999999%
Designed service availability	99.99%	99.99%	99.99% (restored data)
Minimum billed size of objects	Calculate by actual size of objects	64 KB	64 KB
Minimum storage duration	Not required	30 days	60 days
Data retrieval fee	No data retrieval fee	Charged by the size of retrieved data, in GB	Charged by the size of restored data, in GB

Item	Standard	Infrequent Access	Archive
Latency	Latency in ms	Latency in ms	It takes one minute to restore data from the frozen state to the readable state.
Images processing	Supported	Supported	Supported, but data needs to be restored first.

Note:

"Data" in "data retrieval fee" refers to the size of data read from the underlying distributed storage system. The data transferred over the public network is billed as part of the outbound traffic costs.

Supported APIs

API	Standard	Infrequent Access	Archive
Bucket creation, deletion, and query			
PutBucket	Supported	Supported	Supported
GetBucket	Supported	Supported	Supported
DeleteBucket	Supported	Supported	Supported
Bucket ACL			
PutBucketAcl	Supported	Supported	Supported
GetBucketAcl	Supported	Supported	Supported
Bucket logging			
PutBucketLogging	Supported	Supported	Supported
GetBucketLogging	Supported	Supported	Supported
Bucket default static page			
PutBucketWebsite	Supported	Supported	Not supported
GetBucketWebsite	Supported	Supported	Not supported
Bucket anti-leech protection			
PutBucketReferer	Supported	Supported	Supported
GetBucketReferer	Supported	Supported	Supported
Bucket lifecycle			

API	Standard	Infrequent Access	Archive
PutBucketLifecycle	Supported	Supported	Supported, data deletion only
GetBucketLifecycle	Supported	Supported	Supported
DeleteBucketLifecycle	Supported	Supported	Supported
Bucket Cross-Origin Replication			
PutBucketReplication	Supported	Supported	Supported
Bucket Cross-Origin Resource Sharing			
PutBucketcors	Supported	Supported	Supported
GetBucketcors	Supported	Supported	Supported
DeleteBucketcors	Supported	Supported	Supported
Object operations			
PutObject	Supported	Supported	Supported
PutObjectACL	Supported	Supported	Supported
GetObject	Supported	Supported	Supported, but data needs to be restored first
GetObjectACL	Supported	Supported	Supported
GetObjectMeta	Supported	Supported	Supported
HeadObject	Supported	Supported	Supported
CopyObject	Supported	Supported	Supported
OptionObject	Supported	Supported	Supported
DeleteObject	Supported	Supported	Supported
DeleteMultipleObjects	Supported	Supported	Supported
PostObject	Supported	Supported	Supported
PutSymlink	Supported	Supported	Supported
GetSymlink	Supported	Supported	Supported
RestoreObject	Not supported	Not supported	Supported
Multipart operations			

API	Standard	Infrequent Access	Archive
InitiateMultipartUpload	Supported	Supported	Supported
UploadPart	Supported	Supported	Supported
UploadPartCopy	Supported	Supported	Supported
CompleteMultipartUpload	Supported	Supported	Supported
AbortMultipartUpload	Supported	Supported	Supported
ListMultipartUpload	Supported	Supported	Supported
ListParts	Supported	Supported	Supported
Image Processing	Supported	Supported	Supported

4.2 Storage Classes Conversion

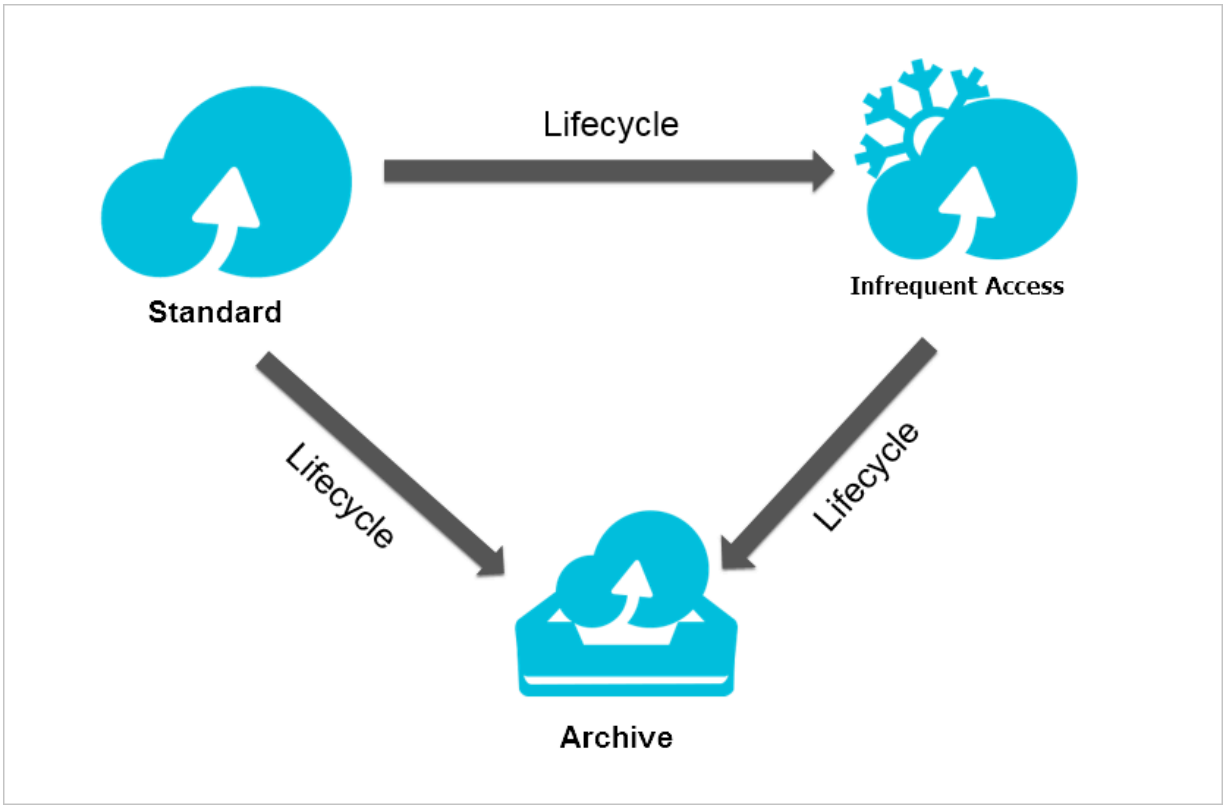
This topic describes how to convert the storage class of an object between Standard, IA, and Archive.

Lifecycle Object Transition

OSS supports three *storage classes*: Standard, Infrequent Access, and Archive.

The Object Transition mechanism is now available in OSS Lifecycle Management function in all regions across China. The following storage classes are supported for automatic conversion:

- Standard -> Infrequent Access
- Standard -> Archive
- Infrequent Access -> Archive

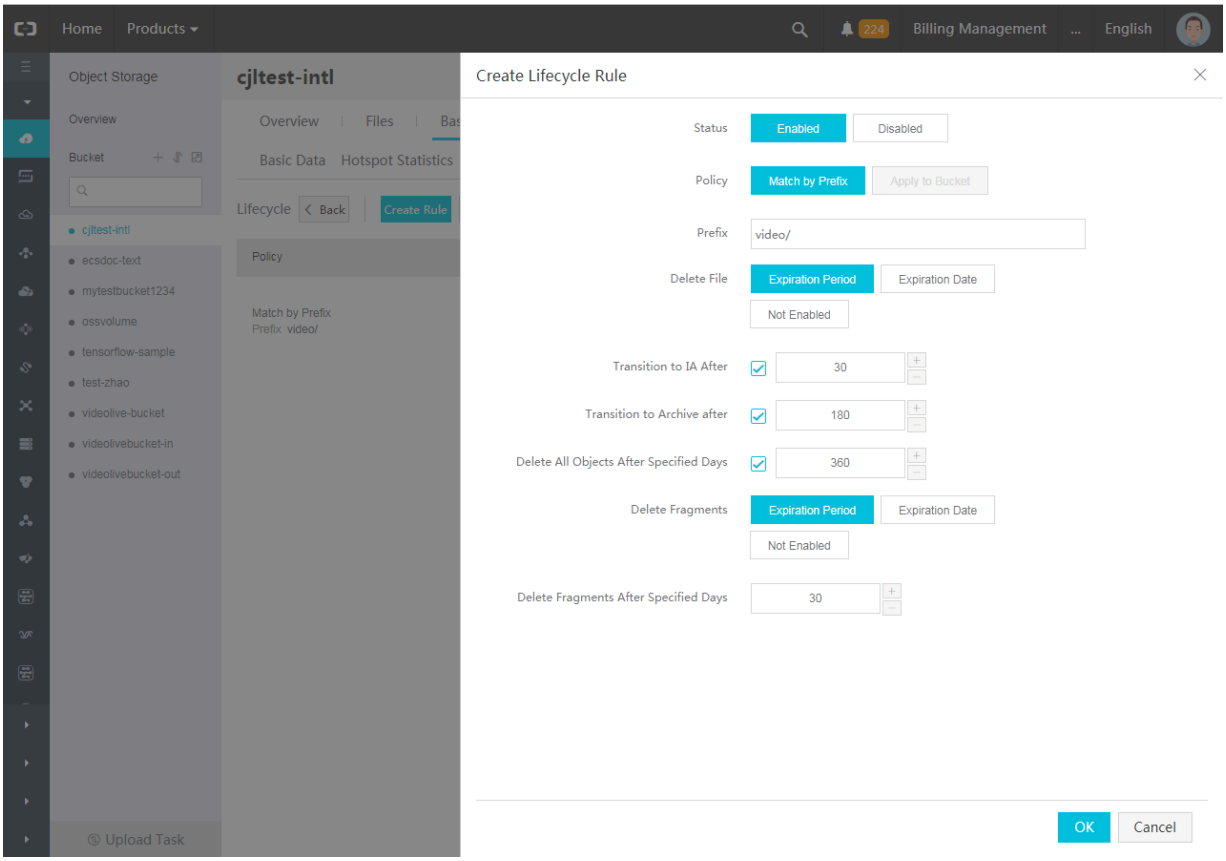


Examples

You can configure lifecycle policies for objects with a given prefix in one bucket as follows:

- They are converted to Infrequent Access class after being stored for 30 days.
- They are converted to Archive class after being stored for 180 days.
- They are deleted automatically after being stored for 360 days.

You can complete the configuration of the preceding lifecycle policies in the console. For more information, see *Set lifecycle*.



Note:

If the following three parameters are configured:

`Transition to IA After`, `Transition to Archive After`, and `Delete All Objects After Specified Days`, then the number of days set for each parameter must meet the following criteria:

Days for converting to Infrequent Access < Days for converting to Archive < Specified days for deleting

Notes

After the Object type conversion, the storage cost is calculated based on the unit price of converted storage class.

Notes for Infrequent Access and Archive storage types:

- Minimum billable size:

  Objects smaller than 128 KB are charged as 128 KB.

- Minimum storage period:

  The stored data is required to be saved for at least 30 days. Charges will be incurred if you delete files that are stored for less than 30 days.

- Restore time of Archive type:

  It takes one minute for Archive type Object to restore the data to a readable state. If real-time read is required in the business scenario, we recommend that you convert the file to the Infrequent Access storage class instead of Archive class. Otherwise, after converting the file to the Archive class, the data cannot be read in real time.

- Data access charges:

  Both Infrequent Access and Archive classes are required to pay data access charges as a separate charge item to outbound traffic. If the average access frequency per Object is higher than once per month, you are not advised to convert the data to Infrequent Access or Archive class.

Storage classes conversion in other ways

For conversions from Archive type to Standard class or Infrequent Access class, or from Infrequent Access class to Standard class, you can read the Object and rewrite it to the Bucket of corresponding storage class. The default storage class of Object is determined by the Bucket.

For example, for the conversion of Infrequent Access Object in the Bucket of Standard type to Standard Object, you can read and rewrite the Object. Based on the type of the Bucket, the newly-written Object is of Standard storage class.

For the Object that has been converted to Archive class, you can only read it after performing Restore operation and restore it to a readable state.

For more information, see *Create and use the Archive bucket*.

4.3 Create and use the Archive bucket

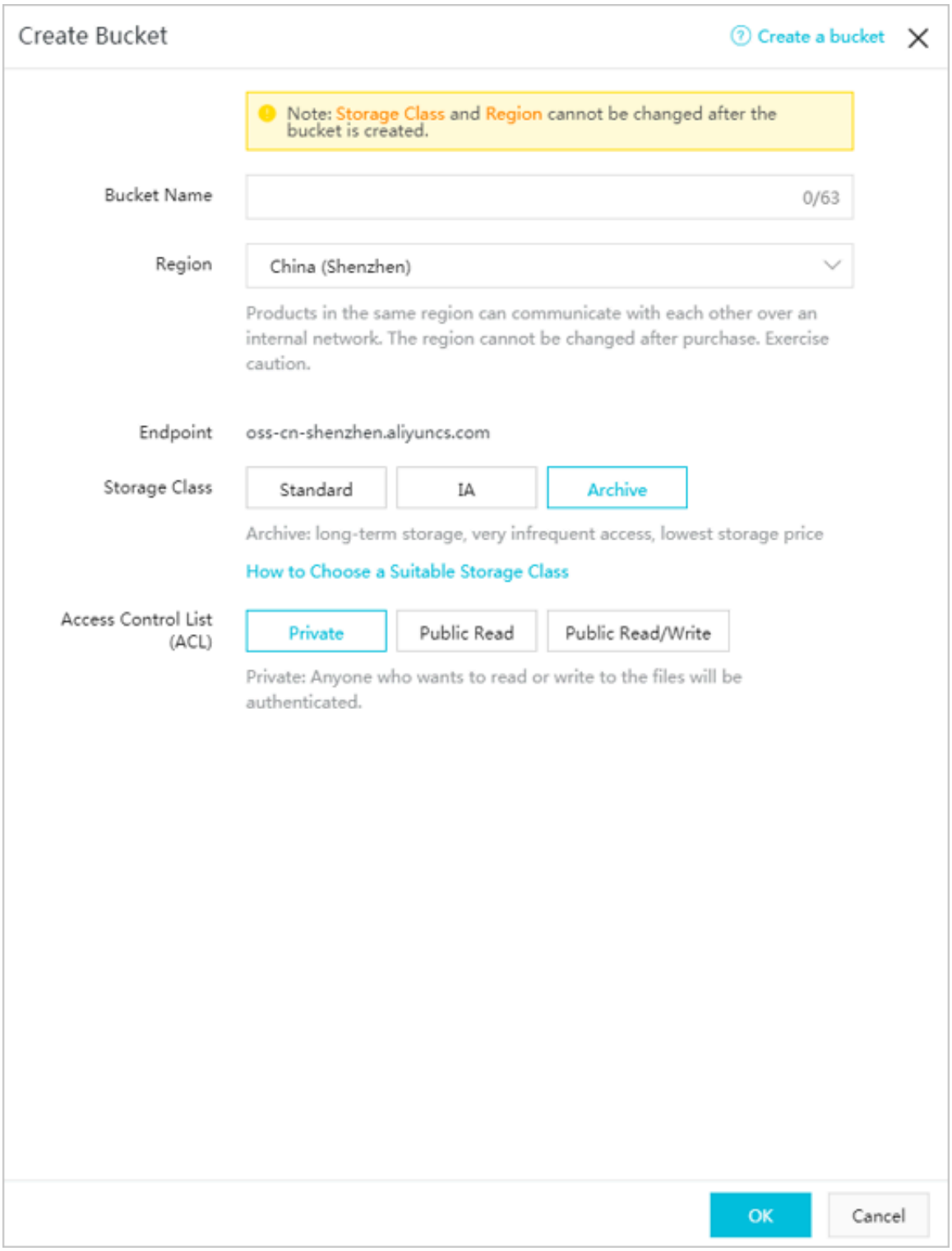
OSS provides three storage classes, among which the Archive storage class has the lowest price. However, you have to restore the archived data to a readable state if you want to access it. This topic describes how to create and use the bucket of the Archive storage class. For more information about the three storage classes, see *Introduction to storage classes*.

Create an Archive bucket

You can choose to create an Archive bucket by using the console, APIs/SDKs, or command line tools.

- **Create an Archive bucket by using the console**

  To create an Archive bucket in the console, select `Archive` **for** `Storage Class` **.**



- **Create an Archive bucket by using APIs/SDKs**

  Take the Java SDK for example:

```
 OSSClient   ossClient  =  new   OSSClient ( endpoint ,  accessKeyI
 d ,  accessKeyS  ecret );
 CreateBuck  etRequest   createBuck  etRequest = new   CreateBuck
 etRequest ( bucketName );
//  Set   the   bucket   ACL   to   public - read .  The   default
 ACL   policy   is   private - read - write .  createBuck  etRequest
 . setCannedA  CL ( CannedAcce  ssControlL  ist . PublicRead );
//  Set   the   storage   class   to   Archive .  The   default
 storage   class   is   Standard .
 createBuck  etRequest . setStorage  Class ( StorageCla  ss .
 Archive );
 ossClient . createBuck  et ( createBuck  etRequest );
```

  `createBuck etRequest . setStorage Class ( StorageCla ss . Archive );` **indicates that the storage class of the created bucket is Archive.**

- **Create an Archive bucket by using OSS command line tools**

  Take ossutil for example:

```
./ ossutil   mb   oss ://[ bucket   name ] -- storage - class =
 Archive
```

  Replace `[ bucket name ]` **with your own bucket. Set** `-- storage - class Archive` **.**

Use an Archive bucket

- **Upload data to an Archive bucket**

  Archive buckets support PutObject and MultipartUpload, but do not support AppendObject. The objects uploaded by PutObject and MultipartUpload can be directly stored in Archive buckets.

- **Download data from an Archive bucket**

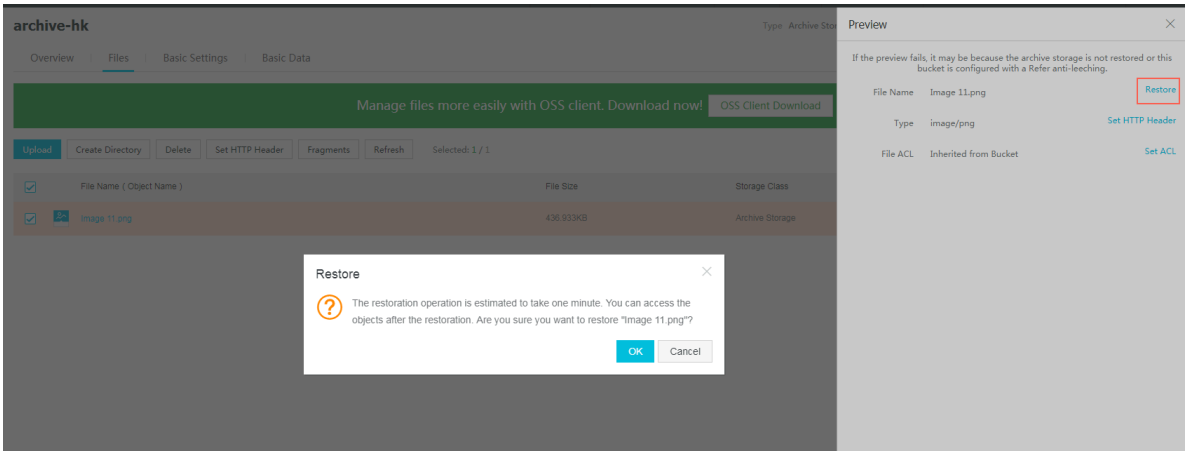
  Objects stored in Archive buckets are not accessible in real time. You must first initiate a restore request and then wait for one minute until the object is available.

  The restore process of an archived object is as follows:

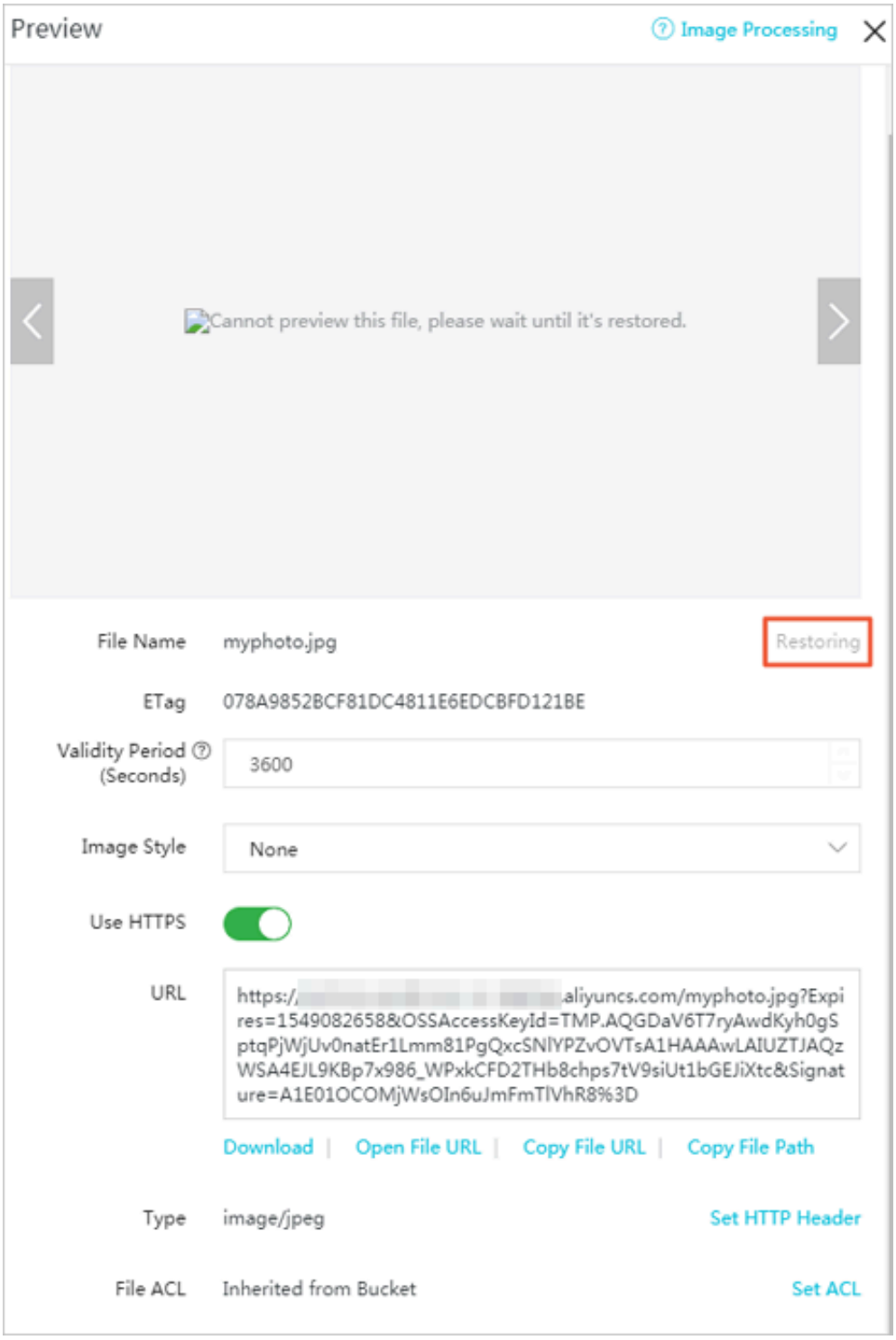
  1. **The archived object is in the frozen state at the beginning.**
  2. **After a restore request is submitted, the object enters the restoring state.**
  3. **One minute later, the object enters the restored state and can be read.**
  4. **The restored state lasts for one day by default, and can be prolonged to a maximum of seven days. Once this period ends, the object returns to the frozen state.**

You can choose to restore an archived object by using the console, APIs/SDKs, or command line tools.

- Restore an archived object by using the console



To restore an archived object in the console, enter the Preview page of the object, and then click Restore. It takes one minute to restore the object. During this process, the object is in the restoring state.



- **Restore an archived object by using APIs/SDKs**

  Take the Java SDK for example. Call the restoreObject method to restore an object:

```
 ObjectMeta  data   objectMeta  data  =  ossClient . getObjectM
 etadata ( bucketName ,  key );
//  check   whether   the   object   is   archive   class
 StorageCla  ss   storageCla  ss  =  objectMeta  data . getObjectS
 torageClas  s ();
 if  ( storageCla  ss  ==  StorageCla  ss . Archive ) {
    //  restore   object
```

```
     ossClient . restoreObj  ect ( bucketName ,  key );
    //  wait   for   restore   completed
     do  {
         Thread . sleep ( 1000 );
         objectMeta  data  =  ossClient . getObjectM  etadata (
 bucketName ,  key );
    }  while  (!  objectMeta  data . isRestoreC  ompleted ());

//  get   restored   object
 OSSObject   ossObject  =  ossClient . getObject ( bucketName ,  key
 );
 ossObject . getObjectC  ontent (). close ();
```

- **Restore an archived object by using OSS command line tools**

  Take ossutil for example:

  ```
  ./ ossutil   restore   oss ://[ Bucket   name ]/[ Object   name ]
  ```

  Replace `[ Bucket name ]` and `[ Object name ]` with your own bucket and object.

5 Access OSS

5.1 Quick start

This topic describes how to perform basic OSS operations, such as create a bucket, upload an object, and download an object.

1. Log on to the *OSS console*.
2. Create a bucket.
3. Upload and download files.

For more information, see *Get started with Alibaba Cloud OSS*.

Get familiar with OSS upload and download

Before you use OSS SDKs, we recommend that you have basic familiarity with the OSS upload and download methods.

OSS uses RESTful APIs to perform operations and all requests are standard HTTP requests.

OSS provides different upload methods to meet different requirements. You can:

- Use the Put Object method to upload a single file smaller than 5 GB to OSS. For more information, see *Simple upload*.
- Use the Post Object method (HTTP form) to upload a file smaller than 5 GB to OSS from a browser. For more information, see *Form upload*.
- Use the Multipart Upload method to upload a file larger than 5 GB. For more information, see *Multipart upload*.
- Use the Append Object method to directly append content to the end of an object. This method is particularly well suited for video monitoring and live video broadcasting. For more information, see *Append object*.

OSS also provides different download methods. For more information, see *Simple download* and *Multipart download*.

General process of using OSS SDKs

1. Obtain the AccessKeyId and AccessKeySecret from the console.
2. Download the OSS SDKs in your preferred programming language from GitHub.

3. Perform uploads, downloads, and other operations.

For more information about how to use the OSS SDKs for different programming languages, see *OSS SDK Reference*.

5.2 OSS-based app development

Development sequence diagram

Typical OSS-based app development involves the following four components:

- OSS: Provides functions such as upload, download, and upload callback.
- Developer's mobile client (mobile application or webpage application, called the client for short): Uses the service provided by the developer to access OSS.
- Application server: Interacts with the client. This server is used for the developer's service.
- Alibaba Cloud STS: Issues temporary credentials.

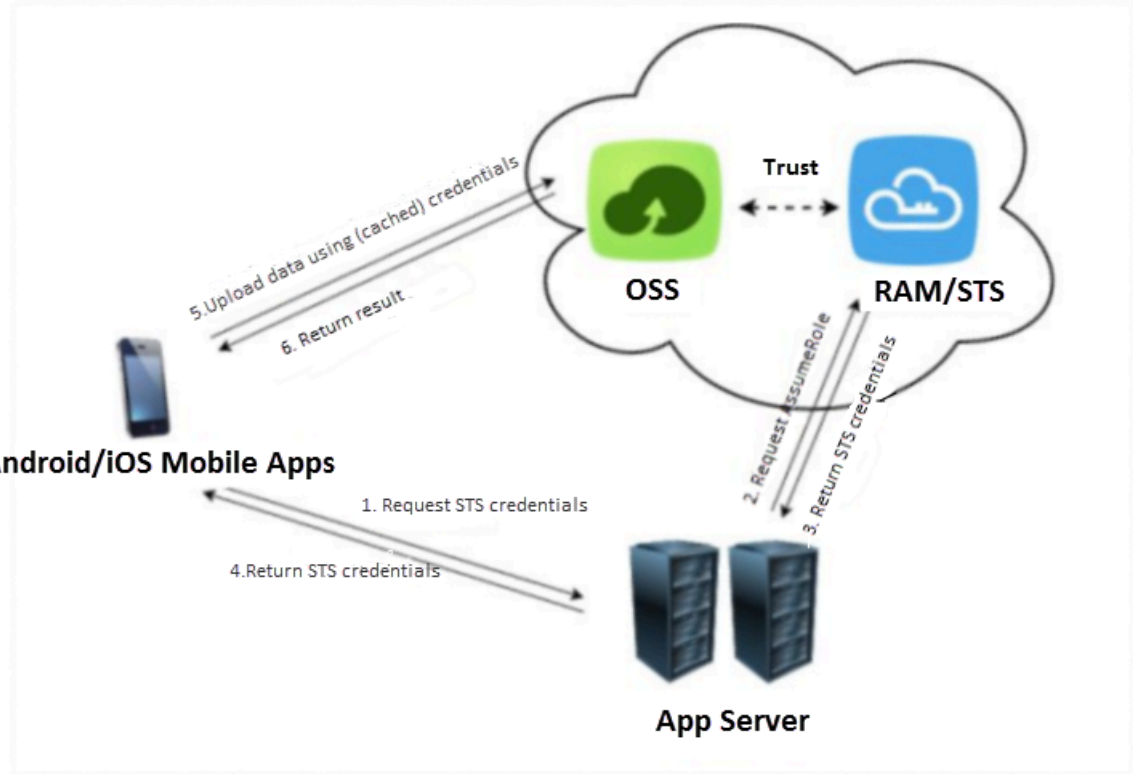
Best practices

- *Set up direct data transfer for mobile apps*
- *Set up data callback for mobile apps*
- *Permission control*

Service development process

- Data upload with temporary credential authorization

  The following figure shows the process of data upload with temporary credential authorization:



The description of the process is as follows:

1. The client sends the application server the request of uploading data to OSS.
2. The application server sends a request to STS.
3. STS returns temporary credentials (STS AccessKey and token) to the application server.
4. The client obtains the authorization (STS AccessKey and token) and calls the mobile client SDK to upload data to OSS.
5. The client successfully uploads data to OSS. If callback is not set, the process is complete. If callback is set, OSS calls the relevant interface.

Note that:

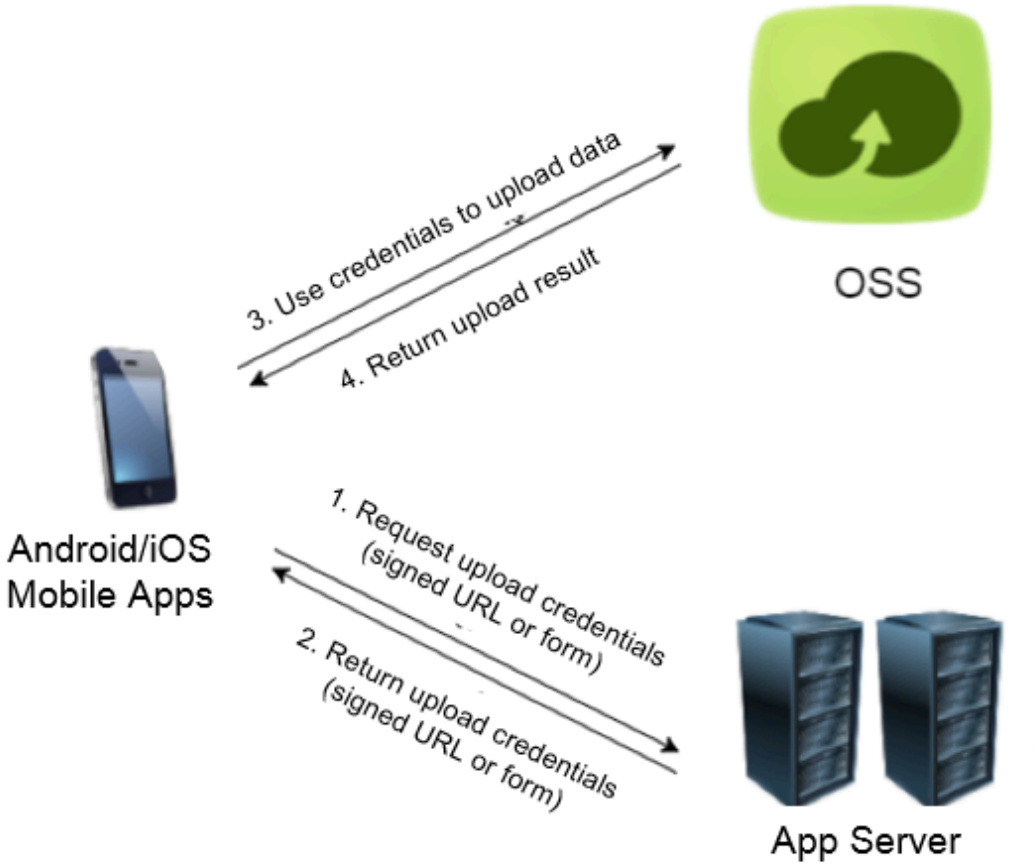
- The client does not have to request authorization from the application server for each upload attempt. Each time the authorization is obtained, the client caches temporary credentials returned by STS until it expires.

- **STS provides fine-grained access control for upload, which restricts client access permissions at the object level. This completely isolates the objects uploaded to OSS by different clients, and thus greatly enhances application security.**

For more information, see *Authorized third-party upload*.

- Data upload with signed URL or form authorization

  The following figure shows the process of data upload with signed URL or form authorization:



  The description of the process is as follows:

  1. The client sends the application server the request of uploading data to OSS.
  2. The application server returns credentials (signed URL or form) to the client.
  3. The client obtains authorization (signed URL or form) and calls the mobile client SDK to upload data or uses form upload to directly upload data to OSS.
  4. The client successfully uploads data to OSS. If callback is not set, the process is complete. If callback is set, OSS calls the relevant interface.

  For more information, see *Authorized third-party upload*.

- Data download with temporary credential authorization

  The process of data download with temporary credential authorization is similar to that of data download with temporary credential authorization:

  1. The client sends the application server the request of downloading data from OSS.
  2. The application server sends a request to STS and obtains temporary credentials (STS AccessKey and token).
  3. The application server returns the temporary credentials (STS AccessKey and token) to the client.
  4. The client obtains the authorization (STS AccessKey and token) and calls the mobile client SDK to download data from OSS.
  5. The client successfully downloads data from OSS.

  Note that:

  - For download, the client also caches temporary credentials to increase access speed.
  - STS also provides fine-grained access control for download. The access control for both upload and download helps to isolate the OSS storage space for each mobile client.

- Data download with signed URL authorization

  The process of signed URL authorization for download is similar to that of signed URL authorization for upload:

  1. The client sends the application server the request of downloading data from OSS.
  2. The application server returns the signed URL to the client.
  3. The client obtains authorization (signed URL) and calls the mobile client SDK to download data from OSS.
  4. The client successfully downloads data from OSS.

   Note:
  The client cannot store the developer's AccessKey. You can get only the URL signed by the application server or the temporary credentials issued with STS, that is, the AccessKey of the STS and token).

Best practices

- *What is RAM and STS*

Reference

- **Android SDK:** *Upload objects*
- **iOS SDK:** *Upload objects*

6 Compliant retention strategy

6.1 Introduction

Alibaba Cloud OSS fully supports the Write Once, Read Many (WORM) feature and allows users to store and use data in a manner that they cannot delete or modify the data, which conforms to the compliance requirements of Securities and Exchange Commission (SEC) and Financial Industry Regulation Authority (FINRA) of USA.

Note:

- As the only cloud service certificated by Cohasset Associates in China, Alibaba Cloud OSS is in compliance with the strict electronic archiving requirements of various regulations, such as EC Rule 17a-4(f), FINRA 4511, and CFTC 1.31. For more information, see *OSS Cohasset Assessment Report*.
- The compliant retention strategy feature is now in the beta testing phase in the China South 1 (Shenzhen) region and will apply to all regions in the future.
- You can only set a compliant retention strategy for buckets in OSS.

OSS supports strong compliant retention strategies. You can set a time-based compliant retention strategy for your buckets. After the strategy is locked, all users can upload objects to the buckets or read objects in the bucket. However, before the retention period for the objects expires, the protected objects and the retention strategy cannot be deleted. You can delete an object protected by the retention strategy only when the retention period expires. The WORM feature supported by OSS applies to various industries, such as finance, insurance, medical, and securities. You can build a "cloud-based compliant storage space" based on OSS.

Note:

During the valid period of a compliant retention strategy, you can *set lifecycle rules* to convert the storage class of the objects in the bucket protected by the strategy, which reduces storage costs while ensuring the compliance.

Operation

You can set a compliant retention strategy in the OSS console. For more information, see *Set a compliant retention strategy*.

Description

Currently, you can set only one time-based retention strategy with a retention period ranging from 1 day to 70 years.

Assume that you created a bucket named examplebucket on June 1, 2013, and then uploaded three objects named file1.txt, file2.txt, and file3.txt to the bucket at different times. After that, a compliant retention strategy was created for the bucket on July 1, 2014 with a protection period of 5 years. The following table describes the upload dates and expiration dates of the preceding three objects.

Object	Upload date	Expiration date
file1.txt	June 1, 2013	May 31, 2018
file2.txt	July 1, 2014	June 30, 2019
file3.txt	September 30, 2018	September 29, 2023

- Effective rules

  After a time-based retention strategy is created for a bucket, it is in the InProgress state by default, and the state is valid for 24 hours. Within the validity period, resources that apply to the strategy in the bucket are protected.

  - Within 24 hours after a compliant retention strategy is created for a bucket, the bucket owner and authorized users can delete the strategy if it is not locked. If the compliant retention strategy is locked, it cannot be deleted and the protection period of the strategy cannot be shorten. However, you can extend the protection period of the strategy.
  - If a compliant retention strategy is not locked after it is created for 24 hours, it is not locked.
  - If you try to delete or modify the data in a bucket protected by a compliant retention strategy, OSS API returns the `409 FileImmutable` message.
- Deletion rules
  - A time-based compliant retention strategy is a metadata property of a bucket. When a bucket is deleted, the compliant retention strategy and access strategies

set for the bucket are also deleted. Therefore, the owner of an empty bucket can delete the retention strategy set for the bucket by deleting the bucket.

- If a compliant retention strategy is not locked within 24 hours after it is created for a bucket, the bucket owner and authorized users can delete it.
- If a bucket stores objects which are in the protection period of a compliant retention strategy, you cannot delete neither the bucket nor the strategy set for the bucket.

6.2 Set a compliant retention strategy

A compliant retention strategy is used to specify the protection period of objects in a bucket. No one can modify or delete a protected object during the protection period.

 Note:

- For more information about compliant retention strategies, see *Introduction*.
- The compliant retention strategy feature is in the beta testing phase in the China South 1 (Shenzhen) region and will be supported in all regions.

Procedure

1. Log on to the *OSS console*.
2. In the bucket list on the left, click the name of the bucket that you want to set a compliant retention strategy for.
3. Click the Basic Settings tab, locate the Retention Strategy area, and click Configure.
4. Click Create Strategy.
5. In the Create Strategy dialog box, set the Retention period for the compliant retention strategy. The value range of Retention period is 1 day to 70 years.
6. Click OK.

    Note:
   After a compliant retention strategy is created, it is in the IN_PROGRESS state. You can Lock or Delete a compliant retention strategy in this state.
7. Click Lock.
8. Confirm the compliant retention strategy and click OK.

    Note:

- After this step, the strategy is in the LOCKED state. You can click Edit to extend the retention period.
- If you try to delete or modify the data in a bucket protected by a compliant retention strategy in the console, the `The file is locked and cannot be operated` error message is returned.

Calculate the retention time for an object

You can calculate the retention time for a protected object by adding the last update time of the object and the retention period specified in the compliant retention period. For example, a compliant retention strategy is set for a bucket, and the retention period specified in the strategy is 10 days. If the last update time of an object in the bucket is 2019-3-1 12:00, the object is protected until 2019-3-11 12:01. For more information about the compliant retention strategy, see *Introduction*.

6.3 FAQ

This topic answers the questions that are frequently asked in the use the compliant retention strategy for Alibaba Cloud OSS resources.

What are the advantages of the compliant retention strategy?

The compliant retention strategy ensures that data is stored in a manner conforming to the compliance requirements of industry regulations. Data protected by a compliance retention strategy cannot be modified or deleted by any user within the protection period. However, if you use RAM policies or bucket policies to protect the data, it may still be modified or deleted.

When should I set a compliant retention strategy?

If you want to store important data that is not allow to modify or delete (such as medical records, technical documentations, and contracts) for a long period, you can store the data in a bucket and set a compliant retention strategy for the bucket to protect the data.

Can I set a compliant retention strategy for an object?

Currently, you can set a compliant retention strategy only for a bucket but not a directory or an object.

Can I modify the storage class of an object after setting a compliant retention strategy for the bucket that stores the object?

After setting a compliant retention strategy for a bucket, you can *set lifecycle rules* to convert the storage class of the objects in the protected bucket to save storage costs.

How do I delete a bucket for which a compliant retention strategy is set?

- You can directly delete the bucket if no objects are stored in it.
- If objects that are out of the protection period are stored in the bucket, a failure occurs when you try to delete the bucket. In this case, you can delete the objects in the bucket first and then delete the bucket.
- If objects that are in the protection period are stored in the bucket, you cannot delete the bucket.

Are my objects within the protection period of a compliant retention strategy kept when my account has overdue bills?

If your account has overdue bills, Alibaba Cloud applies data retention strategies to your data based on the terms and conditions in your contract.

Can I set a compliant retention strategy as an authorized RAM user?

APIs related to the compliant retention strategy are open and integrated into RAM policies. By using a RAM user authorized by RAM policies, you can create or delete a compliant retention strategy in the console or through the APIs and SDKs.

7 Disaster recovery

7.1 Redundant storage across zones

Data in OSS is separately stored in three zones within the same region. Users can access the data even if one of the three zones is unavailable. With this redundant storage mechanism, OSS achieves 99.9999999999% data reliability and 99.99% data availability.

The redundant storage mechanism provides OSS with the disaster recovery capability in the data center level, that is, OSS can provide services with strong consistency even if a data center is not available because of network disconnection, power outage, or other disaster events. During failover, services are switched without interruption and data loss, ensuring that the failover process is not perceived by users. With the disaster recovery capability, OSS can meet the strict requirement that the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) of key services must be 0.

Note:
Currently, redundant storage across zones is supported only in the China South 1 (Shenzhen) and China North 2 (Beijing) regions. If you want to use redundant storage across zones in the China North 2 (Beijing) region, *open a ticket* to apply for the trial of this function.

Supported storage class

Redundant storage across zones supports two storage classes: Standard and Infrequent Access (IA). The following table compares the two storage classes from different dimensions.

Index	Standard	IA
Data reliability	99.9999999999%	99.9999999999%
Designed service availability	99.99%	99.99%
Storage measurement unit	Actual object size	64 KB
Minimum storage period	No limit	30 days

Index	Standard	IA
Data retrieval fee	No fee is charged for data retrieval.	Charged based on the size of retrieved data (GB).
Data access	Real-time access with latency of milliseconds	Real-time access with latency of milliseconds
Image processing	Supported	Supported

Enable redundant storage across zones

You can enable redundant storage across zones for a bucket only when you create the bucket.

If you want to separately store data in an existing bucket in multiple zones for redundant storage, use data migration tools (such as ossimport and SDK) to copy the data to a bucket with the redundant storage across zones feature enabled.

Usage

For more information about the usage of redundant storage across zones in the OSS console, see *Create a bucket*.

8 Buckets

8.1 Set bucket read and write permissions

When creating a bucket, the bucket owner can set the read and write permissions for the bucket using the Access Control List (ACL). After a bucket is created, the bucket owner can modify the bucket ACL according to business requirements. Currently, three access permissions are available for a bucket:

Permission	Access restriction
public-read-write	Anyone (including anonymous users) can perform read and write operations on the objects stored in the bucket . The fees incurred by these operations are borne by the bucket owner. Use this permission with caution.
public-read	Only the bucket owner and authorized users can perform write operations on the objects stored in the bucket. Other people (including anonymous users) can perform read operations on the objects.
private	Only the bucket owner and authorized users can perform read and write operations on the objects stored in the bucket. Other people cannot access the objects in the bucket without authorization.

For more information about ACL, see *Access control*.

Reference

Set bucket ACL:

- Console: *Set ACL*
- SDK: Java SDK-*Set bucket ACL*
- API: *Put BucketACL*

Get bucket ACL:

- Console: After logging on to the OSS console, view the ACL on the Basic Settings tab page.
- SDK: Java SDK- *Get bucket ACL*
- API: *Get BucketACL*

8.2 Obtain bucket region information

You can obtain the region information of a bucket. A region represents the physical location of a data center. The returned Location field indicates the region where a bucket is located. For example, if the physical location is East China 1 (Hangzhou), the returned Location field is oss-cn-hangzhou. For more information about regions, see *Regions and endpoints*.

Reference

- Console: After logging on to the console, you can view the region and endpoint information on the bucket overview page.
- API: *Get Bucket Location*
- SDK: Java SDK - *Get the bucket location*

8.3 View the bucket list

You can view a list displaying all the buckets that you have created.

Reference

- Console: After you log on to the console, you can directly view a list of all created buckets.
- API: *GetService*
- SDK: Java SDK - *List buckets*

Additional links

- *Create a bucket*

8.4 Attach a custom domain name

After an object is uploaded to an OSS bucket, a URL is automatically generated for the object. You can use this URL to access the object in the bucket. To access an uploaded object by using a custom domain name, you must attach the custom domain name to the bucket where the object is stored and add a CNAME record that directs to the Internet domain name of the bucket.

 Notice:

In accordance with the requirements of the Regulations on the Administration of the Internet of the People's Republic of China, all users who need to attach custom domain names must file their domain names in advance to the Ministry of Industry and Communications. If your domain name is not on file, you can *file* it through the ICP service provided by Alibaba Cloud.

Concepts

The following describes key concepts about attaching a custom domain name to a bucket:

- `User domain name (also called custom domain name or self-hosted domain name)`: indicates the domain name that you buy from a domain name provider.
- `OSS domain name (also called bucket domain name)`: indicates the domain name that OSS assigns to your bucket. You can use this domain name to access the resources in your bucket. To access an OSS bucket by using your user domain name, you must attach the user domain name to the OSS domain name of the bucket, that is, add a CNAME record in the Domain Name System (DNS) of Alibaba Cloud.
- `Alibaba Cloud CDN domain name`: indicates the CDN acceleration domain name that Alibaba Cloud Content Distribution Network (CDN) assigns to your user domain name. To use the CDN acceleration service to access the resources in your bucket, you must attach your user domain name to a CDN acceleration domain name, that is, add a CNAME record in the DNS of Alibaba Cloud.
- `Auto CDN cache update`: If you modify an object in your bucket but the object cache on the CDN node does not expire, users can only access the object that is not modified. In this case, you must manually update the object cache on the CDN node. To simplify operations, OSS provides the auto CDN cache update function. After you enable this function, all modifications on the objects in your bucket are automatically updated to the CDN node. For more information, see *Enable auto CDN cache update*.

Application scenarios

For example, you have a website with the domain name `img.abc.com`, and the website contains a picture with the following URL: `http://img.abc.com`

`/ logo . png` . For easier management, you want to redirect all access requests for the picture to OSS without code modifying the code, that is, keep the URL of the picture unchanged. In this case, you can attach a custom domain name to your bucket. To do so, follow these steps:

1. Create a bucket named *abc - img* , and upload the picture to the bucket.
2. Attach the custom domain name `img . abc . com` to the bucket *abc - img* through the OSS console.
3. After the custom domain name `img . abc . com` is attached to the bucket *abc - img* , OSS maps the domain name to the bucket.
4. Add a CNAME rule on your DNS server to map the custom domain name `img . abc . com` to `abc - img . oss - cn - hangzhou . aliyuncs . com` (the OSS domain name of the bucket *abc - img*).
5. After receiving a request for `http :// img . abc . com / logo . png` , OSS redirects the request to the bucket *abc - img* based on the mapping relationship between `img . abc . com` and *abc - img* . That is, users who access the picture with the URL `http :// img . abc . com / logo . png` are redirected to the following URL: `http :// abc - img . oss - cn - hangzhou . aliyuncs . com / logo . png` .

The following table describes the access processes before and after you attach the custom domain name.

	Before attaching the custom domain name	After attaching the custom domain name
Access process	1. A user sends a request to access `http :// img . abc . com / logo . png` . 2. DNS resolves the IP address of your server from the request. 3. The user accesses the picture logo.png on your server.	1. A user sends a request to access `http :// img . abc . com / logo . png` . 2. DNS resolves the URL `abc - img . oss - cn - hangzhou . aliyuncs . com` from the request. 3. The user accesses the picture *logo . png* in the OSS bucket *abc - img* .

References

- *Attach a custom domain name*
- *Attach a CDN acceleration domain name*
- *Certificate hosting*

8.5 Cross-region replication

This topic describes the application scenarios and limits of cross-region replication.

Bucket Cross-Region Replication enables automatic and asynchronous replication of objects across buckets in different OSS data centers (regions), which synchronizes operations (such as creating, updating, and deleting) performed on objects in the source bucket to the target bucket in a different region. This feature could be a boon to customers looking for cross-region disaster recovery for their buckets or data replication. Objects in the target bucket are precise copies of objects in the source bucket. They have the same object name, metadata, and content (such as creation time, owner, user-defined metadata, object ACL, and object content).

 Notice:

- Currently, the cross-region replication feature is supported only between the regions in Mainland China and between the US West 1 (Silicon Valley) and US East 1 (Virginia) regions.
- The cross-region replication function is in the beta testing phase and is free currently. The function is charged after it is officially released.

Application scenarios

You may configure cross-region replication for your buckets for a variety of reasons, including:

- Compliance requirements: Although, by default, OSS stores multiple copies of each object on a physical disk, compliance requirements may dictate that you store a copy of the data at a further distance. Cross-region replication allows you to replicate data between distant OSS data centers to satisfy these compliance requirements.
- Minimize latency: Your customers are in two geographic locations. To minimize latency in accessing objects, you can maintain object copies in OSS data centers that are geographically closer to your users.

- Data backup and disaster recovery: You have high requirements on data security and availability, and you want to explicitly maintain copies of all written data in a second data center. In case one OSS data center is damaged by a catastrophic event like earthquake and tsunami, you can use backup data from the other one.
- Data replication: For business reasons, you may need to migrate data from one OSS data center OSS to another.
- Operational reasons: You have computing clusters in two different data centers that analyze the same set of objects. You may choose to maintain object copies in these regions.

Operation method

Operation method	Description
OSS console	Web application that is easy to use
Java SDK	Various and complete SDK demos in different languages

Usage

Cross-region replication supports synchronization of buckets with different names. If the two buckets are in different regions, you can use this feature to synchronize the data of the source bucket to the target one in real time. This feature now offers the following capabilities:

- Real-time synchronization: This provides the ability to monitor data additions, deletions, and modifications in real time and synchronize these changes to the target bucket. For files of 2 MB in size, data is synchronized in a matter of minutes to guarantee data consistency between the source and the target.
- Historical data migration: This provides the ability to synchronize historical data from a bucket to form two identical data copies.
- Real-time display of synchronization progress: This shows the point in time of the last synchronization for real-time data synchronization and the percentage of completion for historical data migration.
- Easy configuration: The OSS console provides an easy-to-use interface for configuration management.
- Mutual synchronization: You can achieve mutual synchronization between bucket A and bucket B as follows: Configure the synchronization from bucket A to bucket B first, and then configure the synchronization from bucket B to bucket A.

Restrictions

- For two buckets that are in synchronization, because you can operate on both buckets at the same time, copying an object from the source bucket may overwrite the object with the same name in the target bucket. Be cautious when using this feature.
- Because Bucket Replication uses an asynchronous copying method, it can take from minutes to hours to copy data to the target bucket, depending on the size of the objects being replicated.
- Cross-region synchronization only works when no synchronization to a third bucket is enabled for the two buckets to be synchronized. For example, if synchronization to Bucket B is enabled for Bucket A, you can no longer enable synchronization to Bucket C for Bucket A, unless you delete the former configurat ion first. Similarly, if synchronization to Bucket B is enabled for Bucket A, it is not allowed to enable synchronization from Bucket C to Bucket B.
- Synchronization is supported only between two buckets in different regions.
- Currently, the cross-region replication feature is supported only between the regions in Mainland China and between the US West 1 (Silicon Valley) and US East 1 (Virginia) regions.

8.6 Anti-leech settings

OSS collects service fees based on use. To prevent users' data on OSS from being leeched, OSS supports anti-leech based on the field referer in the HTTP header.

You can log on to the *OSS console* or use APIs to configure a referer whitelist for a bucket or whether to allow access by requests where referer is blank.

For example, for a bucket named oss-example, set its referer whitelist to `https://www.aliyun.com/`. Then, only requests with a referer of `https://www.aliyun.com/` can access the objects in the bucket.

Detail analysis

- Anti-leech verification is performed only when users access objects through URL signatures or anonymously. When the request header contains the "Authorization" field, anti-leech verification is not performed.
- A bucket supports multiple referer fields, which are separated by the Enter key on the console, and by the comma "," on API.

- The referer field supports the wildcard “*” and “?”.
- Users can set whether to allow access requests with empty referer fields.
- When the whitelist is empty, the system checks if the referer field is null (otherwise, all requests get rejected).
- When the whitelist is not empty and the rules do not allow null referer fields, only requests with referers in the whitelist are allowed. Other requests (including null referer requests) are then rejected.
- If the whitelist is not empty and the rules allow empty referer fields, requests with empty referer and with the referers in the whitelist are allowed. Other requests get rejected.
- The three bucket permissions (private, public-read, and public-read-write) check the referer field.

Wildcard details:

- Asterisk “*”: The asterisk can be used to represent 0 or multiple characters. If you are looking for an object name prefixed with AEW but have forgotten the remaining part, you can enter AEW* to search for all types of files starting with AEW, such as AEWT.txt, AEWU.EXE and AEWI.dll. If you want to narrow down the search scope, you can enter AEW*.txt to search for all .txt files starting with AEW, such as AEWIP.txt and AEWDF.txt.
- Question mark “?”: The question mark can be used to represent one character. If you enter love?, all types of files starting with love and ending with one character get displayed, such as lovey and lovei. If you want to narrow the search scope, you can enter love?.doc to search for all .doc files starting with love and ending with one character, such as lovey.doc and loveh.doc.

Reference

- API: *PutBucketReferer*
- Console: *Set anti-leech*

8.7 Cross-origin resource sharing

Cross-origin access, or the cross-origin of JavaScript, is a browser restriction set with the purpose of security, such as the same-origin policy. When Website A tries to use the JavaScript code in its webpage to access Website B, the attempt is rejected by the browser because A and B are two websites of different origins.

Cross-origin access must be used frequently, such as when OSS is used at the backend for the user’ s website www.a.com. The upload function implemented with JavaScript is provided in the webpage. However, requests could only be sent to www .a.com in the webpage, and all the requests sent to other websites are rejected by the browser. Thus the data uploaded by users has to be relayed to other sites through www.a.com. If cross-origin access is set, users could upload their data directly to OSS instead of relaying it through www.a.com.

Cross-origin

resource sharing (CORS) is the standard across-origin solution provided by HTML5. Currently, the CORS standard is supported by OSS for cross-origin access. For more information, see *W3C CORS Norms*. CORS indicates the origin from where the request is originated

1. By using a header containing the origin of the HTTP request. As in the earlier example, origin header contains www.a.com.
2. After receiving the request, the server judges based on certain rules whether the request must be accepted or not. If yes, then the server attaches the Access-Control-Allow-Origin header in the response. The header contains www.a.com, indicating that cross-origin access is allowed. If the server accepts all the cross-origin requests, set the Access-Control-Allow-Origin header to *.
3. The browser determines whether the cross-origin request is successful or not based on whether the corresponding header has been returned or not. If no corresponding header is attached, the browser blocks the request. If the request is not a simple one, the browser initially send an OPTIONS request to obtain the CORS configuration of the server. If the server does not support the following operations , the browser blocks the following requests.

OSS provides the configuration of the CORS rule, accepting or rejecting corresponding cross-origin requests as required. The rule is configured at the bucket level. The details are available in *PutBucketCORS*.

Key points

- Attaching relevant CORS headers and other actions are automatically executed by the browser, and no additional action is required by the user. Only in the browser environment could the CORS operations be meaningful.

- Whether a CORS request is accepted is completely independent of OSS authentication and other such measures. The OSS CORS rule is only used to determine whether to attach the relevant CORS headers. Whether the request is required to be blocked or not, this can be exclusively determined by the browser.
- When using cross-origin requests, make sure the browser's cache function is enabled. For example, the same cross-origin resource have been requested by two webpages running on the same browser (originated from www.a.com and www.b.com) at the same time respectively. If the request of www.a.com is received by the server initially, the server returns to the user the resource with the Access-Control-Allow-Origin header "www.a.com". When www.b.com initiates its request, the browser returns its previous cached request to the user. As the header content does not match the CORS request, the subsequent request fails.

Reference

- API: *Introduction*
- SDK: Java SDK-*CORS*
- Console: *Set CORS*

8.8 Delete a bucket

Before you can delete the bucket you have created, you must delete all the objects and fragments in the bucket. We recommend that you define object lifecycle rules to delete all the objects and fragments in a bucket. For more information about how to define object lifecycle rules, see *Manage object lifecycle*.

Reference

- API: *Delete Bucket*
- SDK: Java SDK - -*Delete a bucket*
- Console: *Delete a bucket*

9 Upload files

9.1 Simple upload

Simple upload refers to the upload of a single object by using the Put Object method in the OSS API. Simple upload is applicable to the scenario where a single HTTP request interaction completes an upload, for example, the upload of a small object.

Set object metadata when uploading an object

When using the simple upload, you can set object metadata that describes the object, for example, Content-Type and other standard HTTP headers. You can also set user-defined information. For more information, see *Object metadata*.

Upload restrictions

- The maximum size of a single object is 5 GB.
- The naming conventions of objects are as follows:
  - Object names must use UTF-8 encoding.
  - Object names must be at least 1 byte and no more than 1,023 bytes in length.
  - Object names cannot start with a backslash (\ ) or a forward slash (/).

Upload large objects

In the single upload, objects are uploaded through a single HTTP request. Therefore, it may take a long time for you to upload large objects. If you experience bad network connection, the upload has a high failure rate. For objects larger than 5 GB, we recommend that you use *multipart upload*.

Security and authorization

To prevent unauthorized third parties from uploading objects to your bucket, OSS provides access control both on the bucket level and on the object level. For more information, see *Access control*. OSS also provides account-level authorization for third-party uploads. For more information, see *Authorized third-party uploads*.

Further operations

After uploading objects to OSS, you may want to:

- Initiate a callback request to a specified application server. For more information, see *Upload callback*.
- Process the uploaded images. For more information, see *Image processing*.

Reference

- API: *PutObject*
- Java SDK: *Simple upload*
- Console: *Upload objects*

Best practices

- *RAM and STS User Guide*
- *Web client direct data transfer and upload callback*

9.2 Form upload

Form upload refers to the upload of an object by using the Post Object method in the OSS API. The object to be uploaded cannot be larger than 5 GB. This method can be used in HTML web pages to upload objects. A typical scenario is web applications.

Take a job-search website as an example. The comparison between the process with and without using form upload is as follows:

Process without using form upload	Process using form upload
1. A website user uploads a resume. 2. The website server responds to the upload page. 3. The resume is uploaded to the server. 4. The server uploads the resume to OSS.	1. A website user uploads a resume. 2. The website server responds to the upload page. 3. The resume is uploaded to OSS.

Upload restrictions

- The maximum size of a single object is 5 GB.
- The naming conventions of objects are as follows:
  - Object names must use UTF-8 encoding.
  - Object names must be at least 1 byte and no more than 1,023 bytes in length.
  - Object names cannot start with a backslash (\ ) or a forward slash (/).

Advantages of form upload

If the form upload is not used, files are uploaded to the web server first, and then the web server forwards the files to OSS. In case of huge uploads, the web server becomes the bottleneck and needs to be scaled up. If the form upload is used, files are uploaded directly from the client to OSS without the forwarding of the web server. OSS handles all upload requests and guarantees the service quality.

Security and authorization

To prevent unauthorized third parties from uploading objects to your bucket, OSS provides access control both on the bucket level and on the object level.

To grant upload permissions to a third party, you can use the PostObject interface. For more information, see *PostObject*.

Procedures for form upload

1. Construct a Post policy.

   The policy form field of the Post request is used to verify the validity of the request. For example, the policy can specify the size and name of objects to be uploaded, the redirect URL of the client, and the status code the client receives after a successful upload. For more information, see *Post Policy*.

   In the following example of policy, the expiration time for uploads by website users is 2115-01-27T10:56:19Z (a long expiration period is set for tests only and is not recommended in actual use) and the maximum file size is 104857600 bytes.

   ```
   This example uses the Python code and the policy
   is a string in JSON format .
   policy ="{\" expiration \":\" 2115 - 01 - 27T10 : 56 : 19Z \",\"
   conditions \":[[\" content - length - range \", 0 , 104857600
   ]]}"
   ```

2. Encode the policy string using Base64.
3. Use the OSS AccessKeySecret to sign the Base64-encoded policy.
4. Construct an HTML page for uploads.
5. Open the HTML page and select and upload files.

A complete Python code example is as follows:

```
# coding = utf8
import md5
import hashlib
import base64
import hmac
```

```
from   optparse   import   OptionPars  er
def   convert_ba  se64 ( input ):
    return   base64 . b64encode ( input )
def   get_sign_p  olicy ( key ,  policy ):
    return   base64 . b64encode ( hmac . new ( key ,  policy ,
hashlib . sha1 ). digest ())
def   get_form ( bucket ,  endpoint ,  access_key  _id ,  access_key
_secret ,  out ):
   # 1 .  Construct   a   Post   policy
    policy ="{\" expiration \":\" 2115 - 01 - 27T10 : 56 : 19Z \",\"
conditions \":[[\" content - length - range \",  0 ,  1048576 ]]}"
    print (" policy : % s " %  policy )
   # 2 .  Encode   the   policy   string   using   Base64
    base64poli  cy  =  convert_ba  se64 ( policy )
    print (" base64_enc  ode_policy : % s " %  base64poli  cy )
   # 3 .  Use   the   OSS   AccessKeyS  ecret   to   sign   the
Base64 - encoded   policy
    signature  =  get_sign_p  olicy ( access_key  _secret ,
base64poli  cy )
   # 4 .  Construct   an   HTML   page   for   uploads
    form  = '''
   < html >
       < meta   http - equiv = content - type   content =" text /
html ;  charset = UTF - 8 ">
       < head >< title > OSS   form   upload  ( PostObject )</ title
></ head >
       < body >
           < form    action =" http ://% s .% s "  method =" post "
enctype =" multipart / form - data ">
               < input   type =" text "  name =" OSSAccessK  eyId "
value ="% s ">
               < input   type =" text "  name =" policy "  value ="%
s ">
               < input   type =" text "  name =" Signature "  value
="% s ">
               < input   type =" text "  name =" key "  value ="
upload /${ filename }">
               < input   type =" text "  name =" success_ac
tion_redir  ect "  value =" http :// oss . aliyun . com ">
               < input   type =" text "  name =" success_ac
tion_statu  s "  value =" 201 ">
               < input   name =" file "  type =" file "  id =" file
">
               < input   name =" submit "  value =" Upload "  type ="
submit ">
           </ form >
       </ body >
   </ html >
   ''' % ( bucket ,  endpoint ,  access_key  _id ,  base64poli  cy ,
signature )
    f  =  open ( out , " wb ")
    f . write ( form )
    f . close ()
    print (" form   is   saved   into  % s " %  out )
if   __name__  == ' __main__ ':
    parser  =  OptionPars  er ()
    parser . add_option ("", "-- bucket ",  dest =" bucket ",  help
=" specify  ")
    parser . add_option ("", "-- endpoint ",  dest =" endpoint ",
help =" specify ")
    parser . add_option ("", "-- id ",  dest =" id ",  help ="
access_key  _id ")
    parser . add_option ("", "-- key ",  dest =" key ",  help ="
access_key  _secret ")
```

```
    parser . add_option ("", "-- out ",  dest =" out ",  help =" out
  put   form ")
   ( opts ,  args ) =  parser . parse_args ()
    if   opts . bucket   and   opts . endpoint   and   opts . id
and   opts . key   and   opts . out :
        get_form ( opts . bucket ,  opts . endpoint ,  opts . id ,
opts . key ,  opts . out )
    else :
        print  " python  % s  -- bucket = your - bucket  -- endpoint
= oss - cn - hangzhou . aliyuncs . com  -- id = your - access - key -
id  -- key = your - access - key - secret  -- out = out - put - form
- name " %  __file__
```

Save this code example as post_object.py and run it by using python post_object.py.

```
Usage :
python   post_objec  t . py  -- bucket = Your   bucket  -- endpoint =
The   bucket ' s   OSS   domain   name  -- id = Your   AccessKeyI  d
-- key = Your   AccessKeyS  ecret  -- out = Output   file   name
Example :
python   post_objec  t . py  -- bucket = oss - sample  -- endpoint
= oss - cn - hangzhou . aliyuncs . com  -- id = tphpxp  -- key =
ZQNJzf4QJR  krH4  -- out = post . html
```

Note:

In the constructed form,

- `success_ac tion_redir ect value = http :// oss . aliyun . com` indicates the redirect URL after a successful upload. You can replace it with your own page.
- `success_ac tion_statu s value = 201` indicates that Status Code 201 is returned after a successful upload. This value can be replaced.

If the specified HTML file is post.html, open post.html and select the file to be uploaded. In this example, the client redirects to the OSS homepage `http://oss.aliyun.com` after a successful upload.

Usage

- API: *PostObject*
- Java SDK: *Form upload*

Best practices

- *Web client direct upload*
- *Cross-origin Resource Sharing (CORS)*

9.3 Multipart upload

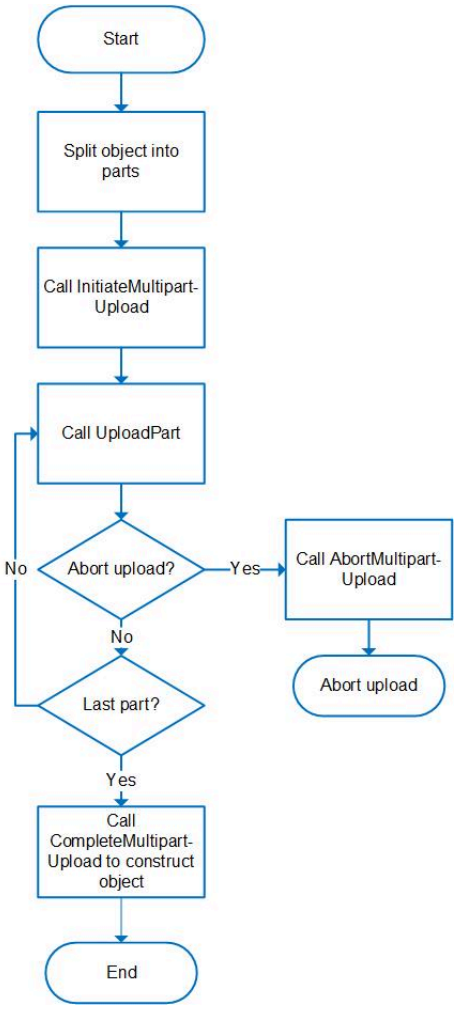
For the object larger than 5 GB, you can use the multipart upload to split it into multiple data blocks (called parts in OSS) and upload them separately. When you have uploaded all parts, OSS constructs the object from the uploaded parts.

We recommend that you use the multipart upload in the following scenarios:

- Poor network connectivity: If the upload of one part fails, you can re-upload only the failed part instead of all parts.
- Resumable upload required: An upload in progress can be paused and resumed at any time.
- Upload acceleration: Multiple parts can be uploaded concurrently to speed up the process.
- Streaming upload: Objects of unknown sizes can be uploaded at any time. This scenario is common in industry applications such as video surveillance.

Workflow

The workflow of the multipart upload is shown as follows:



The description of the workflow is as follows:

1. You split the object into multiple parts.
2. You initiate a multipart upload task. For more information, see (*InitiateMultipartUpload*).
3. You upload the parts one by one or concurrently. For more information, see (*UploadPart*).
4. After all the parts are uploaded, OSS combines them into the original object. For more information, see (*CompleteMultipartUpload*).

When you use the multipart upload, take the following into consideration:

- All the parts, except the last one, must not be smaller than 100 KB. Otherwise, the call to the *CompleteMultipartUpload* interface fails.
- After the object is split into parts, the parts are ordered by the partNumbers specified during the upload. The upload speed does not correlate to the number of

parts uploaded concurrently, because both the network conditions and the device load must be considered.

- By default, when the upload is complete but the call to the *CompleteMultipartUpload* interface fails, the uploaded parts will not be deleted automatically. You can call the *AbortMultipartUpload* interface to terminate the upload and save the storage space. To automatically delete the uploaded parts, see *Manage object lifecycle*.

Resumable upload

The uploaded parts will not disappear unless you delete them. Therefore, the multipart upload can be considered as the resumable upload.

If the system crashes during a multipart upload, you can resume the upload by using the *ListMultipartUploads* and the *ListParts*interface to list the uploaded parts in each task. This allows uploads to be resumed from the last uploaded part. The same logic applies to pausing and resuming uploads.

The multipart upload is particularly well suited for the data transfer between mobile devices and the large file upload.

Restrictions

- The maximum size of an object is determined by the size of parts. The multipart upload supports a maximum of 10,000 parts, and each part must be at least 100 KB (except for the last part, which may be smaller) and no more than 5 GB. Therefore, the object size must not exceed 48.8 TB.
- The naming conventions of objects are as follows:
  - Object names must use UTF-8 encoding.
  - Object names must be at least 1 byte and no more than 1,023 bytes in length.
  - Object names cannot start with a backslash (\ ) or a forward slash (/).

Security and authorization

To prevent unauthorized third parties from uploading objects to your bucket, OSS provides access control both on the bucket level and on the object level. For more information, see *Access control*.

OSS also provides account-level authorization for third-party uploads. For more information, see *Authorized third-party uploads*.

Further operations

After uploading objects to OSS, you may want to:

- Initiate a callback request to a specified application server. For more information, see *Upload callback*.
- Process the uploaded data. For more information, see *Cloud data processing*.

Usage

- API:
  - *MultipartUpload*
  - *InitiateMultipartUpload*
  - *UploadPart*
  - *UploadPartCopy*
  - *CompleteMultipartUpload*
  - *AbortMultipartUpload*
  - *ListMultipartUploads*
  - *ListParts*
- SDK: Java SDK- MultipartUpload in *Upload objects*

Best practices

- *RAM and STS best practices*
- *Web client direct upload*

9.4 Append object

Applicable scenarios

The *Simple upload*, *Form upload*, and *Multipart upload* methods create normal-type objects which have fixed content after the upload is finished. They can only be read, but cannot be modified. If the object content changes, the user must upload an object of the same name to overwrite the content. This is a major difference between OSS and file systems.

This feature makes many application scenarios inconvenient, such as video monitoring and live video broadcast, since video data is constantly produced in real time. Using other upload methods, users must slice the video stream into small

pieces and then upload them as new objects. In actual use, these methods have obvious defects:

- The software architecture is quite complex and users must consider intricate issues such as file fragments.
- Storage space is required for metadata, e.g. the list of generated objects. Thus, each request must read the metadata to judge if any new object has been generated . This puts a high level of access pressure on the server. In addition, each client request must be transmitted twice, causing a certain amount of delay.
- If the object parts are small, the delay is quite short. However this complicates the management of most objects. If the object parts are large, the data suffers a substantial delay.

To simplify development and reduce costs in such a scenario, OSS provides the append object method, which allows users to directly append content to the end of an object. This method is used to operate on Appendable objects. The objects uploaded by other methods are Normal objects. The data appended is instantly readable.

With append object, the previous scenario becomes simple. When video data is produced, they can be immediately added to the same object through the append object method. The client only needs to regularly retrieve the object length and compare it with the previous value. If new readable data is found, this triggers a read operation to retrieve the newly uploaded data segments. This method greatly simplifies the architecture and enhances the scalability of applications.

In addition to video scenarios, the append object method can also be used to append log data.

Upload restrictions

- Size limit: The maximum object size is 5 GB in this mode.
- Naming restrictions
  - Object names must use UTF-8 encoding.
  - Object names must be at least 1 byte and no more than 1,023 bytes in length.
  - Object names cannot start with a backslash (\ ) or a forward slash (/).
- File type: Only files created through append object can be appended with new data . Therefore, new data cannot be appended to files created through simple upload, form upload, or multipart upload.

- Subsequent operation restrictions: No files created through append object can be copied, but you can modify the meta-information for the file itself.

Upload security and authorization

To prevent unauthorized third parties from uploading objects to the developer's bucket, OSS provides bucket-level and object-level access permission control. For more information, see *Access control*. In addition to bucket-level and object-level access permissions, OSS also provides account-level authorization to authorize third-party uploads. For more information, see *Authorized third-party upload*.

Post-upload Operations

To process uploaded images, users can use *Image Processing*. For audio/video file format conversion, users can use *Media Processing*.

Reference for using the function

- API: *Append Object*
- Java SDK: *Append object*

Note:
Append object method does not support upload callback.

Best practices

- *RAM and STS User Guide*

9.5 Authorized third-party upload

Applicable scenarios

In standard client/server system architecture, the server is used for receiving and processing requests from the client. If OSS is used as a backend storage service, the client sends objects to the application server to upload, then forward, the objects to the OSS. In this process, the data need to be transmitted twice. Regarding high access volume scenarios, the server requires high bandwidth resources to satisfy multiple clients' simultaneous upload needs, challenging the architecture's scalability.

To resolve this issue, OSS provides an authorized third-party upload function. This means each client can directly upload files to the OSS, bypassing the need for a server

. This reduces the cost for application servers and takes full advantage of the OSS' s ability to process massive data volumes.

Currently, two methods are provided to grant upload permissions: URL signature and STS.

URL signature

The URL signature method adds an OSS AccessKeyID and Signature fields to the request URL, allowing users to directly use this URL for an upload. Each URL signature has an expiration time to guarantee security. For more information, see *Add a signature to the URL*.

Temporary access credentials

Temporary access credentials are granted through the Alibaba Cloud Security Token Service and provide users with access authorization. For information on the implementation of temporary access credentials, see *STS Java SDK*. The process for temporary access credentials is as follows:

1. The client initiates an authorized request to the server. The server verifies the legitimacy of the client. If it is a legitimate client, then the server uses its own AccessKey to make a request to the STs for authorization. For more information, see *Access control*.
2. The server returns the obtained temporary credentials to the client.
3. The client uses the obtained temporary credentials to initiate an upload request to OSS. For more information, see *Temporary authorization access*. The client can cache this credential for upload until the credential expires and then request new credentials from the server.

Best practices

- *RAM and STS User Guide*
- *Web client direct data transfer and upload callback*

9.6 Upload callback

Use cases

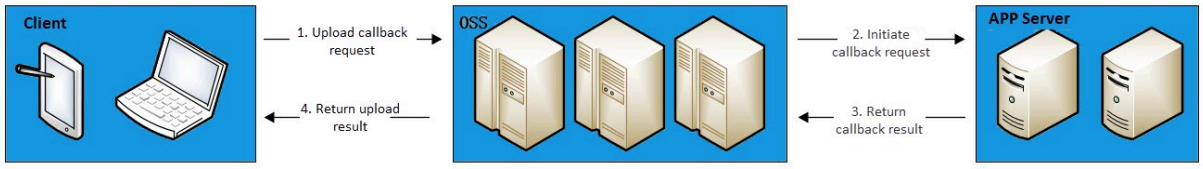
When an object upload is completed, the OSS can provide a callback to the applicatio n server. To implement the callback, you only need to attach the relevant Callback

parameter to the request sent to the OSS. APIs that currently support callbacks include PutObject, PostObject, and CompleteMultipartUpload.

A typical use case for upload callback is to work with the upload by an authorized third-party. The client specifies the callback of the server when it uploads objects to the OSS. After the upload task of the client is completed in the OSS, the OSS automatically initiates an HTTP request for the callback to the application server. This promptly notifies the server that the upload is completed, so the server can complete operations such as database modifications. When the callback request receives a response from the server, the OSS returns the status to the client.

When the OSS sends a POST callback request to the application server, the POST request's body contains some parameters that carry certain information. Such parameters are divided into two types: system-defined parameters (such as bucket name and object name) and user-defined parameters. You can specify user-defined parameters based on the application logic when sending a request including callback to the OSS. You can use user-defined parameters to carry information relevant to the application logic, such as the user ID of the request initiator. For information on user-defined parameter usage, see *Callback*.

Appropriate use of the upload callback can decrease the complexity of the client's logic and reduce the consumption of network resources. The process is as follows:



Note:

- Supported regions include Mainland China regions, Hong Kong region, Asia Pacific South 1, Asia Pacific SE 2, US East, US West, Asia Pacific Northeast 1, Middle Europe 1 and Middle East 1.
- Currently only simple uploads (PutObject), form uploads (PostObject) and multipart uploads (Complete Multipart Upload) operations support upload callback.

Reference

- API: *Callback*

- SDK: iOS *Callback notification after upload*
- *Upload callback*

Best practices

- *Direct data transfer practices on web clients and upload callback*
- *How to build a callback application server (sample code is available for download)*

10 Download files

10.1 Simple download

A simple download occurs when a user downloads an uploaded file (object). The object download is accomplished through an HTTP GET request. For the rules of generating object URLs, see *Accessing OSS*. For the access to an object by a user-defined domain name, see *Accessing OSS with User-defined Domain Names*.

A user may access a certain object in two conditions:

- This object does not have anonymous read permission, but the user has a corresponding AccessKey, which can be used to sign the GET request and access the object.
- This object has anonymous read permission, so all users can directly access the object through GET requests.

For more information about object and bucket access permission control, see *Access control*.

To authorize a third-party user to download an object from a private bucket, see *Authorized third-party download*.

To use multipart download, see *Multipart download*.

Reference

- API: *GetObject*
- SDK: Java SDK-*Object*
- Console: *Get object URL*

Best practices

- *RAM and STS User Guide*

10.2 Multipart download

OSS provides a "start object download from specified point" function. This allows users to spilt large objects into multiple downloads, which improves speed and

reliability of downloads. If a download is paused or interrupted, it resumes at the point of interruption once restarted.

Similar to a simple upload, the user must have read permission for the object. Multipart downloads are supported when the Range parameter is set. If the Range parameter is specified in the request header, the returned message contains the length of the entire file and the range returned in this response. For example, Content-Range: bytes 0-9/44 indicates that the length of the entire file is 44, and the range in the response body is 0–9. If the range requirement is not met, the system transfers the entire file and does not include Content-Range in the result. The return code is 206.

Reference

- API: *GetObject*

10.3 Authorized third-party download

Use a URL signature, or provide temporary access credentials, to grant third party authorization to download objects in a private bucket. These methods are recommended as they prevent directly giving the AccessKey to users requesting download permissions, which can weaken account security.

URL signature

A developer can add a signature into the URL and forward this URL to a third party to authorize access. The third-party user can then access this URL using an HTTP GET request to download the object.

- Implementation method

  Example URL that includes a signature:

  ```
  http ://< bucket >.< region >. aliyuncs . com /< object >?
  OSSAccessK  eyId =< user   access_key  _id >& Expires =< unix
  time >& Signature =< signature_  string >
  ```

  The signature in the URL must include the following three parameters:

  - OSSAccessKeyId, which is the developer’s AccessKeyId.
  - Expires, which is the developer’s expected URL expiration time.
  - Signature, which is the developer’s signature string. For more information, see *Add a signature to a URL*.

    Note:
    This link must undergo URL encoding.

- Reference

  - API: *Get Object*
  - SDK: Java SDK-*Using URL Signature to Authorize Access*
  - Console: *Get object URL*

    Note:
    If the bucket permission is set to private read/write permission, the access URL provided on the console contains a signature.

Temporary access credentials

Security Token Service (STS) can be used to provide temporary credentials to third-party users. By adding a signature in the request header, users can then access the object. This authorization method is applicable to mobile scenario downloads. For more information on the implementation of temporary access credentials, see *STS Java SDK*.

Implementation method

Third-party users send a request to the application server to obtain an AccessKeyID, AccessKeySecret, and STS Token issued by STS. Upon receipt, the AccessKeyID, AccessKeySecret, and STS Token are used as a signature to request the developer’s object resource.

Reference

- **API:** *Temporary Access Credentials*
- **SDK: Use STS temporary authorization in Java SDK-***Object*
- **Console:** *Get object URL*

Best practices

- *RAM and STS User Guide*

11 Manage files

11.1 Object Meta

Object Meta describes the attributes of files uploaded to OSS. These attributes are classified into two types: HTTP standard attributes (HTTP Headers) and User Meta (custom metadata). File metadata can be configured when files are uploaded or copied.

- HTTP standard attributes

Name	Description
Cache-Control	Cache action of the web page when the object is downloaded
Content-Disposition	Name of the object when downloaded
Content-Encoding	Content encoding format when the object is downloaded
Content-Language	Specifies the content language encoding when the object is downloaded
Expires	Expiry time
Content-Length	Size of the object
Content-Type	File type of the object
Last-Modified	Time of last modification

- User Meta

  This attribute allows you to enrich the description of objects using custom metadata. In OSS, all parameters prefixed with "x-oss-meta-" are considered as User Meta, such as x-oss-meta-location. A single object can have multiple similar parameters, but the total size of all User Meta cannot exceed 8 KB. User Meta information is returned in the HTTP header during GetObject or HeadObject operations.

Set object Meta when uploading objects

You can set object Meta when uploading objects.

Reference:

- API: *PutObject*
- SDK: Java SDK-Set the HTTP header and User-defined metadata in *Upload objects*

You can set object Meta when using multipart uploads.

Reference:

- API: *InitiateMultipartUpload*
- SDK: Java SDK-*Initialize multipart upload*

Modify object Meta after uploading objects

To modify the object metadata without modifying the actual data, using the copy object interface is recommended. In this way, you only need to apply the new metadata in the HTTP header and set the copy source and destination addresses to the current address of the object.

Reference

- API: *CopyObject*
- SDK: Java SDK-*Use CopyObjectRequest to copy objects*

Retrieve object Meta

This feature applies when you must retrieve object Meta, but not the object data.

Reference:

- API: *HeadObject*
- SDK: Java SDK-*Get object metadata*

11.2 View the object list

You can use this feature to view the objects uploaded to your bucket. Up to 1,000 objects in a selected bucket can be displayed at one time. The following four parameters provide users with extended capabilities:

Name	Function
Delimiter	Groups object name characters. All objects whose names are found between the specified prefix and the first occurrence of the Delimiter act as a group of elements: CommonPrefixes.

Name	Function
Marker	Sets up the returned results to begin from the first entry after the Marker, and is sorted in alphabetical order.
MaxKeys	Limits the maximum number of objects returned for one request. If this parameter specified, the default value is 100. The MaxKeys value cannot exceed 1, 000.
Prefix	Indicates that only the objects whose keys contain the specified prefix are returned. Note that keys returned from queries using a prefix still contains the prefix.

Folder simulation

OSS does not support folders, or directory sorting. All elements are stored as objects. Creating a simulated folder means creating an object with a size of 0 that can then be uploaded and downloaded. The console displays any object ending with "/" as a folder . So you can use the preceding method to create a simulated folder.

Users can use a combination of Delimiters and Prefixes to simulate folder functions as follows: The combination of Delimiter and Prefix is as follows:

- Setting the Prefix as the name of a folder enumerates the files starting with this prefix, recursively returning all files and subfolders (directories) in this folder. The file names are shown in Contents.
- Setting the Delimiter as "/" means that the returned values enumerate the files in the folder and the subfolders (directories) returned in the CommonPrefixes section . Recursive files and folders in subfolders are not displayed.

```
For   example :
In   this   example ,  the   OSS   bucket   oss - sample ,  contains
  the   following   objects :
File   D
Directory   A / File   C
Directory   A / File   D
Directory   A / Directory   B / File   B
Directory   A / Directory   B / Directory   C / File   A
Directory   A / Directory   C / File   A
Directory   A / Directory   D / File   B
Directory   B / File   A
1 .  List   first - level   directorie  s   and   files
Based   on   the   API   request   convention  s ,  you   must   set
  the   Prefix   to  "",  and   the   Delimiter   to  "/":
```

```
 The   returned   results   are   as   follows :
<? xml   version =" 1 . 0 "  encoding =" UTF - 8 "?>
< ListBucket  Result >
  < Name > oss - sample </ Name >
  < Prefix ></ Prefix >
  < Marker ></ Marker >
  < MaxKeys > 1000 </ MaxKeys >
  < Delimiter >/</ Delimiter >
  < IsTruncate  d > false </ IsTruncate  d >
  < Contents >
    < Key > File   D </ Key >
    < LastModifi  ed > 2015 - 11 - 06T10 : 07 : 11 . 000Z </
 LastModifi  ed >
    < ETag >" 8110930DA5  E04B1ED5D8  4D6CC4DC90  80 "</ ETag >
    < Type > Normal </ Type >
    < Size > 3340 </ Size >
    < StorageCla  ss > Standard </ StorageCla  ss >
    < Owner >
      < ID > oss </ ID >
      < DisplayNam  e > oss </ DisplayNam  e >
    </ Owner >
  </ Contents >
  < CommonPref  ixes >
    < Prefix > Directory   A /</ Prefix >
  </ CommonPref  ixes >
  < CommonPref  ixes >
    < Prefix > Directory   B /</ Prefix >
  </ CommonPref  ixes >
</ ListBucket  Result >
 We   can   see   that :
 Contents   returns   the   first - level   file : " File   D ".
 CommonPref  ixes   returns   the   first - level   directorie  s : "
 Directory   A /"  and  " Directory   B /", but   the   files   in
 these   directorie  s   are   not   shown .
 2 .  List   second - level   directorie  s   and   files   under
 Directory   A
 Based   on   the   API   request   convention  s ,  you   must   set
   the   Prefix   to  " Directory   A ",  and   the   Delimiter   to
 "/":
 The   returned   results   are   as   follows :
<? xml   version =" 1 . 0 "  encoding =" UTF - 8 "?>
< ListBucket  Result >
  < Name > oss - sample </ Name >
  < Prefix > Directory   A /</ Prefix >
  < Marker ></ Marker >
  < MaxKeys > 1000 </ MaxKeys >
  < Delimiter >/</ Delimiter >
  < IsTruncate  d > false </ IsTruncate  d >
  < Contents >
    < Key > Directory   A / File   C </ Key >
    < LastModifi  ed > 2015 - 11 - 06T09 : 36 : 00 . 000Z </
 LastModifi  ed >
    < ETag >" B026324C69  04B2A9CB4B  88D6D61C81  D1 "</ ETag >
    < Type > Normal </ Type >
    < Size > 2 </ Size >
    < StorageCla  ss > Standard </ StorageCla  ss >
    < Owner >
      < ID > oss </ ID >
      < DisplayNam  e > oss </ DisplayNam  e >
    </ Owner >
  </ Contents >
  < Contents >
    < Key > Directory   A / File   D </ Key >
```

```
    < LastModifi  ed > 2015 - 11 - 06T09 : 36 : 00 . 000Z </
 LastModifi  ed >
    < ETag >" B026324C69  04B2A9CB4B  88D6D61C81  D1 "</ ETag >
    < Type > Normal </ Type >
    < Size > 2 </ Size >
    < StorageCla  ss > Standard </ StorageCla  ss >
    < Owner >
      < ID > oss </ ID >
      < DisplayNam  e > oss </ DisplayNam  e >
    </ Owner >
  </ Contents >
  < CommonPref  ixes >
    < Prefix > Directory   A / Directory   B /</ Prefix >
  </ CommonPref  ixes >
  < CommonPref  ixes >
    < Prefix > Directory   A / Directory   C /</ Prefix >
  </ CommonPref  ixes >
  < CommonPref  ixes >
    < Prefix > Directory   A / Directory   D /</ Prefix >
  </ CommonPref  ixes >
</ ListBucket  Result >
 We   can   see   that :
 Contents   returns   the   second - level   files : " Directory   A /
 File   C "  and  " Directory   A / File   D ".
 CommonPref  ixes   returns   the   second - level   directorie  s :
 " Directory   A / Directory   B /", " Directory   A / Directory   C
 /",  and  " Directory   A / Directory   D /".  The   file   names
 under   these   directorie  s   are   not   shown .
```

Reference

- API: *GetBucket*
- SDK: Java SDK-*List objects in a bucket*

11.3 Copy an object

Copying an object is copying the files in the bucket. In certain situations, you want to copy an object to another bucket, without modifying its content. The standard process is to first download the object, and then upload the object to the new bucket. However, because data is identical for both objects, network bandwidth is wasted. To overcome this issue, OSS provides the CopyObject function to copy objects directly within the OSS without the need to transmit large volumes of data between the user and the OSS.

Additionally, because OSS does not support renaming, we recommend that the OSS CopyObject interface is called for renaming an object. This means you can first copy the original data to an object, apply a new name, and then delete the original file. To only modify an object's Object Meta (object metadata), you can also call the CopyObject interface and set the source address and destination address to the same

value. In this way, the OSS only updates the Object Meta. For more information about Object Meta, see *Object Meta*.

Before copying an object, note the following precautions:

- You must have permissions to operate the source object. Otherwise the operation fails.
- Data cannot be copied across regions. For example, an object in a Hangzhou bucket may not be copied to a Qingdao bucket.
- Objects up to 1 GB are supported.
- Appended objects cannot be copied.

Reference:

- API: *CopyObject*
- SDK: Java SDK-*CopyObject*

Copy large objects

The OSS supports the function of copying large files similar to *Multipart upload*.

The only difference is that the process *UploadPart* is replaced by the process *UploadPartCopy*.

The syntax of *UploadPartCopy* is similar to that of *UploadPart*. However, instead of being directly uploaded from the HTTP request, data is retrieved from the source object.

Reference:

- API: *UploadPartCopy*
- SDK: Java SDK-*Copy a large object*

11.4 Delete an object

You can delete objects that have been uploaded to OSS buckets using one of the following methods:

- Single deletion, in which only a specified object is deleted.
- Batch deletion, in which up to 1,000 objects can be deleted at one time.
- Auto deletion, in which large numbers of objects can be deleted according to certain rules. For example, to regularly delete objects that are created a specified number of days ago, or to regularly empty the entire bucket, we recommend that

you use *Lifecycle management*. Once the rules are specified, OSS uses these rules to recycle expired objects. This reduces the number of user attempts at deletion requests, and helps streamline the deletion process.

Reference

- API: *Delete Object* and *Delete Multiple Objects*
- SDK: Java SDK *Delete Files*
- Console: *Delete Files*

11.5 Manage lifecycle rules

You can configure lifecycle rules for OSS buckets or objects by using the `PutBucketLifecycle` interface to automatically delete expired objects and parts or change the storage class of expired objects to IA or Archive, saving storage costs.

Note:

For more information about the `PutBucketLifecycle` interface, see *PutBucketLifecycle*.

A lifecycle rule includes the following information:

- Matching policy: Specifies the objects and parts that lifecycle rule applies to.
  - Matched by prefix: Indicates that the lifecycle rule applies to objects and parts with the specified prefix. You can create multiple lifecycle rules for objects and parts with different prefixes. The prefix specified in each rule must be different.
  - Matched by tag: Indicates that the lifecycle rule applies to objects with the specified tag key and tag value. You can specify multiple tags for a lifecycle rule so that the rule applies to all objects with these tags. Parts cannot be matched to lifecycle rules by tags.

    Note:

The object tagging function is in the beta testing phase. You can *open a ticket* to apply for the trial of this function. For more information about the object tagging function, see *Object tagging*.

  - Matched by prefix and tag: Indicates that the lifecycle rule applies to objects with the specified prefix and one or more specified tags.
  - Matched to the entire bucket: Indicates that the lifecycle rule applies to all objects and parts in the bucket. You can create only one lifecycle rule that applies to a bucket.

- Object expiration policy: Specifies the expiration time for objects and the operation to be performed on expired objects.
  - Expiration date: Specifies an expiration date and the operation to be performed on expired objects. An object modified before the specified date expires, and the specified operation is performed on the object.

 Note:
Operations that can be performed on an expired object include: convert the storage class of the object to IA, convert the storage class of the object to Archive, and delete the object.

- Part expiration policy: Specifies the expiration time for parts and the operation to be performed on expired parts.
  - Expiration period: Specifies an expiration period (N days). Parts are deleted N days after it is modified for the last time.
  - Expiration date: Specifies an expiration date. All parts modified before the specified date are deleted.

A rule applies to an object if the object name prefix matches the prefix specified in the rule. For example, a bucket includes the following objects:

```
logs / program . log . 1
logs / program . log . 2
logs / program . log . 3
doc / readme . txt
```

If the prefix specified in a rule is "logs/", the rule applies to the three objects prefixed with "logs/". If the prefix specified in the rule is "doc/readme.txt", it applies only to the object named doc/readme.txt.

You can also set overdue deletion rules. For example: if the last date of objects that are prefixed with logs/ is 30 days ago, the objects are deleted according to the specified overdue deletion time.

If an object matches an overdue rule, the OSS includes the x-oss-expiration header in the response to the GetObject or HeadObject requests for the object. The header contains two key-value pairs: expiry-date indicates the expiration date of the object, and rule-id indicates the matched rule ID.

Operation method

Method	Description
OSS console	Web application that is easy to use
Java SDK	Various and complete SDK demos of different languages
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Node.js SDK	
Ruby SDK	

Examples

You can set lifecycle rules for a bucket through OSS APIs. A lifecycle rule is in XML format, as shown in the following example:

```
< LifecycleC  onfigurati  on >
< Rule >
< ID > delete   logs   after   10   days </ ID >
< Prefix > logs /</ Prefix >
< Status > Enabled </ Status >
< Expiration >
< Days > 10 </ Days >
</ Expiration >
</ Rule >
< Rule >
< ID > delete   doc </ ID >
< Prefix > doc /</ Prefix >
< Status > Disabled </ Status >
< Expiration >
< CreatedBef  oreDate > 2017 - 12 - 31T00 : 00 : 00 . 000Z </
 CreatedBef  oreDate >
</ Expiration >
</ Rule >
```

```
< Rule >
< ID > delete   xx = 1 </ ID >
< Prefix > rule2 </ Prefix >
< Tag >< Key > xx </ Key >< Value > 1 </ Value ></ Tag >
< Status > Enabled </ Status >
< Transition >
< Days > 60 </ Days >
< StorageCla  ss > Archive </ StorageCla  ss >
</ Transition >
</ Rule >
</ LifecycleC  onfigurati  on >
```

Elements in the preceding example are described as follows:

- `< ID >`: Indicates the unique identifier for a rule.
- `< Status >`: Indicate the status of the lifecycle rule with two values: Enabled or Disabled. Only rules with the Enabled status is applied.
- `< Prefix >`: Indicates that the rule applies only to objects with the specified prefix.
- `< Expiration >`: Indicates the operation expiration period. The sub-element `< CreatedBef  oreDate >` or `< Days >` indicates the absolute and relative expiration time respectively.
  - CreatedBeforeDate: Specifies an expiration date and the operation to be performed on expired objects. An object modified before the specified date expires, and the specified operation is performed on the object.
  - Days: Specifies an expiration period (N days) and the operation to be performed on expired objects. An object expires N days after it is modified for the last time, and the specified operation is performed on the object.

The first rule indicates that objects that are prefixed with logs/ and were modified 10 days ago are deleted. The second rule indicates that objects that are prefixed with doc/ and were modified before December 31, 2014 are deleted. However, the rule does not take effect because it is in Disabled status. The third rule indicates that the storage class of objects that are tagged with "xx=1" and were modified 60 days ago is converted to Archive.

Detail analysis

- Prefix and tag
  - The naming conventions for a prefix are the same as those for an object.
  - If the prefix in a rule is empty, the rule applies to all objects in the bucket.
  - Prefixes specified in rules for a bucket must be unique. For example, if two rules are configured for a bucket and the prefixes in the rules are logs/ and logs/program respectively, OSS returns an error.
  - The key of a tag cannot be empty. A tag can include letters, numbers, spaces, and the following symbols:

    + - = . _ : /
  - The prefixes in the matching policies (matched by prefix and tag) of different rules can be overlapped. For example, assume that rule 1 and rule 2 are configured for the same bucket. In rule 1, the specified prefix is `logs /` and the specified tag is "K1=V1". In rule 2, the specified prefix is `logs / program` and the specified tag is "K1=V2". Then, rule 1 applies to objects with the `logs` or `logs / program` prefix and the "K1=V1" tag. Rule 2 applies to objects with the `logs` prefix and the "K1=V1" tag.
- Effective time
  - If a rule is set to delete objects on a specified date, the date must be midnight UTC and comply with the ISO8601 format, for example, 2017-01-01T00:00:00.000Z. In this example, OSS deletes matched objects after midnight on January 1, 2017.
  - If a rule is set to delete objects after a specified number of days, OSS sums up the last update time (Last-Modified) and the specified number of days, and then round the sum to the midnight UTC timestamp. For example, if the last update time of an object is 01:00 a.m. on April 12, 2014 and the number of days specified in the matched rule is 3, the expiry time is zero o'clock on April 16, 2017.
  - OSS deletes the objects matched the rule at the specified time. Note that objects are usually deleted shortly after the specified time.
  - The update time of an unmodified object is typically the time of its creation. If an object undergoes the put operation for multiple times, the last update time is the time of the last Put operation. If an object was copied to itself, the last update time is the time at when the object was last copied.

- Fees

  Only successful lifecycle asynchronous request operations are recorded and charged.

- Actions performed when rules conflict

  - The prefixes and tags specified in two rules are the same.

Rule	Prefix	Tag	Action
rule1	abc	a=1	Matched objects are deleted after 20 days.
rule2	abc	a=1	The storage class of matched objects is converted to Archive after 20 days.

  Result: Objects with the abc prefix and the "a=1" tag are deleted after 20 days (deletion is performed first). The second rule does not take effect because the matched objects have already been deleted.

Rule	Prefix	Tag	Action
rule1	abc	a=1	The storage class of matched objects is converted to IA after 365 days.

Rule	Prefix	Tag	Action
rule2	abc	a=1	The storage class of matched objects is converted to Archive before March, 1, 2018.

Result: If objects with the abc prefix and the "a=1" tag match the two rules at the same time, the storage of the objects is converted to Archive. If the objects only match one of the rules, the action specified in the matched rule is performed.

- The tags specified in two rules are the same, and the prefixes specified in the rules are overlapped.

Rule	Prefix	Tag	Action
rule1		a=1	The storage class of matched objects is converted to IA after 20 days.
rule2	abc	a=1	Matched objects are deleted after 120 days.

Result: The storage class of all objects with the "a=1" tag is converted to IA after 20 days. Objects with the abc prefix and the "a=1" tag are deleted after 120 days.

Rule	Prefix	Tag	Action
rule1		a=1	The storage class of matched objects is converted to Archive after 10 days.

Rule	Prefix	Tag	Action
rule2	abc	a=1	The storage class of matched objects is converted to IA after 20 days.

Result: The storage class of all objects with the "a=1" tag is converted to Archive. The second rule does not take effect on objects with the abc prefix and the "a=1" tag because the storage class of Archive objects cannot be converted to IA.

FAQ

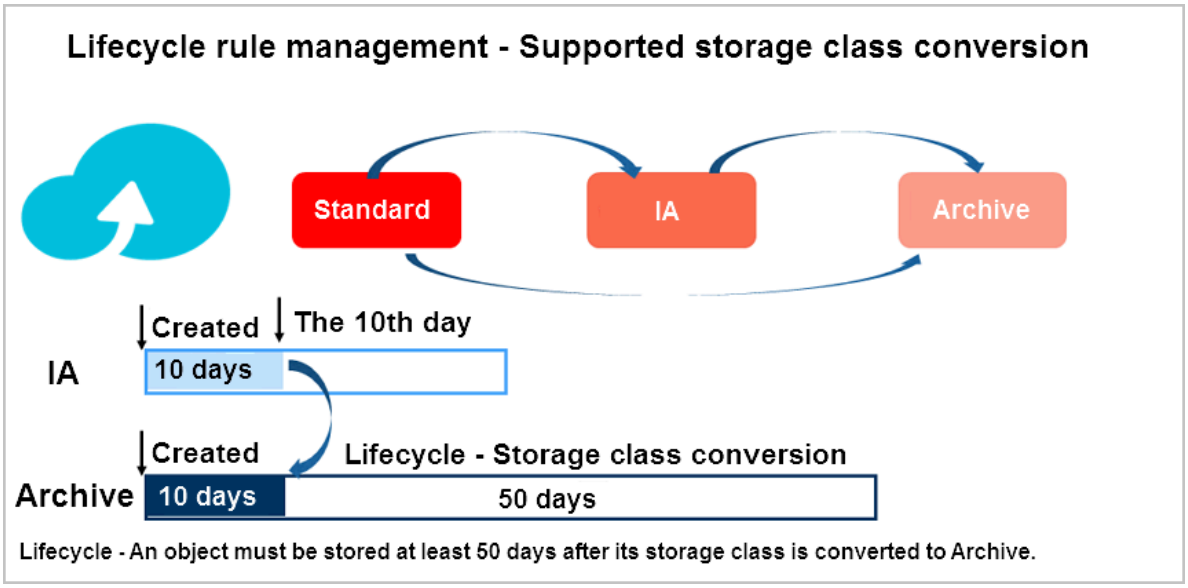
- Is there a minimum storage period for a lifecycle rule if the rule is set to convert storage class or delete an expired object?

  If a lifecycle rule is set to convert the storage class (from Standard to IA or Archive or from IA to Archive) of objects, no minimum storage period is required for the rule. However, a lifecycle rule set to delete expired objects requires a minimum storage period. The minimum storage period is 30 days for objects of the IA storage class and 60 days for objects of the Archive storage class.

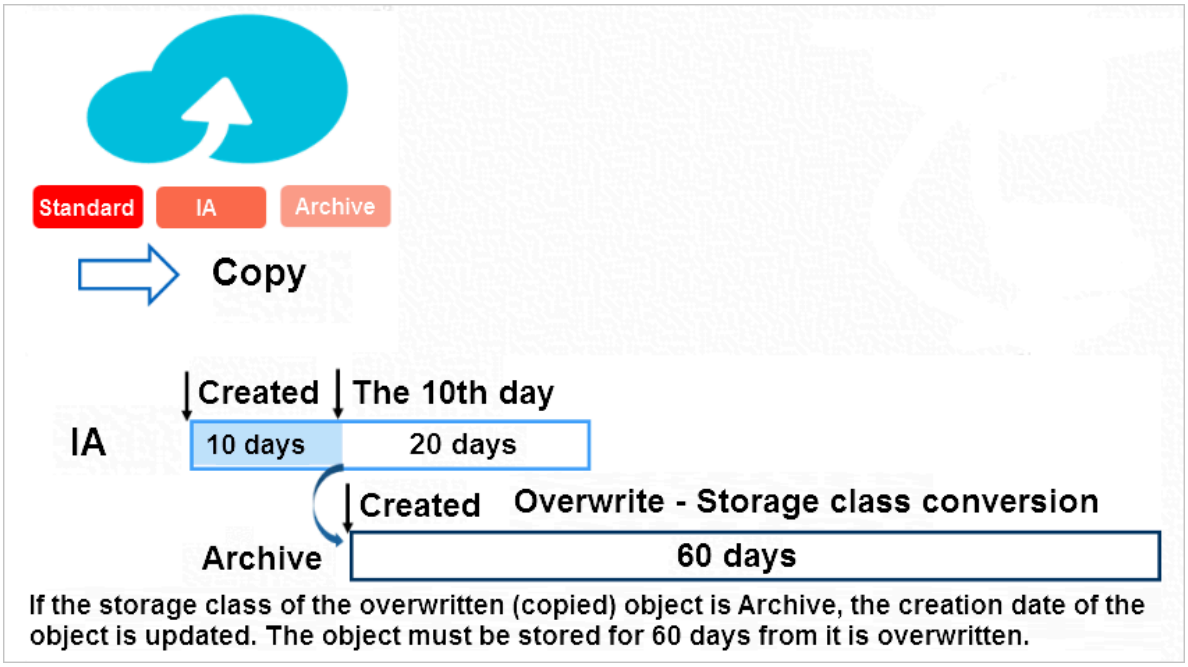
  The minimum storage period for a lifecycle rule set to delete an expired object indicates the time period from the time when the object is created until the time when the object is deleted. If the object is modified (such as by CopyObject or

AppendObject operations) during this period, the minimum storage period is calculated from the last modified date.

Example 1: The storage class of an object is converted from IA to Archive 10 days after it is created. The creation time of the object does not change. The converted object must be stored for at least 50 days.



Example 2: If the storage class of an object is converted from IA to Archive through a CopyObject operation 10 days after it is created, fees of 20 days are incurred for the deletion. The creation time of the object updates. The converted object must be stored for at least 60 days.



- Billing logic for requests

  The conversion of storage class or the deletion of expired objects performed according to lifecycle rules generate requests. The requests are charged by OSS. For example:

  - If the storage class of 1000 objects are converted from Standard to Archive according to a lifecycle rule, 1000 POST requests are generated.
  - If 1000 expired objects are deleted according to a lifecycle rule, 1000 Delete requests are generated.

  For more information, see *Billing methods*.

- Are the conversion of storage class or the deletion of expired objects performed according to lifecycle rules recorded?

  The conversion of storage class or the deletion of expired objects performed according to lifecycle rules are recorded in logs. Fields in the logs are described as follows:

  - Operation
    - CommitTransition: Indicates that the rule is set to convert the storage class of objects.
    - ExpireObject: Indicates that the rule is set to delete expired objects.
  - Sync Request
    - lifecycle: Indicates the operation to be performed by the lifecycle rule.

11.6 Manage back-to-origin settings

Configuring back-to-origin rules allows OSS to retrieve requested data from origin sites in multiple ways, meeting the requirements of frequently accessed (hot) data migration and specific request redirection requirements.

This setting enables the URL of each OSS GET request to be matched, which then specifies an origin retrieval method. A maximum of five rules can be configured. Requests are compared to the rules in a set sequence, until matched to a valid rule. The specified method can be either mirroring or redirection.

Mirroring

The process is as follows: A client requests data of an object. OSS determines the object does not exist, and forwards the request to the source URL. The source URL returns the object through the OSS, which goes to the client. OSS simultaneously writes the object data to process future requests.

Example scenario

Mirroring write-back is designed to seamlessly migrate data to OSS. This means any service that is already running on a user-established site, or on another cloud product , can be migrated to OSS without interruption to the service. A detailed scenario is as follows:

- An origin site is generating new hot data, and also has legacy cold data stored.

  First, a user can use the migration tool *ossimport* to migrate cold data to the OSS. During migration, the user can configure mirroring write-back and set the origin site’s URL to OSS. Even if some newly generated data does not migrate when the domain name is switched to the OSS, the user can still access it through OSS and the files are saved to OSS after they have been accessed for the first time. After switching the domain name for an origin site that no longer produces new data, the site is scanned, and all non-migrated data is imported to the OSS. In this situation, the user may disable mirroring write-back.

  If the configured origin site is an IP address, after the domain name is migrated to the OSS, data can still be mirrored to the origin site.
- However, if it is a domain name, no mirroring can be produced because the domain name is resolved to the OSS or CDN. In this situation, the user can apply for another domain name to mirror the origin site.

  This domain name and the in-service domain name would both be resolved to the same IP address. This allows origin site imaging to continue when the service domain name is migrated.

Usage rules

- OSS only executes mirroring write-back to request an object from the origin site when GetObject() returns a 404 code.
- The URL requested from the origin site is `MirrorURL + object` and the name of the file written back to the OSS is object. For example, assume that: A bucket is named `example - bucket` . Mirroring write-back is configured. The MirrorURL is `http :// www . example - domain . com /`. The file `object . jpg` does not exist in this bucket. To download the file, the OSS initiates a GET request to `http :// www . example . com / object . jpg` , records the result, and returns it to the user. The file is then available on OSS as `object . jpg` . This is the same as migrating an object with the same name to the OSS. Note that if the MirrorURL carries path information, such as `http :// www . example - domain . com / dir1 /`, the process is the same as the preceding example, but the OSS origin retrieval URL is `http :// www . example - domain . com / dir1 / image / example_ob ject . jpg` although the object written to the OSS remains as `object . jpg` . This process is the same as migrating an object from an origin site directory to the OSS.
- The header and querystring information transmitted to the OSS is not sent to the origin site.
- If the origin site returns data in chunks, the OSS returns data to the user in chunks.
- The OSS returns and saves the following header information from the origin site:

```
Content - Type
Content - Encoding
Content - Dispositio  n
Cache - Control
Expires
Content - Language
Access - Control - Allow - Origin
```

- An x-oss-tag response header is added to mirroring write-back files, with the value “MIRROR” + space + url_decode (origin retrieval URL). In the proceeding example, this would be

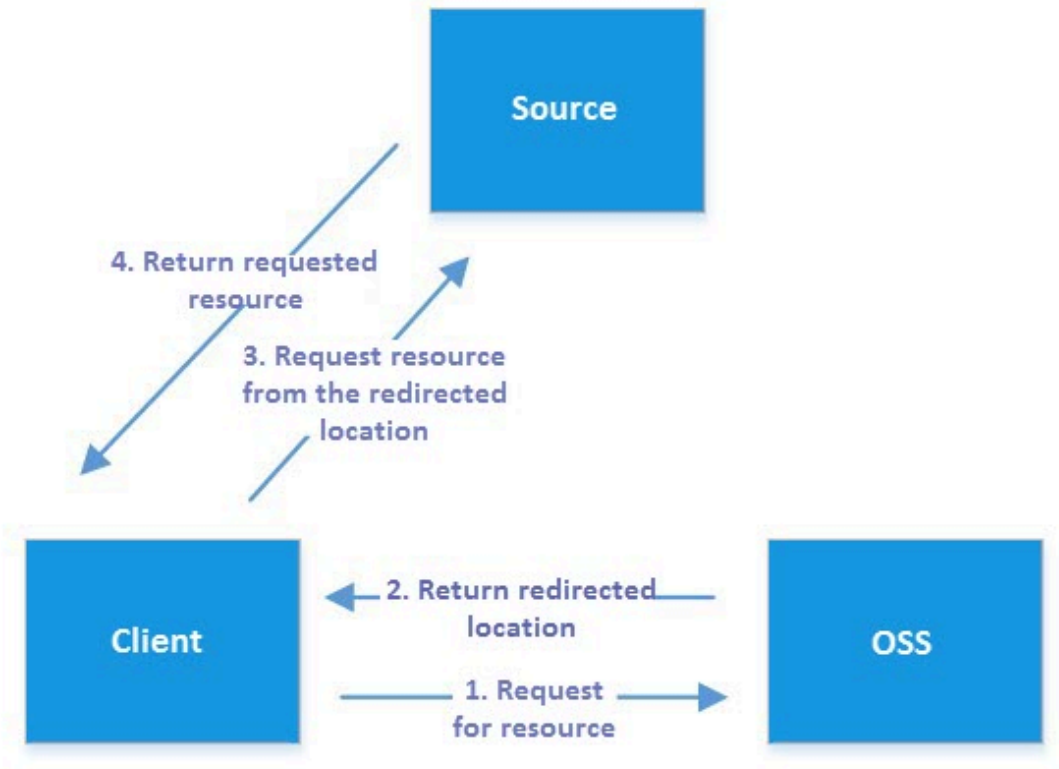
```
x - oss - tag : MIRROR   http % 3a % 2f % 2fwww . example - domain
. com % 2fdir1 % 2fimage % 2fexample_  object . jpg
```

After the file is written back to the OSS, so long as it is not overwritten again, this header is added each time it is downloaded to indicate that it is taken from mirroring.

- Assuming that the file has already been written to the OSS through mirroring write-back, if the corresponding file on the origin site is changed, the OSS does not update the file that exists on the OSS because this file which is already present on the OSS does not meet the mirroring write-back conditions.
- If the file does not exist in the mirroring source, the returned result is the HTTP status 404, which is forwarded through the OSS to the user. If the mirroring source returns another non-200 status code (including file retrieval failure due to network-related causes), the OSS returns 424 to the user, the error code for `MirrorFail ed`.

Redirection

The URL redirection function returns a 3xx hop to the user based on user-defined conditions and corresponding hop configurations. Users can use this hop function to redirect files and provide various services based on this action. The process is as follows:



Application scenarios

- Migrate data sources to OSS

  Users can asynchronously migrate data to the OSS. In this way, requests for un-migrated data use the URL rewrite method to return a 302 redirect request to the user. The user’s client then returns the data from the user’s data source based on the location in the 302 redirect request.
- Configure page redirect function

  If a user wants to hide objects using a certain header prefix, a customized page can be displayed to visitors.
- Configure the redirected page when a 404 or 500 error occurs

  If a 404 or 500 error occurs, the user can be redirected to a live page. This makes sure that OSS errors are undetected by a user.

Reference

- Console: *Origin retrieval Rule Management*

11.7 SelectObject

SelectObject is commonly used in log file analysis and can be used together with Alibaba Cloud Bigdata products. This topic describes how to use the Python SDK and Java SDK of SelectObject to achieve the preceding application scenarios.

Introduction

Object Storage Service (OSS) built on Alibaba Cloud’s Apsara distributed system is a massive, secure, and highly reliable cloud storage solution that offers low cost storage accessible anywhere in the world. OSS possesses excellent scaling abilities for storage capacity and processes, and supports RESTful APIs. Not only can OSS store media files, but it can also be utilized as a data warehouse for massive data file storage . OSS can seamlessly integrate with Hadoop 3.0, and services that are run on EMR (such as Spark/Hive/Presto, MaxCompute, HybridDB and the newly-released Data Lake Analytics) support data processing and retrieval directly from OSS.

However, the current GetObject interface provided by OSS determines that the big data platform can only download all OSS data locally and then for analysis and filter. A lot of bandwidth and client resources are wasted in querying scenarios.

To address this problem, the SelectObject interface is provided. This method allows big data platforms to access OSS to perform basic filtering on data through conditions

and Projection, and return useful data only to the big data platform. In this way, the bandwidth and the amount of data processed at the client-side is greatly reduced, making OSS-based data warehousing and data analysis a highly attractive option.

SelectObject provides Java and Python SDKs. SDKs in other langugae will be available soon. SelectObject supports JSON files and CSV files that conform to the RFC 4180 standard and are encoded in UTF-8 (including Class CSV files such as TSV, in which row and column separators and quote characters can be customized). SelectObject supports objects of the standard and IA storage class access storage types (objects of the Archive storage class must be restored before use). SelectObject also supports files encrypted in the following two methods: 1. Use encryption fully managed by OSS . 2. Use CMKs managed by KMS for encryption.

The following two types of JSON files are supported: DOCUMENT and LINES. A JSON file of the DOCUMENT type is a single JSON object. A JSON file is composed of lines of JSON objects separated by delimiters. However, the file itself may not be a valid JSON object. SelectObject supports common delimiters, such as "\n" and "\r\n" so that you do not need to specify the delimiters.

- Supported SQL syntax:
  - SQL statements: Select, From, and Where
  - Data types: string, int (64bit), double (64bit), decimal (128), timestamp, and bool
  - Operations: logical conditions (AND, OR, and NOT), arithmetic expressions (+, -, *, /, and %), comparison operations (>, =, <, >=, <=, and !=), string operations (LIKE, and ||)
- Multipart query

  SelectObject provides the multipart query mechanism that is similar to the multipart download mechanism of GetObject. Data are divided into parts by rows or splits. Dividing data by rows are commonly used but may result in unbalanced workloads when dividing sparse data. Dividing data by splits is more efficient because the size of each split (which includes multiple rows) is roughly the same, which enables better load balancing performance.

- Data type

  The type of CSV data is string in OSS by default. You can use the CAST function to convert the data type. For example, the following SQL query statement converts _1 and _2 into int and compares them.

  ```
  Select * from OSSObject where cast ( _1 as int ) > cast ( _2 as int )
  ```

  Furthermore, SelectObject supports implicit type conversion in the WHERE condition. For example, the first and the second columns in the following statement are converted to int:

  ```
  Select _1 from ossobject where _1 + _2 > 100
  ```

  If the CAST function is not specified in the SQL statement, the type of a JSON file is determined by the data type in the file. A standard JSON file can include the following built-in data types: null, bool, int64, double, and string.

Python SDK example

```
import os
import oss2

def select_cal l_back ( consumed_b ytes , total_byte s =
None ):
 print (' Consumed Bytes :' + str ( consumed_b ytes ) + '\ n
')

# Initialize s OSS informatio n , such as AccessKeyI d ,
 AccessKeyS ecret , and Endpoint .
# Obtains environmen t variables or replace variables
 such as < yourAccess KeyId > with actual values .
#
# This example uses the China East 1 ( Hangzhou )
 region , which has the following endpoints :
# http :// oss - cn - hangzhou . aliyuncs . com
# https :// oss - cn - hangzhou . aliyuncs . com

access_key _id = os . getenv (' OSS_TEST_A CCESS_KEY_ ID ', '<
 yourAccess KeyId >')
access_key _secret = os . getenv (' OSS_TEST_A CCESS_KEY_ SECRET
', '< yourAccess KeySecret >')
bucket_nam e = os . getenv (' OSS_TEST_B UCKET ', '< yourBucket
>')
endpoint = os . getenv (' OSS_TEST_E NDPOINT ', '< yourEndpoi nt
>')

# Creates an OSS bucket . All object - related methods
 must be called by the bucket .
bucket = oss2 . Bucket ( oss2 . Auth ( access_key _id ,
 access_key _secret ), endpoint , bucket_nam e )
key = ' python_sel ect . csv '
content = ' Tom Hanks , USA , 45 \ r \ n '* 1024
filename = ' python_sel ect . csv '
```

```
#  Uploads   a   CSV   file .
 bucket . put_object ( key ,  content )
#  Configure   the   parameters   of   SelectObje  ct .
 csv_meta_p  arams  = {' CsvHeaderI  nfo ': ' None ',
' RecordDeli  miter ': '\ r \ n '}
 select_csv  _params  = {' CsvHeaderI  nfo ': ' None ',
' RecordDeli  miter ': '\ r \ n ',
' LineRange ': ( 500 ,  1000 )}

 csv_header  =  bucket . create_sel  ect_object  _meta ( key ,
 csv_meta_p  arams )
 print ( csv_header . rows )
 print ( csv_header . splits )
 result  =  bucket . select_obj  ect ( key , " select  *  from
 ossobject   where   _3  >  44 ",  select_cal  l_back ,  select_csv
 _params )
 select_con  tent  =  result . read ()
 print ( select_con  tent )

 result  =  bucket . select_obj  ect_to_fil  e ( key ,  filename ,
" select  *  from   ossobject   where   _3  >  44 ",  select_cal
 l_back ,  select_csv  _params )
 bucket . delete_obj  ect ( key )

### JSON   DOCUMENT
 key  =  ' python_sel  ect . json '
 content  =  "{\" contacts \":[{\" key1 \": 1 ,\" key2 \":\" hello
 world1 \"},{\" key1 \": 2 ,\" key2 \":\" hello   world2 \"}]}"
 filename  =  ' python_sel  ect . json '
#  Upload   a   JSON   DOCUMENT   file
 bucket . put_object ( key ,  content )
 select_jso  n_params  = {' Json_Type ': ' DOCUMENT '}
 result  =  bucket . select_obj  ect ( key , " select   s . key2
 from   ossobject . contacts [*]  s   where   s . key1  =  1 ",  None
 ,  select_jso  n_params )
 select_con  tent  =  result . read ()
 print ( select_con  tent )

 result  =  bucket . select_obj  ect_to_fil  e ( key ,  filename ,
" select   s . key2   from   ossobject . contacts [*]  s   where   s .
 key1  =  1 ",  None ,  select_jso  n_params )

 bucket . delete_obj  ect ( key )
### JSON   LINES
 key  =  ' python_sel  ect_lines . json '
 content  =  "{\" key1 \": 1 ,\" key2 \":\" hello   world1 \"}\ n {\"
 key1 \": 2 ,\" key2 \":\" hello   world2 \"}"
 filename  =  ' python_sel  ect . json '
#  Uploads   a   JSON   LINE   file .
 bucket . put_object ( key ,  content )
 select_jso  n_params  = {' Json_Type ': ' LINES '}
 json_heade  r  =  bucket . create_sel  ect_object  _meta ( key ,
 select_jso  n_params )
 print ( json_heade  r . rows )
 print ( json_heade  r . splits )

 result  =  bucket . select_obj  ect ( key , " select   s . key2
 from   ossobject   s   where   s . key1  =  1 ",  None ,  select_jso
 n_params )
 select_con  tent  =   result . read ()
 print ( select_con  tent )
 result  =  bucket . select_obj  ect_to_fil  e ( key ,  filename ,
" select   s . key2   from   ossobject   s   where   s . key1  =  1
 ",  None ,  select_jso  n_params )
```

```
bucket . delete_obj  ect ( key )
```

SelectObject API in Python

- select_object
  - select_object example:

```
def   select_obj  ect ( self ,  key ,  sql ,
                    progress_c  allback = None ,
                    select_par  ams = None                    ):
```

The preceding example code executes the SQL statement on the target CSV files and returns the query results.

- The SQL statements can be directly used as the SQL parameter and does not need to be base64-encoded.
- The progress_callback parameter indicates a callback function used to report the query progress, which is optional.
- The select_params parameters specifies the parameters and actions of the SelectObject operation.
- Headers can be used to specify the header information included in the request, which act the same role as they do in the GetObject operation. For examples, you can set the bytes header to specify the query range.

  - Parameters supported by select_params

Parameter	Description
Json_Type	■ Not specified: The file is a CSV file by default. ■ DOCUMENT: The file is a JSON file. ■ LINES: The file is a JSON LINE file.
CsvHeaderInfo	Specifies the header information about the CSV file. Valid values: `None` , `Ignore` , `Use` ■ None: This file does not include header information. ■ Ignore: This file includes header information but is not used in the SQL statement. ■ Use: This file includes header information. The column nmaes are used in the SQL statement.

Parameter	Description
CommentCharacter	Specifies the comment character in the CSV file. A comment character can be only one character. The value of this parameter is None, that is, no comment character is specified.
RecordDelimiter	Specifies the delimiter used to separate rows in the CSV file. The value of this parameter can be two characters in maximum and is `\ n` by default.
OutputRecordDelimiter	Specifies the line breaks in the output CSV file. The default value of this parameter is `\ n` .
FieldDelimiter	Specifies the delimiter used to separate columns in the CSV file. The value of this parameter can be only one character and is , by default.
OutputFieldDelimiter	Specifies the delimiter used to separate columns in the output CSV file. The value of this parameter is , by default.
QuoteCharacter	Specifies the quote characters used in the CSV file. The value of this parameter can be only one character and is double quotation mark by default . Delimiters enclosed in the quotation marks are processed as normal characters.
SplitRange	Specifies the split range in multipart queries. The value of this parameter is in the (start, end) format. The range includes the start and end values, indicating that splits from the `start` to the `end` are queried.
LineRange	Specifies the row range in multipart queries. The value of this parameter is in the (start, end) format. The range includes the start and end values, indicating that rows from the `start` to the `end` are queried.
CompressionType	Specifies the compression type. The default value of this parameter is None. You can set the value to GZIP.

Parameter	Description
KeepAllColumns	Indicates that all columns in the CSV file are included in the returned result. However, only columns included in the select statement have values. The default value of this parameter is false. For example: A CSV file includes the following columns: firstname, lastname, age. And the SQL statement is `select firstname , age from ossobject`. If the value of KeepAllColumns is true, the output result is `firstname ,, age`, in which a comma is added to indicate the lastname column that is not specified in the statement. If the value of KeepAllColumn is false, the output result is `firstname , age`. This paremeter is used to allow the code that processes the data returned by GetObject to process the data returned by SelectObject without being modified.
OutputRawData	■ If the value of this parameter is true, the data returned by the SelectObject is output without being encapsulated by frames. However, if data is not returned for a long period, a timeout error may occur. ■ If the value of this parameter is false, the output data is encapsulated by frames. The default value of this parameter is false.
EnablePayloadCrc	Indicates that CRC values are calculated for each frame. The default value of this parameter is false.
OutputHeader	Specifies that the first row in the output result is the header information. This parameter only applies to CSV files.

Parameter	Description
SkipPartialDataRecord	■ If the value of this parameter is true, the current record is skipped when the value of a column is missed on a CSV file or a key in the JSON file does not exist. ■ If the value of this parameter is false, the value of a column without value is processed as null. For example: If a row includes the following columns: firstname, lastname, and age, and the SQL statement is `select _1 , _4 from ossobject`. If the value of this parameter is true, this row is skipped. If the value of this parameter is false, the following result is returned: firstname,\n.
MaxSkippedRecordsAllowed	Specifies the maximum number of skipped rows. The default value of this parameter is 0, indicating that an error is returned if a row is skipped.
ParseJsonNumberAsString	Numbers in the JSON file are resolved as strings if the value of this parameter is true and are resolved as integers or floats if the value is false. The accuracy of float numbers in a JSON file degrades when the numbers are resolved as float. To keep the accuracy, you can set the value of this parameter to true and convert the type of the column into decimal.

- Returned result of select_object: A SelectObjectResult object is returned. You can use the read() function or the __iter___ method to obtain all results. If the size of the result is large, the read() function is not the optimal method to obtain the result because this function blocks until all results are returned and use excessive memory resources.

  We recommend that you use the __iter__ method to obtain results and process each chunk. This method uses less memory resources and allows you to process each chunk immediately after it is processed by the OSS server instead of waiting for all the results to be returned.

· select_object_to_file

```
def select_obj ect_to_fil e ( self , key , filename , sql ,
                progress_c allback = None ,
                select_par ams = None
```

```
                    ):
```

The preceding code executes the SQL statement on a file that includes the specified key and writes the query result into the specified file.

Other paraemters are the same as those of *select_object*.

- create_select_object_meta
  - select_meta_params

```
def   create_sel  ect_object  _meta ( self ,  key ,  select_met
a_params = None ):
```

The preceding code create select meta for a file that includes the specified key or obtain select meta from a file that includes the specified key. Select meta includes the following information about the file: total number of rows, total number of columns (for CSV files), and total number of splits.

If metadata has been created for the file, this function does not re-create the metadata unless the value of OverwriteIfExists is set to true.

To create select meta for a file, the file must be completely scanned

  - Parameters supported by select_meta_params

Parameter	Description
Json_Type	■ Not specified: The file is a CSV file. ■ LINES: The file is a JSON LINES file. ■ DOCUMENTS: Not supported.
RecordDelimiter	Specifies the delimiter used to separate rows in CSV files.
FieldDelimiter	Specifies the delimiter used to separate columns in CSV files.
QuoteCharacter	Specifies the quote characters used in CSV files. Delimiters enclosed in the quotation marks are processed as normal characters.
CompressionType	Specifies the compression typo. Only None is supported.

Parameter	Description
OverwriteIfExists	Indicates that the operation creates new select meta for the file and overwrites the original select meta. This parameter is unnecessary in common scenarios.

- Returned result of create_select_object_meta: A GetSelectObjectMetaResult object is returned, which includes two attributes: rows and splits. The select_resp object in the result includes the columns attribute that indicates the number of columns in the CSV file.

Java SDK sample

```
package   samples ;

import   com . aliyun . oss . ClientBuil  derConfigu  ration ;
import   com . aliyun . oss . model .*;
import   com . aliyun . oss . OSS ;
import   com . aliyun . oss . OSSClientB  uilder ;


import   java . io . IOExceptio  n ;
import   java . util . ArrayList ;
import   java . util . List ;
import   java . util . concurrent . Callable ;
import   java . util . concurrent . ExecutorSe  rvice ;
import   java . util . concurrent . Executors ;
import   java . util . concurrent . Future ;

import   com . aliyun . oss . common . auth .*;
import   com . aliyuncs . DefaultAcs  Client ;
import   com . aliyuncs . exceptions . ClientExce  ption ;
import   com . aliyuncs . http . MethodType ;
import   com . aliyuncs . http . ProtocolTy  pe ;
import   com . aliyuncs . profile . DefaultPro  file ;
import   com . aliyuncs . profile . IClientPro  file ;
import   com . aliyuncs . sts . model . v20150401 . AssumeRole
Request ;
import   com . aliyuncs . sts . model . v20150401 . AssumeRole
Response ;

import   java . text . SimpleDate  Format ;
import   java . util . Calendar ;

/**
 *  Examples   of   create   select   object   metadata   and   select
   object .
 *
 */
class   MulipartSe  lector   implements   Callable < Integer > {

    private    OSS   client ;
    private    String   bucket ;
    private    String   key ;
    private    int   start ;
    private    int   end ;
    private    String   sql ;
```

```
    public   MulipartSe  lector ( OSS   client ,  String   bucket ,
 String   key ,  int   start ,  int   end ,  String   sql ){
        this . client  =  client ;
        this . bucket  =  bucket ;
        this . key  =  key ;
        this . start  =  start ;
        this . end  =  end ;
        this . sql  =  sql ;
   }

   @ Override
    public   Integer   call ()  throws   Exception  {
        SelectObje  ctRequest   selectObje  ctRequest  =
                new   SelectObje  ctRequest ( bucket ,  key )
                        . withInputS  erializati  on (
                                new   InputSeria  lization ().
 withCsvInp  utFormat (
                                        new   CSVFormat ().
 withHeader  Info ( CSVFormat . Header . None ). withRecord  Delimiter
 ("\ n ")
                                                . withFieldD  elimiter
 ("|")))
                        . withSplitR  ange ( start ,  end )
                        . withOutput  Serializat  ion ( new
 OutputSeri  alization (). withCsvOut  putFormat ( new   CSVFormat
 ()). withCrcEna  bled ( true ));

        selectObje  ctRequest . setExpress  ion ( sql );
        OSSObject   ossObject  =  client . selectObje  ct (
 selectObje  ctRequest );
        byte []  buffer  =  new   byte [ 4096 ];
        int   bytesRead ;
        int   totalSize  =  0 ;
        try  {
            while  (( bytesRead  =  ossObject . getObjectC  ontent
 (). read ( buffer )) != - 1 ) {
                totalSize  +=  bytesRead ;
            }
            String   result  =  new   String ( buffer ,  0 ,
 totalSize  -  1 );
            return   new   Integer ( Integer . parseInt ( result ));
       }
        catch  ( IOExceptio  n   e ){
            System . out . println ( e . toString ());
            return   new   Integer ( 0 );
       }
   }
}
 class   RoleCreden  tialProvid  er  {
    public   static   final   String   REGION_CN_  HANGZHOU  = " cn -
 hangzhou ";
   //  Current   STS   API   version
    public   static   final   String   STS_API_VE  RSION  = " 2015 -
 04 - 01 ";
    public   static   final   String   serviceAcc  essKeyId  = "<
 access   Key   Id   that   can   do   assume   role >";
    public   static   final   String   serviceAcc  essKeySecr  et  =
 "< access   key   secret >";

    public   static   final   long   DurationSe  conds  =  15  *  60
 ;

    private   Credential  s   credential ;
```

```
    private   Calendar   expireTime ;

    private   String   roleArn ;
    private   DefaultAcs  Client   client ;

    public   RoleCreden  tialProvid  er ( String   roleArn )  throws
  InvalidCre  dentialsEx  ception  {
        this . roleArn  =  roleArn ;
   }

    private   AssumeRole  Response   assumeRole  ( String
accessKeyI  d ,  String   accessKeyS  ecret ,  String   roleArn ,
String   roleSessio  nName ,  String   policy ,  ProtocolTy  pe
protocolTy  pe ,  long   durationSe  conds )  throws   ClientExce
ption  {
        try  {
           //  create   Aliyun   Acs   Client   for   sending
OpenAPI   requests
            if  ( this . client  ==  null ) {
                IClientPro  file   profile  =  DefaultPro  file .
getProfile ( REGION_CN_  HANGZHOU ,  accessKeyI  d ,  accessKeyS
ecret );
                this . client  =  new   DefaultAcs  Client ( profile
);
            }
            //  create   an   instance   of   AssumeRole  Request
and   set   properties
             final   AssumeRole  Request   request  =  new
AssumeRole  Request ();
             request . setVersion ( STS_API_VE  RSION );
             request . setMethod ( MethodType . POST );
             request . setProtoco  l ( protocolTy  pe );
             request . setRoleArn ( roleArn );
             request . setRoleSes  sionName ( roleSessio  nName );
             request . setPolicy ( policy );
             request . setDuratio  nSeconds ( durationSe  conds );
            //  send   request   and   get   the   response
             final   AssumeRole  Response   response  =  client .
getAcsResp  onse ( request );
             return   response ;
       }  catch  ( ClientExce  ption   e ) {
             throw   e ;
       }
   }


    public   Credential  sProvider   GetCredent  ialProvide  r ()
             throws   IOExceptio  n  {
       //  Parameters   in   AssumeRole   requests :  RoleArn ,
RoleSessio  nName ,  Policy ,  and   DurationSe  conds
       //  RoleArn   can   be   obtained   on   the   RAM   console .
       //  RoleSessio  nName   is   the   name   of   the   session
  for   the   temporary   token .  It   is   used   to   identify
users   in   audit   or   identify   users   that   you   want   to
issue   tokens   to .
       //  RoleSessio  nName   can   only   include   letters ,
numbers ,'-'  and   ' _ '  and   cannot   include   spaces .
       //  For   more   informatio  n   about   the   format ,  see
the   API   documentat  ion .
        SimpleDate  Format   timeFormat  =  new   SimpleDate  Format
(" yyyy - MM - dd ");
        String   roleSessio  nName  = " AssumingRo  le " +
timeFormat . format ( Calendar . getInstanc  e (). getTime ());
       //  read   OSS   data
```

```
         String   policy  = "{\ n " +
                "    \" Version \": \" 1 \", \ n " +
                "    \" Statement \": [\ n " +
                "        {\ n " +
                "            \" Action \": \" oss :*\", \ n " +
                "            \" Resource \": [\ n " +
                "                \" acs : oss :*:*:*\"\ n " +
                "            ], \ n " +
                "            \" Effect \": \" Allow \"\ n " +
                "        }\ n " +
                "    ]\ n " +
                "}";
        //  The   protocal   type   must   be   set   to   HTTPS .
         ProtocolTy  pe   protocolTy  pe  =  ProtocolTy  pe . HTTPS ;
         try  {
             final   AssumeRole  Response   response  =  assumeRole (
 serviceAcc  essKeyId ,  serviceAcc  essKeySecr  et ,
                     roleArn ,  roleSessio  nName ,  policy ,
 protocolTy  pe ,  DurationSe  conds );
             String   ossAccessI  d  =  response . getCredent  ials
 (). getAccessK  eyId ();
             String   ossAccessK  ey  =  response . getCredent  ials
 (). getAccessK  eySecret ();
             String   ossSts  =  response . getCredent  ials ().
 getSecurit  yToken ();

             return   new   DefaultCre  dentialPro  vider (
 ossAccessI  d ,  ossAccessK  ey ,  ossSts );

        }  catch  ( ClientExce  ption   e ) {
             throw   new   InvalidCre  dentialsEx  ception (" Unable
 tp   get   the   temporary   AK :" +  e );
        }
    }

     public   void   setClient ( DefaultAcs  Client   client ) {
         this . client  =  client ;
    }

     public   void   setCredent  ials ( Credential  s   creds ) {
         this . credential  =  creds ;
    }

     public   Credential  s   getCredent  ials () {
         if  ( credential  !=  null  &&  expireTime . after ( Calendar
 . getInstanc  e ())) {
             return   credential ;
        }

         try  {
             Credential  sProvider   provider  =  GetCredent
 ialProvide  r ();
             credential  =  provider . getCredent  ials ();
             expireTime  =  Calendar . getInstanc  e ();
             expireTime . add ( Calendar . SECOND , ( int )
 DurationSe  conds  -  60 );
             return   credential ;
        }  catch  ( IOExceptio  n   e ) {
             throw   new   InvalidCre  dentialsEx  ception (" Unable
 tp   get   the   temporary   AK :" +  e );
        }
    }
}
 public   class   SelectObje  ctSample  {
```

```
    private   static   String   endpoint  = "< oss   endpoint >";
private   static   String   bucketName  = "< bucket >”;
private   static   String   key  = "< object >";
private   static   String   roleArn  = "< service   role ’ s   ARN
>";//  You   can   obtain   the   ARN   of   a   RAM   use   on   the
  RAM   console .  The   RAM   user   must   have   the   permission
  to   access   OSS .
private   static   String   recordDeli  miter  = "\ n ";
private   static   int   threadCoun  t  =  10 ;

    public   static   void   main ( String []  args )  throws
Exception  {
        ClientBuil  derConfigu  ration   config  =  new   ClientBuil
derConfigu  ration ();
        RoleCreden  tialProvid  er   provider  =  new   RoleCreden
tialProvid  er ( roleArn );
        Credential  s   credential  s  =  provider . getCredent  ials
();
       // OSS   client  =  new   OSSClientB  uilder (). build (
endpoint ,  accessKeyI  d ,  accessKeyS  ecret ,  config );
        System . out . println (" Id  " +  credential  s .
getAccessK  eyId ());
        System . out . println (" Key  " +  credential  s .
getSecretA  ccessKey ());
        System . out . println (" Token  " +  credential  s .
getSecurit  yToken ());
        OSS   client  =  new   OSSClientB  uilder (). build (
endpoint ,  credential  s . getAccessK  eyId (),  credential  s .
getSecretA  ccessKey (),  credential  s . getSecurit  yToken (),
config );
        int   totalSplit  s  =  1 ;
        try  {
            SelectObje  ctMetadata   selectObje  ctMetadata  =
client . createSele  ctObjectMe  tadata (
                    new   CreateSele  ctObjectMe  tadataRequ  est (
bucketName ,  key )
                            . withInputS  erializati  on (
                                    new   InputSeria  lization ().
withCsvInp  utFormat (
                                            new   CSVFormat ().
withHeader  Info ( CSVFormat . Header . None ). withRecord  Delimiter
( recordDeli  miter ))));

            totalSplit  s  =  selectObje  ctMetadata . getCsvObje
ctMetadata (). getSplits ();
            System . out . println ( selectObje  ctMetadata .
getCsvObje  ctMetadata (). getTotalLi  nes ());
            System . out . println ( totalSplit  s );

       }
        catch  ( Exception   e )
       {
  e . printStack  Trace ();
 }

        String   sql  = " select   count (*)  from   ossobject ";

        ExecutorSe  rvice   executor  =  Executors . newFixedTh
readPool ( threadCoun  t );
        long   startTime  =  System . currentTim  eMillis ();
        List < Future < Integer >>  list  =  new   ArrayList < Future
< Integer >>();
        int   n  =  threadCoun  t  <  totalSplit  s  ?  threadCoun  t
:  totalSplit  s ;
```

```
        for ( int   i  =  0 ;  i  <  n ;  i ++) {
            int   start  =  i  *  totalSplit  s / n ;
            int   end  =  i  ==  n - 1  ?  totalSplit  s  -  1  : ( i
 + 1 )*  totalSplit  s  / n  -  1 ;
            System . out . println (" start :" +  start  + "  end :"
 +  end );
            Callable < Integer >  task  =  new   MulipartSe  lector (
 client ,  bucketName ,  key ,  start ,  end ,  sql );
            Future < Integer >  future  =  executor . submit ( task
 );
            list . add ( future );
       }

        long   totalLines  =  0 ;
        for ( Future < Integer >  task  :  list ){
            totalLines  +=  task . get (). longValue ();
       }
        long   endTime  =  System . currentTim  eMillis ();
        System . out . println (" total   lines :" +  totalLines );
        System . out . printf (" Total   time  % dms \ n " , (
 endTime  -  startTime ));

    }
}
```

SQL statement examples

- SQL statement examples (for CSV files)

Scenario	SQL statement
Return the first 10 rows.	select * from ossobject limit 10
Return all integers in column 1 and column 3 in which the integers in column 1 are larger than those in column 3.	select _1, _3 from ossobject where cast(_1 as int) > cast(_3 as int)
Return the number of records in which the first column starts with 'John'	select count(*) from ossobject where _1 like 'John%'
Return all records in which the time in column 2 is later than 2018-08-09 11:30 :25 and the value in column 3 is larger than 200.	select * from ossobject where _2 > cast(' 2018-08-09 11:30:25' as timestamp) and _3 > 200
Return the average value, sum, maximum value, and minimum value of the floats in column 2.	select AVG(cast(_2 as double)), SUM(cast(_2 as double)), MAX(cast(_2 as double)), MIN(cast(_2 as double))
Return all records in which the strings concatenated by the values in column 1 and column 3 starts with 'Tom' and ends with 'Anderson'.	select * from ossobject where (_1 || _3) like 'Tom%Anderson'

Scenario	SQL statement
Return all records in which the values in column 1 are divisible by 3.	select * from ossobject where (_1 % 3) == 0
Return all records in which the values in column 1 is in the range from 1995 to 2012.	select * from ossobject where _1 between 1995 and 2012
Return all records in which the values in column 5 is N, M, G, or L.	select * from ossobject where _5 in ('N', 'M', 'G', 'L')
Return all records in which the product of the values in column 2 and column 3 is larger than the sum of 100 and the value in column 5.	select * from ossobject where _2 * _3 > _5 + 100

- SQL statement examples (for JSON files)

  Assuming that we have the following JSON file:

```
{
  " contacts ":[
{
  " firstName ": " John ",
  " lastName ": " Smith ",
  " isAlive ":  true ,
  " age ":  27 ,
  " address ": {
    " streetAddr  ess ": " 21   2nd   Street ",
    " city ": " New   York ",
    " state ": " NY ",
    " postalCode ": " 10021 - 3100 "
  },
  " phoneNumbe  rs ": [
    {
      " type ": " home ",
      " number ": " 212   555 - 1234 "
    },
    {
      " type ": " office ",
      " number ": " 646   555 - 4567 "
    },
    {
      " type ": " mobile ",
      " number ": " 123   456 - 7890 "
    }
  ],
  " children ": [],
  " spouse ":  null
}, …… #  Similar   nodes   are   omitted .
```

```
]}
```

The following table describes SQL statement examples for the preceding JSON file.

Scenario	SQL statement
Return all records in which the value of age is 27.	select * from ossobject.contacts[*] s where s.age = 27
Return all home phone numbers.	select s.number from ossobject.contacts[*]. phoneNumbers[*] s where s.type = “home”
Return all records in which the value of spouse is null.	select * from ossobject s where s.spouse is null
Return all records in which the value of children is null.	select * from ossobject s where s.children[0] is null 说明: The preceding statement is used because an empty arry cannot be specified in other ways.

Best practice

- Multipart query for large files

  If columns in a CSV file do not include row delimiters, it is the most simple way to divide the file into parts based on bytes because you do not need to create metadata for the file. If columns in a CSV file includes row delimiters or a JSON file is queried, follow these step:

  1. Call `CreateSelectObjectMeta` to get the total number of splits for the file. If you need to call `SelectObject` for the file, call it asynchronously before the query, which reduces the scan time.
  2. Select the appropriate concurrency n based on client-side resources, and divide the total number of splits by the concurrency n to get the number of splits that each query needs to contain.
  3. Perform multipart queries by adding parameters, such as `split - range = 1 - 20` , in the request body.
  4. Merge the results if required.

- Use SelectObject for objects of the normal type. We recommend that you do not use SelectObject to query objects of the multipart and appendable types because of their poor performance caused by differences in their internal structure.

- Narrow the json path in the From statement used to query a JSON file.

  For example:

  Assuming that the following JSON file is queried:

```
{  contacts :[
        {“ firstName ”:” John ”, “ lastName ”:” Smith ”, “
 phoneNumbe  rs ”:[{“ type ”:” home ”, “ number ”:” 212 - 555 - 1234
 ”}, {“ type ”:” office ”, “ number ”:” 646 - 555 - 4567 ”}, {“ type
 ”:” mobile ”, “ number ”:” 123   456 - 7890 ”}], ” address ”:{“
 streetAddr  ess ”: “ 21   2nd   Street ”, “ city ”:” New   York ”,
 “ state ”: NY , “ postalCode ”:” 10021 - 3100 ”}
         }
]}
```

  You can use the following SQL statemtn to query all streetAddress in which the postalCode starts with 10021.

```
select   s . address . streetAddr  ess   from   ossobject . contacts
[*]  s   where   s . address . postalCode   like  ‘ 10021 %’
```

  You can also use the following SQL statement:

```
select   s . streetAddr  ess   from   ossobject . contacts [*].
address   s   where   s . postalCode   like  ‘ 10021 %’
```

  The performance of the second statement is better because the json path in the statement is more accurate.

- Process high-accuracy float points in JSON files.

  When you need to calculate high-accuracy float points in a JSON file, we recommend you set the value of ParseJsonNumberAsString to true, and use the CAST function to convert the type of float points to Decimal. For example, you can use the following statement to query for an attribute a whose value is 123456789.123456789: `select   s . a   from   ossobject   s   where cast ( s . a   as   decimal ) >  123456789 . 12345 .`

11.8 Object tagging

You can use the object tagging function to classify OSS objects. You can set lifecycle rules for objects with the same tag in batches.

Note:

The object tagging function is in the beta testing phase. You can *open a ticket* to apply for the trial of this function.

The object tagging function uses a key-value pair to tag an object. You can add a tag to an object when uploading the object or add a tag to an existing object.

- You can add up to 10 tags with different keys to an object.
- The maximum length of a key and a value is 128 bytes and 256 bytes individually.
- The key and value are both case-sensitive.
- A tag can contain letters, numbers, spaces, and the following symbols: + - = . _ : /
- Only the owner of a bucket and authorized users can read and write the tags of an object in the bucket. The read and write permissions on the tags of an object are not restricted by the ACL for the object.
- When you copy an object to another region, the tags of the object is also copied to the region.

Application scenarios

The object tagging function is not limited by folders. You can select all objects with the same tag in a bucket and set lifecycle rules for the objects in batches. For example, you can add tags to objects that are generated periodically and do not need to be stored for a long period, and then set a lifecycle rule for the objects with the tags to delete them periodically.

Usage

- APIs that may be used by the object tagging function are listed as follows:
  - *PutObjectTagging*: Adds a tag to an object. If the target object already has a tag, the original tag is overwritten by the new tag.
  - *GetObjectTagging*: Reads the tags of an object.
  - *DeleteObjectTagging*: Deletes the tags of an object.
  - *PutObject*: When using `PutObject` to upload an object, you can specify the `x - oss - tagging` request header to add tags to the object.
  - *InitiateMultipartUpload*: When initiating a multipart upload task, you can specify the `x - oss - tagging` request header to add tags to the object.
  - *CopyObject*: When copying an object, you can specify the `x - oss - tagging - directive` request header to determine whether to copy the tags of the

original object and specify the `x-oss-tagging` request header to specify the tags of the target object.

- *GetObject*: If the user has the permission to read the tags of the target object, the `x-oss-tagging-count` header is included in the response to the `GetObject` request to indicate the number of tags added to the target object.
- *HeadObject*: If the user has the permission to read the tags of the target object, the `x-oss-tagging-count` header is included in the response to the `HeadObject` request to indicate the number of tags added to the target object.

· Required permission

The permissions required by APIs related to the object tagging function are listed as follows:

- GetObjectTagging: Requires the permission to obtain the tags of the object. With this permission, you can view the tags added to an object.
- PutObjectTagging: Requires the permission to configure the tags of the object. With this permission, you can configure tags for an object.
- DeleteObjectTagging: Requires the permission to delete the tags of the object. With this permission, you can delete the tags of an object.

Object tagging and lifecycle rule management

When setting a lifecycle rule, you can specify the condition of the rule so that it applies to objects with a specified prefix or a specified tag. You can also set the prefix and tag as the condition of a rule at the same time.

· If you set a specified tag as the condition of a lifecycle rule, the rule applies to an object only when the key and value in the tag of the object both match the specified tag.

· If you specify a prefix and multiple tags as the condition of a lifecycle rule, the rule applies to an object only when the prefix and tags of the object match the prefix and all tags specified in the rule.

Example:

```
< LifecycleC  onfigurati  on >
< Rule >
< ID > r1 </ ID >
< Prefix > rule1 </ Prefix >
< Tag >< Key > xx </ Key >< Value > 1 </ Value ></ Tag >
< Tag >< Key > yy </ Key >< Value > 2 </ Value ></ Tag >
< Status > Enabled </ Status >
```

```
< Expiration >
< Days > 30 </ Days >
</ Expiration >
</ Rule >
< Rule >
< ID > r2 </ ID >
< Prefix > rule2 </ Prefix >
< Tag >< Key > xx </ Key >< Value > 1 </ Value ></ Tag >
< Status > Enabled </ Status >
< Transition >
< Days > 60 </ Days >
< StorageCla  ss > Archive </ StorageCla  ss >
</ Transition >
</ Rule >
</ LifecycleC  onfigurati  on >
```

If you set the preceding lifecycle rule:

- Objects with the rule1 prefix and "xx=1" and "yy=2" tags are deleted after 30 days.
- The storage class of objects with the rule2 prefix and "xx=1" tag is converted to Archive after 60 days.

Note:

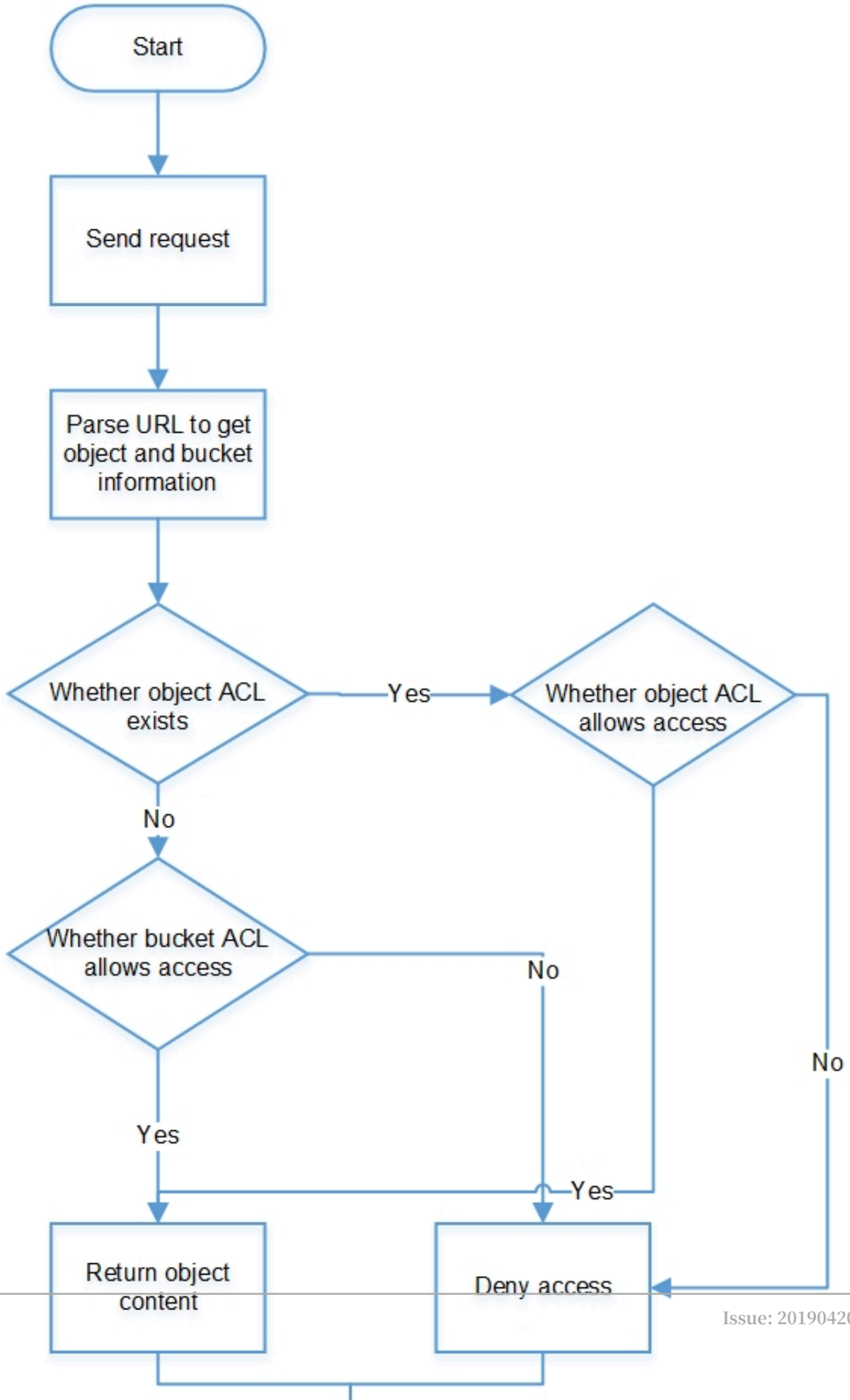
For more information, see *Manage lifecycle rules*.

12 Signature

12.1 OSS request process

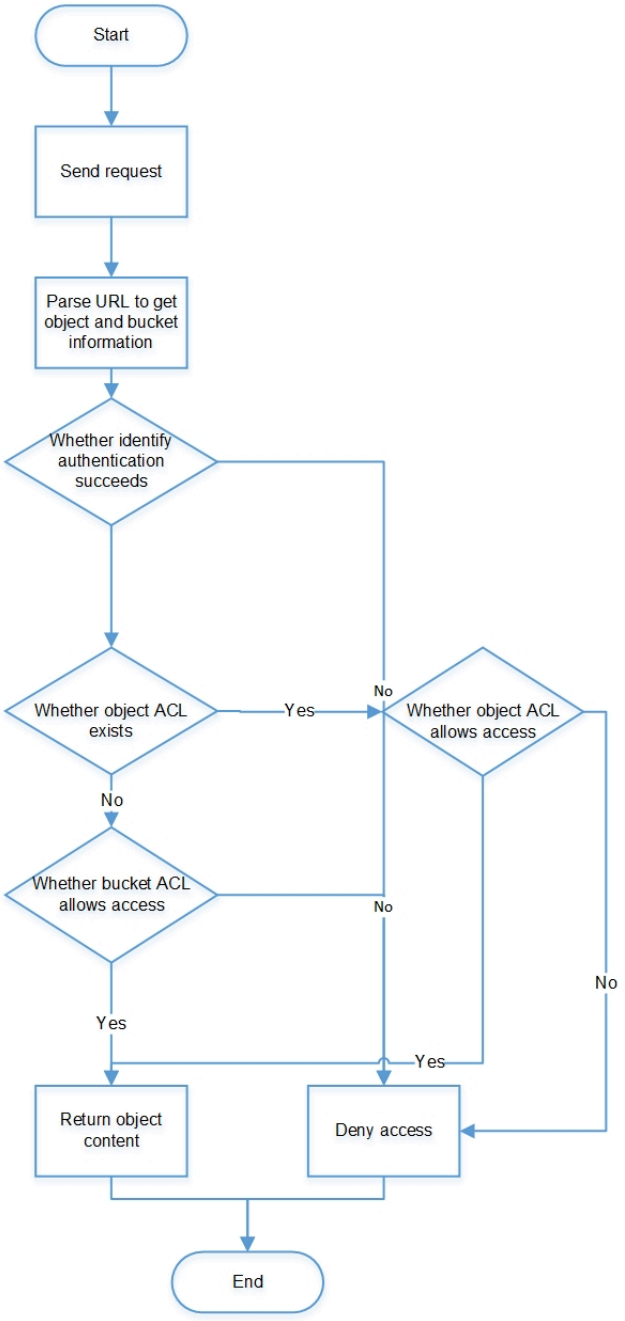
Based on whether the authentication information is included, HTTP requests sent to OSS are divided into two types: requests with authentication information and anonymous requests without authentication information. An anonymous request does not include authentication information. In contrast, a request with authentication information includes signature information in the request header or request URL, which is in compliance with the OSS API documents.

Access to OSS using anonymous requests



1. A user request is sent to the HTTP server of OSS.
2. OSS parses the URL to obtain the target bucket and object.
3. OSS checks whether an ACL is set for the object.
   - If no ACL is set for the object, the process proceeds to step 4.
   - If an ACL is set for the object, OSS checks whether the ACL allows anonymous access.
     - If the ACL allows anonymous access, the process proceeds to step 5.
     - If the ACL does not allow anonymous access, the request is rejected and the process ends.
4. OSS checks whether the bucket ACL allows anonymous access.
   - If the ACL allows anonymous access, the process proceeds to step 5.
   - If the ACL does not allow anonymous access, the request is rejected and the process ends.
5. The request passes the authentication, and the object content is returned to the user.

Access to OSS using requests with authentication information



1. A user request is sent to the HTTP server of OSS.
2. OSS parses the URL to obtain the target bucket and object.

3. OSS obtains the identity information about the requester for authentication based on the AccessKeyId of the request.
   - If the identity information is not obtained, the request is rejected and the process ends.
   - If the identity information is obtained, but the requester is not allowed to access the resource, the request is rejected and the process ends.
   - If the identity information is obtained, but the signature calculated based on the HTTP parameters in the request does not match the signature contained in the request, the request is rejected and the process ends.
   - If the authentication succeeds, the process proceeds to step 4.
4. OSS checks whether an ACL is set for the object.
   - If no ACL is set for the object, the process proceeds to step 5.
   - If an ACL is set for the object, OSS checks whether the object ACL allows access by the user.
     - If the ACL allows access by the user, the process proceeds to step 6.
     - If the ACL does not allow the access, the request is rejected and the process ends.
5. OSS checks whether the bucket ACL allows access by the user.
   - If the ACL allows access by the user, the process proceeds to step 6.
   - If the ACL does not allow access by the user, the request is rejected and the process ends.
6. The request passes the authentication, and the object content is returned to the user.

AccessKey types

Currently, the following three types of *AccessKeys* (AKs) are used to access OSS:

- AK of an Alibaba Cloud account

  An AK of an Alibaba Cloud account indicates the AK of the bucket owner. The AK of an Alibaba Cloud has full access to all resources under the correspond ing account. Each Alibaba Cloud account can have a maximum of five AK pairs

(AccessKeyId and AccessKeySecret) and the AKs can be in either an active or inactive state.

You can log on to the *AccessKey console* to add or delete AK pairs.

An AK pair can be in two states: active and inactive.

- An AK in the active state can be used for authentication.
- An AK in the inactive state cannot be used for authentication.

 Notice:
For security reasons, avoid using the AK of your Alibaba Cloud account.

- AK of a RAM user

  Resource Access Management (RAM) is a resource access control service provided by Alibaba Cloud. AKs of RAM users are authorized by the corresponding Alibaba Cloud account through RAM. These AKs can be used only to access OSS resources in buckets in accordance with the rules defined in RAM. By configuring RAM policies, you can manage multiple users in a centralized manner and control the resources that can be accessed by the users. For example, you can control the permission of a user so that the user can only read a specified bucket. A RAM user is subjected to the Alibaba Cloud account under which it was created, and does not own any actual resources. That is, all resources belong to the corresponding Alibaba Cloud account.
- AK of an STS account

  Security Token Service (STS) is an Alibaba Cloud service that provides temporary access credentials. AKs of STS accounts are authorized by STS. These AKs can be used only to access OSS resources in buckets in accordance with the rules defined in STS.

Authentication implementation

Authentication is implemented in the following three methods:

- AK authentication
- RAM authentication
- STS authentication

When a user sends a request to OSS as an individual identity, authentication is performed on the user as follows:

1. The user generates a signature string based on the request in the format specified by OSS.
2. The user uses the AccessKeySecret to encrypt the signature string and generate a verification code.
3. After receiving the request, OSS locates the corresponding AccessKeySecret based on the AccessKeyId, and obtains the signature string and verification code using the same method.
   - If the calculated verification code is the same as the provided verification code, OSS determines that the request is valid.
   - If the obtained verification code is different from the provided verification code, OSS rejects the request and returns an HTTP 403 error.

Three methods of accessing OSS with authentication

- Access OSS in the console: The authentication process is invisible to users, which means users do not need to worry about authentication configurations when they access OSS in the console. For more information, see *Download an object*.
- Access OSS using SDKs: OSS provides SDKs for multiple development languages, in which the signature algorithm is implemented. Therefore, users only need to input the AK information to access OSS using SDKs. For more information, see the access control part in the SDK documents for different development languages, such as *Java SDK: Authorized access* and *Python SDK: Authorized access*.
- Access OSS using APIs: To write code to package a call to the RESTful API, you must implement a signature algorithm to calculate the signature. For more information, see *Add a signature to the header* and *Add a signature to a URL*.

12.2 Add a signature to the header

You can add an authorization header to carry signature information in an HTTP request to indicate that the message has been authorized.

SDK signature implementation

OSS SDK has implemented the signature. You do not need to worry about the signature issue when you use the OSS SDK. To learn more about the signature implementations of specific languages, see the OSS SDK code. The files for implementing OSS SDK signature are shown in the following table:

SDK	Signature implementation
Java SDK	*OSSRequestSigner.java*
Python SDK	*auth.py*
Net SDK	*OssRequestSigner.cs*
PHP SDK	*OssClient.php*
C SDK	*oss_auth.c*
JavaScript SDK	*client.js*
Go SDK	*auth.go*
Ruby SDK	*util.rb*
iOS SDK	*OSSModel.m*
Android SDK	*OSSUtils.java*

Calculation of the Authorization field

```
Authorizat  ion  = " OSS  " +  AccessKeyI  d  + ":" +  Signature
Signature  =  base64 ( hmac - sha1 ( AccessKeyS  ecret ,
           VERB  + "\ n "
          +  Content - MD5  + "\ n "
          +  Content - Type  + "\ n "
          +  Date  + "\ n "
          +  Canonicali  zedOSSHead  ers
          +  Canonicali  zedResourc  e ))
```

- The `AccessKeyS ecret` indicates the key required for a signature.
- `VERB` indicates the HTTP request method, including PUT, GET, POST, HEAD, and DELETE.
- `\ n` is a line break.
- `Content - MD5` The Content-MD5 is the MD5 value of requested content data. The message content (excluding the header) is calculated to obtain an MD5 value, which is a 128-bit number. This number is encoded with Base64 into a Content-MD5 value. The request header can be used to check the message validity, that is, whether the message content is consistent with the sent content, such as “eB5eJF1ptWaXm4bijSPyxw==”. The request header may be empty. For more information, see *RFC2616 Content-MD5*.
- `Content - Type` indicates the requested content type, such as “application/octet-stream”. It content type may be empty.

- `Date` indicates the time that the operation takes. It must be in GMT format, such as "Sun, 22 Nov 2015 08:16:38 GMT".
- The `CanonicalizedOSSHeaders` indicates an assembly of HTTP headers whose prefixes are "x-oss-".
- The `CanonicalizedResource` indicates the OSS resource that the user wants to access.

Specifically, the values of Date and CanonicalizedResource cannot be empty. If the difference between the value of Date in the request and the time of the OSS server is greater than 15 minutes, the OSS server rejects the request and returns an HTTP 403 error.

Construct CanonicalizedOSSHeaders

All the HTTP headers whose prefixes are x-oss- are called CanonicalizedOSSHeaders. The method to construct CanonicalizedResource is as follows:

1. Convert the names of all HTTP request headers whose prefixes are x-oss- into lowercase letters. For example, convert `X-OSS-Meta-Name:TaoBao` to `x-oss-meta-name:TaoBao`.
2. If the request is sent with the AccessKeyID and AccessKeySecret obtained by the STS, you must also add the obtained security-token value to the signature string in the form of `x-oss-security-token:security-token`.
3. Sort all acquired HTTP request headers in a lexicographically ascending order.
4. Delete any space on either side of a separator between the request header and content. For example, convert `x-oss-meta-name: TaoBao` to `x-oss-meta-name:TaoBao`.
5. Separate all the content and headers with the `\n` separator to form the final CanonicalizedOSSHeaders.

Note:

- CanonicalizedOSSHeaders can be empty, and the `\n` at the end can be removed.
- If only one header must be constructed, it must be `x-oss-meta-a\n`. Note the `\n` at the end.

- If multiple headers must be constructed, it must be `x-oss-meta-a:a\nx-oss-meta-b:b\nx-oss-meta-c:c\n`. Note the `\n` at the end.

Construct CanonicalizedResource

The target OSS resource specified in the request sent by the user is called a CanonicalizedResource. The method for constructing CanonicalizedResource is as follows:

1. Set CanonicalizedResource into a null character string (“”);
2. Add the OSS resource to be accessed in the following format: `/BucketName/ObjectName`. (If ObjectName does not exist, CanonicalizedResource is “/BucketName/ “. If BucketName does not exist either, CanonicalizedResource is “/ “.)
3. If the requested resource includes sub-resources (SubResource), sort all the sub-resources in a lexicographically ascending order and separate the sub-resources using the separator `&` to generate a sub-resource string. Add “?” and the sub-resource string to the end of the CanonicalizedResource string. In this case, CanonicalizedResource is like: `/BucketName/ObjectName?acl&uploadId=UploadId`

 Note:

- The sub-resources supported by OSS currently include: acl, uploads, location, cors, logging, website, referer, lifecycle, delete, append, tagging, objectMeta, uploadId, partNumber, security-token, position, img, style, styleName, replication, replicationProgress, replicationLocation, cname, bucketInfo, comp, qos, live, status, vod, startTime, endTime, symlink, x-oss-process, response-content-type, response-content-language, response-expires, response-cache-control, response-content-disposition, and response-content-encoding.
- Three types of sub-resources are available:
  - Resource identifiers, such as acl, append, uploadId, and symlink sub-resources. For more information, see *Bucket-related operations* and *Object-related operations*.
  - Specify response header fields such as `response-***`. For more information, see the `Request Parameters section` of *GetObject* .
  - Object handling methods, such as `x-oss-process` . It is used as the object handling method, such as *Image Processing*.

Rules to calculate a signature header

- A signature string must be in the UTF-8 format. Encode a signature string containing Chinese characters with UTF-8 first, and then use it with the AccessKeySecret to calculate the final signature.
- The signing method adopted is the HMAC-SHA1 method defined in *RFC 2104*, where Key is `AccessKeyS ecret` .
- Content-Type and Content-MD5 are not required in a request. If the request requires signature verification, the null value can be replaced with the line break "\n".
- Among all non-HTTP-standard headers, only the headers starting with "x-oss-" require signature strings, and other non-HTTP-standard headers are ignored by OSS. (For example, the "x-oss-magic" header in the preceding example must be added with a signature string.)
- Headers starting with "x-oss-" must comply with the following specifications before being used for signature verification:
  - The header name is changed to lower-case letters.
  - The headers are sorted in a lexicographically ascending order.
  - No space exists before and after the colon, which separates the header name and value.
  - Each header is followed by the line break "\n". If no header is used, CanonicalizedOSSHeaders is set to null.

Example signature

Assume that AccessKeyID is 44CF9590006BF252F707 and AccessKeySecret is OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV.

Request	Signature string calculation formula	Signature string
PUT /nelson HTTP/1.0 Content-MD5: eB5eJF1ptWaXm4bijSPyxw== Content-Type: text/html Date: Thu, 17 Nov 2005 18:49:58 GMT Host: oss-example.oss-cn-hangzhou.aliyuncs.com X-OSS-Meta-Author: foo@bar.com X-OSS-Magic: abracadabra	Signature = base64(hmac-sha1(AccessKeySecret,VERB + “\n” + Content-MD5 + “\n” + Content-Type + “\n” + Date + “\n” + CanonicalizedOSSHeaders + CanonicalizedResource))	“PUT\n eB5eJF1ptWaXm4bijSPyxw==\n text/html\n Thu, 17 Nov 2005 18:49:58 GMT\n x-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nels

The signature calculation method is as follows:

Python sample code:

```
import   base64
import   hmac
import   sha
h  =  hmac . new (" OtxrzxIsfp  FjA7SwPzIL  wy8Bw21TLh  quhboDYROV ",
            " PUT \ nODBGOERFM  DMzQTczRUY  3NUE3NzA5Q  zdFNUYzMDQ
xNEM =\ ntext / html \ nThu ,  17   Nov   2005   18 : 49 : 58   GMT
\ nx - oss - magic : abracadabr  a \ nx - oss - meta - author : foo @
bar . com \ n / oss - example / nelson ",  sha )
Signature  =  base64 . b64encode ( h . digest ())
print (" Signature : % s " %  Signature )
```

The signature calculation result is 26NBxoKdsyly4EDv6inkoDft/yA=. According to the formula Authorization = “OSS “ + AccessKeyID + “:” + Signature, the value of Authorization is OSS 44CF9590006BF252F707:26NBxoKdsyly4EDv6inkoDft/yA=. The value is added with the authorization header to form the message to be sent:

```
PUT  / nelson   HTTP / 1 . 0
Authorizat  ion : OSS   44CF959000  6BF252F707 : 26NBxoKdsy
ly4EDv6ink  oDft / yA =
Content - Md5 :  eB5eJF1ptW  aXm4bijSPy  xw ==
Content - Type :  text / html
Date :  Thu ,  17   Nov   2005   18 : 49 : 58   GMT
Host :  oss - example . oss - cn - hangzhou . aliyuncs . com
X - OSS - Meta - Author :  foo @ bar . com
X - OSS - Magic :  abracadabr  a
```

Detail analysis are as follows:

- If the input AccessKeyID does not exist or is inactive, the error 403 Forbidden is returned. Error code: InvalidAccessKeyId.

- If the authorization value format in the user request header is incorrect, the error 400 Bad Request is returned. Error code: InvalidArgument.
- All the requests of OSS must use the GMT time format stipulated by the HTTP 1.1 protocol. Specifically, the date format is: `date1 = 2DIGIT SP month SP 4DIGIT ; day month year ( for example , 02 Jun 1982 ).` In the aforesaid date format, "day" occupies "2 digits". Therefore, "Jun 2", "2 Jun 1982", and "2-Jun-82" are all invalid date formats.
- If Date is not input into the header or the format is incorrect during signature verification, the error 403 Forbidden is returned. Error code: AccessDenied.
- The request must be entered within 15 minutes based on the current time of the OSS server; otherwise, the error 403 Forbidden is returned. Error code: RequestTimeTooSkewed.
- If the AccessKeyID is active but OSS determines that the signature of the user request is incorrect, the error 403 Forbidden is returned, and the correct signature string for verification and encryption is returned to the user in the response message. The user can check whether or not the signature string is correct based on the response of OSS. Return example:

```
<? xml version =" 1 . 0 " ? >
< Error >
 < Code >
     SignatureD oesNotMatc h
 </ Code >
 < Message >
     The request signature we calculated does not
 match the signature you provided . Check your key
 and signing method .
 </ Message >
 < StringToSi gnBytes >
     47 45 54 0a 0a 0a 57 65 64 2c 20 31 31
   20 4d 61 79 20 32 30 31 31 20 30 37 3a
 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72
   65 61 6c 74 65 73 74 3f 61 63 6c
 </ StringToSi gnBytes >
 < RequestId >
     1E446260FF 9B10C2
 </ RequestId >
 < HostId >
     oss - cn - hangzhou . aliyuncs . com
 </ HostId >
 < SignatureP rovided >
     y5H7yzPsA / tP4 + 0tH1HHvPEw Uv8 =
 </ SignatureP rovided >
 < StringToSi gn >
     GET
 Wed , 11 May 2011 07 : 59 : 25 GMT
/ oss - example ? acl
 </ StringToSi gn >
 < OSSAccessK eyId >
```

```
      AKIAIVAKMS  MOY7VOMRWQ
 </ OSSAccessK  eyId >
</ Error >
```

Content-MD5 calculation method

```
 Content - MD5   calculatio  n
 The   message   content  " 123456789 "  is   used   as   an   example
 .  The   Content - MD5   value   of   the   string
 is   calculated   as   follows :
 The   algorithm   defined   in   related   standards   can   be
 simplified   to   the   following :
 Calculate   the   MD5 - encrypted   128 - bit   binary   array .
 Encode   the   binary   array  ( instead   of   the   32 - bit
 string   code )  with   Base64 .
 Python   is   used   as   an   example .
 The   correct   calculatio  n   code   is :
>>>  import   base64 , hashlib
>>>  hash  =  hashlib . md5 ()
>>>  hash . update (" 0123456789 ")
>>>  base64 . b64encode ( hash . digest ())
' eB5eJF1ptW  aXm4bijSPy  xw =='
 Note :
 The   correct   code   is :  hash . digest (),  used   to   calculate
   a   128 - bit   binary   array
>>>  hash . digest ()
' x \ x1e ^$] i \ xb5f \ x97 \ x9b \ x86 \ xe2 \ x8d #\ xf2 \ xc7 '
 The   common   error   is   to   base   64   the   computed   32 -
 Bit   String   encoding   directly .
 An   incorrect   example :  hash . hexdigest (),  and   a   visible
 32 - bit   string   is   calculated .
>>>  hash . hexdigest ()
' 781e5e245d  69b566979b  86e28d23f2  c7 '
 Result   of   encoding   the   incorrect   MD5   value   with
 Base64 :
>>>  base64 . b64encode ( hash . hexdigest ())
' NzgxZTVlMj  Q1ZDY5YjU2  Njk3OWI4Nm  UyOGQyM2Yy  Yzc ='
```

12.3 Add a signature to a URL

In addition to using an authorization header, you can add signature information to a URL. It enables you to forward a URL to the third party for an authorized access.

Sample code

Python sample code used to add a signature to a URL:

```
 import   base64
 import   hmac
 import   sha
 import   urllib
 h  =  hmac . new (" OtxrzxIsfp  FjA7SwPzIL  wy8Bw21TLh  quhboDYROV ",
             " GET \ n \ n \ n114188912  0 \ n / oss - example / oss -
 api . pdf ",
              sha )
```

```
urllib . quote  ( base64 . encodestri  ng ( h . digest ()). strip ())
```

OSS SDK provides the method for adding a signature into an URL. For the detailed usage, see Authorized access in the OSS SDK Reference.

To add a signature to the OSS SDK URL, see the following table.

SDK	URL signature method	Implementation file
Java SDK	OSSClient.generatePresignedUrl	*OSSClient.java*
Python SDK	Bucket.sign_url	*api.py*
Net SDK	OssClient.GeneratePresignedUri	*OssClient.cs*
PHP SDK	OssClient.signUrl	*OssClient.php*
JavaScript SDK	signatureUrl	*object.js*
C SDK	oss_gen_signed_url	*oss_object.c*

Implementation

URL signature example:

```
http :// oss - example . oss - cn - hangzhou . aliyuncs . com / oss
- api . pdf ? OSSAccessK  eyId = nz2pc56s93  6 ** 9l & Expires =
1141889120 & Signature = vjbyPxybdZ  aNmGa % 2ByT272YEA  iv4 % 3D
```

The URL signature must include at least the following three parameters: `Signature`, `Expires`, and `OSSAccessK  eyId`.

- The `Expires` parameter indicates the time-out period of a URL. The value of this parameter is UNIX time (which is the number of seconds that have elapsed since 00:00:00 UTC, January 1, 1970. For more information, see *Wikipedia*). If the time when OSS receives the URL request is later than the value of the Expires parameter and is included in the signature, an error code request timed-out is returned.
  For example, if the current time is 1141889060, to create a URL that is scheduled to expire in 60 seconds, you can set the value of Expires to 1141889120.The valid period of a URL is 3,600 seconds by default and 64,800 seconds in maximum.
- `OSSAccessK  eyId` refers to the AccessKeyID in the key.

- `Signature` indicates the signature information. For all requests and header parameters that OSS supports, the algorithm for adding a signature to a URL is basically the same as that of *Adding a signature to a header*.

```
Signature  =  urlencode ( base64 ( hmac - sha1 ( AccessKeyS  ecret
,
          VERB  + "\ n "
         +  CONTENT - MD5  + "\ n "
         +  CONTENT - TYPE  + "\ n "
         +  EXPIRES  + "\ n "
         +  Canonicali  zedOSSHead  ers
         +  Canonicali  zedResourc  e )))
```

  The difference is listed as follows:

  - When a signature is added to a URL, the Expires parameter replaces the Date parameter.
  - Signatures cannot be included in a URL and the Header at the same time.
  - If more than one incoming Signature, Expires, or AccessKeyId value is available , the first of each incoming value is used.
  - Whether the request time is later than the Expires time, is verified first before verifying the signature.
  - When you put the signature string into a URL, remember to perform the UrlEncode for a URL.
- When you add a signature to a temporary user URL, the `security - token` must also be entered. The format is as follows:

```
http :// oss - example . oss - cn - hangzhou . aliyuncs . com / oss
- api . pdf ? OSSAccessK  eyId = nz2pc56s93  6 ** 9l & Expires =
1141889120 & Signature = vjbyPxybdZ  aNmGa % 2ByT272YEA  iv4 % 3D &
security - token = SecurityTo  ken
```

Detail analysis

- If you adopt the approach of adding a signature to a URL, the authorized data is exposed on the Internet before the authorization period expires. We recommend that you must assess the usage risks in advance.
- The PUT and GET requests both support adding a signature in a URL.
- When a signature is added to a URL, the sequence of Signature, Expires, and AccessKeyId can be swapped. If one or more Signature, Expires, or AccessKeyId parameter is missing, the error 403 Forbidden is returned. Error code: AccessDeni ed.

- If the current access time is later than the Expires time set in the request, the error 403 Forbidden is returned. Error code: AccessDenied.
- If the format of the Expires time is incorrect, the error 403 Forbidden is returned. Error code: AccessDenied.
- If the URL includes one or more Signature, Expires, or AccessKeyId parameter and the header also includes signature information, the error 400 Bad Request is returned. Error code: InvalidArgument.
- When the signature string is generated, the Date parameter is replaced by the Expires parameter, but the headers such as content-type and content-md5 defined in the preceding section are still included. (Though the Date request header still exists in the request, you can skip adding it to the signature string.)

13 Identity authentication

13.1 What is RAM and STS

RAM and STS are permission management systems provided by Alibaba Cloud.

RAM is primarily used to control account system permissions. RAM enables users to create subaccounts within the range of primary account permissions. Different subaccounts can be allocated different permissions for authorization management.

STS is a security credential (token) management system that grants temporary access permissions. STS allows users to grant access rights to the temporary accounts.

Why RAM and STS?

RAM and STS are designed to resolve the core issue such as how to securely grant access permissions to other users without disclosing the primary account ’ s AccessKey. Disclosure of AccessKey poses a serious security threat because unauthorized users may operate account resources and the risk of data leakage or stealing of important information is high.

RAM provides a long-term permission control mechanism. Various subaccounts assign different permissions to the different users. This way, even the disclosure of subaccount information would not cause a global information leakage. However, subaccounts have long-term validity.

 Note:
Therefore, AccessKey of subaccounts must not be disclosed.

On the contrary, STS provides temporary access authorization by returning a temporary AccessKey and the token. This information can be provided directly to the temporary accounts, allowing them access to OSS. Generally, the permissions obtained from STS are more restrictive and only valid for a limited period of time. Thus, the disclosure of this information has little effect on the system.

These functions are further illustrated with the help of examples.

Basic concepts

The following are some explanations of the basic concepts:

- Subaccount: A subaccount is created from the Alibaba Cloud primary accounts . Once created, it is assigned an independent password and permissions. Each subaccount has its own AccessKey and can perform authorized operations similar to the primary account. Generally, subaccounts can be understood as users with certain permissions or operators with permissions to perform specific operations.
- Role: Role is a virtual concept for certain operation permissions. However, it does not have independent logon passwords or AccessKeys.

   Note:
  Subaccounts can assume roles. When a role is assumed, the permissions granted for a subaccount are the permissions of the role.
- Policy: Policies are rules used to define permissions; for example, they permit users to read or write certain resources.
- Resource: Resources are the cloud resources that users can access like all OSS buckets, a certain OSS bucket, or a certain object in a specific OSS bucket.

A subaccount and roles have the same relationship to each other as you and your identities. At work, you may be an employee, while at home you may be a father. In different scenarios, you may assume different roles. Different roles are assigned corresponding permissions. The concept of “employee” or “father” is not an actual entity that can be the subject of actions. These concepts are only complete when an individual assumes them. This illustrates an important concept: a role may be assumed by multiple people at the same time.

 Note:
Once the role is assumed, this individual automatically obtains all the permissions of the role.

The following example provides better understanding of the concept:

- Assume that Alice is the the Alibaba Cloud user and she has two private OSS buckets, alice_a and alice_b. Alice has full permission for both buckets.
- To avoid leaking her Alibaba Cloud account AccessKey, which would pose a major security risk, Alice uses RAM to create two subaccounts, Bob and Carol. Bob has read/write permission for alice_a and Carol has read/write permission for alice_b . Bob and Carol both have their own AccessKeys. This way, if one is leaked, only

the corresponding bucket is affected and Alice can easily cancel the leaked user permissions on the console.

- Now, for some reason, Alice must authorize another person to read the objects in alice_a. In this situation, she must not only disclose Bob's AccessKey. Rather, she can create a new role like AliceAReader, and grant this role the read permission for alice_a. However, note that, at this time, AliceAReader cannot be used because no AccessKey corresponds to this role. AliceAReader is currently only a virtual entity with the permission to access alice_a.
- To obtain temporary authorization, Alice can call the STS's AssumeRole interface to notify STS that Bob wants to assume the AliceAReader role. If successful, STS returns a temporary AccessKeyId, AccessKeySecret, and SecurityToken, which serve as the access credentials. When these credentials are given to a temporary account, the user obtains temporary permission to access alice_a. The credentials' expiration time is specified when the AssumeRole interface is called.

Why are RAM and STS so complex?

Initially, RAM and STS concepts seem to be complex. This is because flexibility is given to permission control at the cost of simplicity.

Subaccounts and roles are separated to separate the entity that executes operations from the virtual entity that represents a permissions set. If a user requires many permissions including the read and write permissions but each operation only requires part of the total permission set, you can create two roles, one with the read permission and the other with the write permission. Then create a user who does not have any permission but can assume these two roles. When the user needs to read or write data, the user can temporarily assume the role with the read permission or the role with the write permission. This reduces the risk of permission leaks for each operation. Additionally, roles can be used to grant permissions to other Alibaba Cloud users, making the collaboration easier.

Here, flexibility does not mean you have to use all these functions. You only need to use the subset of the functions as required. For example, if you do not need to use temporary access credentials that have an expiration time, you can only use the RAM subaccount function, without STS.

In what follows, we use examples to create a RAM and STS user guide and provide instructions. For the operations in these examples, we do our best to use console and command line operations to reduce the actual amount of codes that must be used. If

you must use code to perform these operations, we recommend that you see the RAM and STS API Manual.

Test tool

During testing, we use osscmd, a tool in the OSS PythonSDK that allows you to directly work on OSS through the command line. osscmd can be obtained from *PythonSDK*.

Typical osscmd usage:

```
 Download   files
./ osscmd   get   oss :// BUCKET / OBJECT   LOCALFILE  -- host =
 Endpoint  - i   AccessKeyI  d  - k   AccessKeyS  ecret
 Here ,  replace   BUCKET   and   OBJECT   with   your   own   bucket
   and   object ,  and   the   endpoint   format   must   be   similar
   to   oss - cn - hangzhou . aliyuncs . com .  For   AccessKeyI  d
   and   AccessKeyS  ecret ,  use   the   informatio  n   correspond
 ing   to   your   own   account
 Upload   files
./ osscmd   put   LOCALFILE   oss :// BUCKET / OBJECT  -- host =
 Endpoint  - i   AccessKeyI  d  - k   AccessKeyS  ecret
 The   meaning   of   each   field   is   the   same   as   for   the
   download   example
```

13.2 RAM user

With Alibaba Cloud RAM, you can create RAM users under your Alibaba Cloud account. Each RAM user has their own AccessKeys. In this case, your Alibaba Cloud account is referred to as the primary account and the created RAM users are referred to as the sub-accounts. The AccessKey of a sub-account can be used only to perform operations authorized by your Alibaba Cloud account and use resources authorized by your Alibaba Cloud account.

Scenario

If multiple users need to use resources under your Alibaba Cloud account, they can only use the AccessKey of your Alibaba Cloud account to access the resources. If this occurs, the following two issues arise:

- Your AccessKey is exposed to multiple users, which increases the risk of mistakenly exposing its contents.
- You cannot control which user or users can access specific resources (such as buckets).

To resolve the preceding issues, you can use Alibaba Cloud RAM to create RAM users with their own AccessKeys under your Alibaba Cloud account. In this case, your

Alibaba Cloud account is referred to as the primary account and the created RAM users are referred to as the sub-accounts. You can perform only operations The AccessKey of a sub-account can be used only to perform operations authorized by your Alibaba Cloud account and use resources authorized by your Alibaba Cloud account.

Implementation

For more information about RAM and how to create a RAM user, see *Introduction*. To grant OSS access permissions to users by creating RAM policies, see *RAM Policy*.

13.3 Access OSS with a temporary access token provided by STS

You can temporarily access OSS by using Security Token Service (STS) provided by Alibaba Cloud. Alibaba Cloud STS is a Web service that provides users with temporary access tokens. Using STS, you can grant an access credential with customized permissions and valid periods to third-party applications and federated users whose IDs are managed by you.

Scenarios

Users managed by your local identity system are referred to as federated users, for example, the users of your applications, local accounts owned by your enterprises, and third-party applications. Federated users may need to access your OSS resources directly. In addition, federated users can also include the users that are created by you and have access to your applications and resources in Alibaba Cloud.

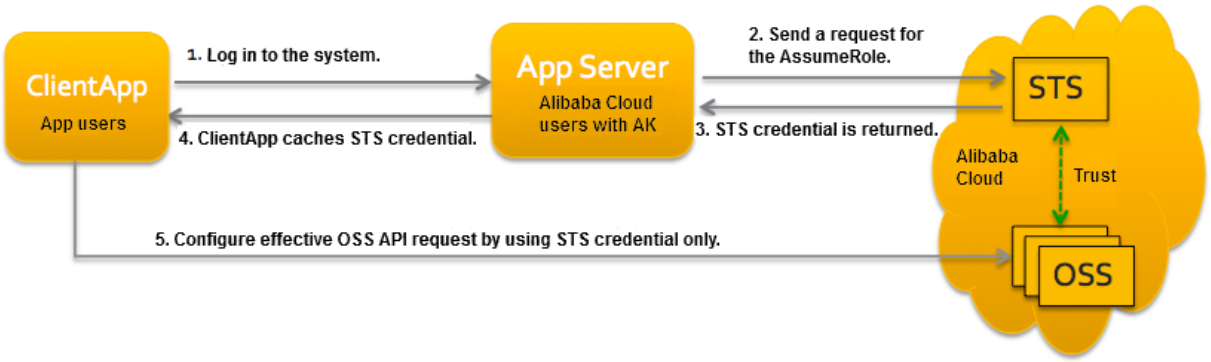
For these users, you can use STS to manage the temporary access tokens for their Alibaba Cloud accounts (or RAM users). You can create temporary access credentials for federated users to grant OSS access permissions to them without providing your long term keys (such as logon password and AccessKeys) of your Alibaba Cloud accounts or RAM users to the federated users. The permissions and valid period of the credential can be customized. You do not need to revoke the permissions of the credential because it automatically becomes invalid after it expires.

Credentials generated by STS include security tokens (SecurityToken) and temporary access keys (AccessKeyId and AccessKeySecret). You can use a temporary access key in the same way as you use the AccessKey of an Alibaba Cloud account or a RAM user to send a request. Each request sent to OSS must carry a security token.

Implementation

A mobile application is used as an example. Assume that you are a mobile application developer and try to use Alibaba Cloud OSS to store end user data for your app. You must keep the data of each application user isolated to prevent the data of an application user from being obtained by other application users. You can use STS to authorize users so that they can directly access your OSS resources.

The following figure describes the process of using STS to grant OSS access to users.



1. An application user logs on to the application server. An application user is an end user of the application and has no relationship to an Alibaba Cloud account. The application server can be logged on by an application user. The application server must define the minimum access permission for each valid application user.
2. The application server request a security token from STS. Before calling STS, the application server must determine the minimum access permission for each application user (described in policy syntax) and the expiration time of the authorization. Then, the application server uses AssumeRole to obtain a security token which indicates a role.
3. STS returns a valid access credential to the application server. The credential includes a security token, a temporary access key (AccessKeyId and AccessKeyS ecret), and the expiration time.
4. The application server returns the access credential to the application user (ClientApp). The credential can be cached by the ClientApp. When the credential becomes invalid, the ClientApp must request a new valid access credential from the application server. For example, if the valid period of the returned access credential is an hour, the ClientApp can request the application server to update the access token every 30 minutes.

5. The ClientApp use the access credential in the local cache to request Alibaba Cloud service APIs. ECS perceives the STS access credential and uses STS to verify the credential so that it can correctly respond user's requests.

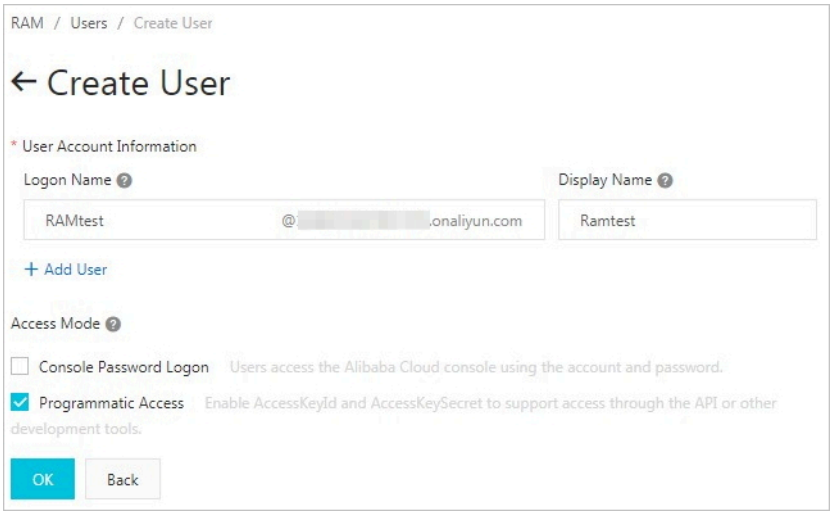
For more information about STS security tokens, role management, and role usage, see *Understand RAM roles*. You can call the *AssumeRole* interface to obtain a valid access credential.

Procedure

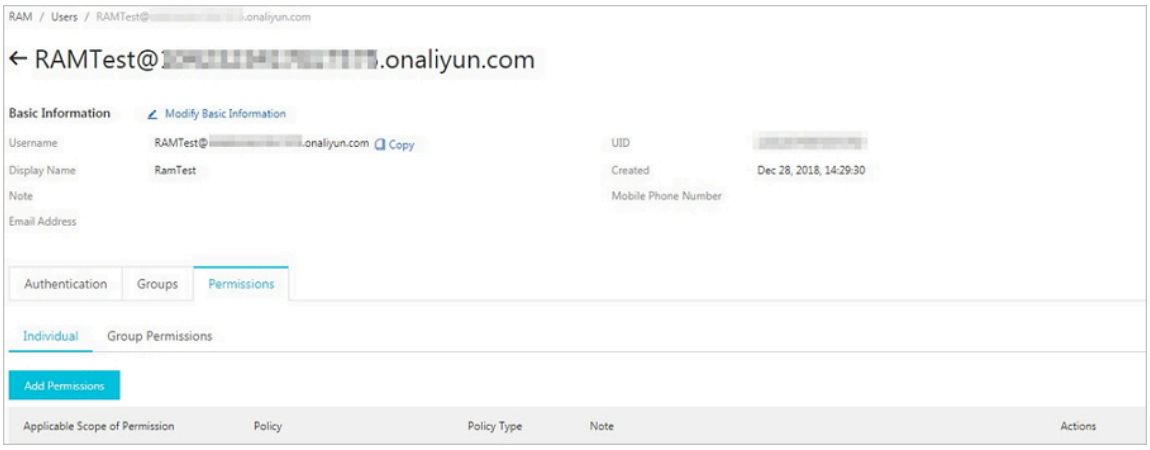
Assume that a bucket named ram-test is used to store user data and it is required that STS should be used to grant permissions to a RAM user so that the user can access OSS buckets.

You can use OSS SDK and STS SDK together to access an OSS instance with a temporary access token provided by STS.

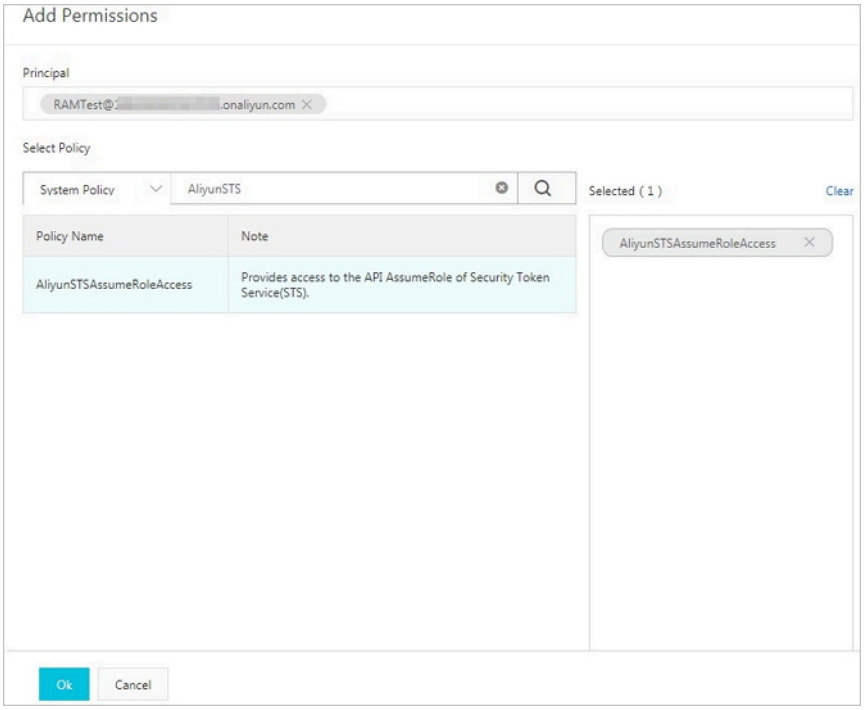
1. Create a RAM user.
   a. Log on to the *RAM console*.
   b. In the RAM page, click Users.
   c. In the Users page, click Create User.
   d. In the Create User page, enter Logon Name and Display Name in the User Account Information area, select Programmatic Access for Access Mode, and then click OK.



   e. Select Permissions > Add Permissions.



   f. In the Add Permissions page, add the AliyunSTSAssumeRoleAccess permission for the created RAM user.



Note:

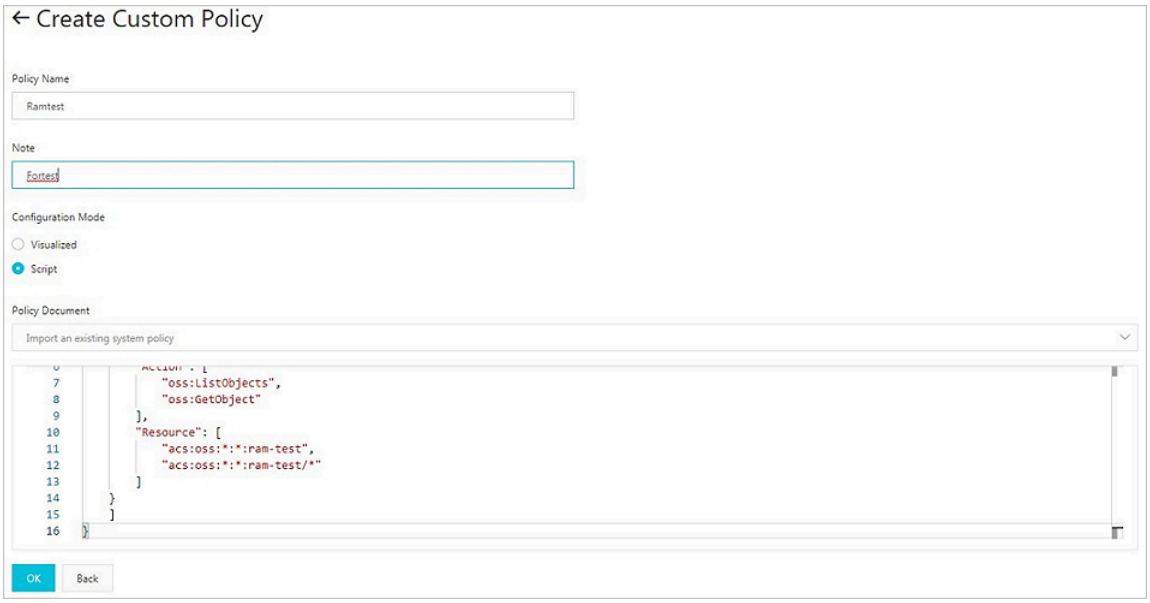
Do not grant other permissions to the RAM user because it automatically obtains all permissions of a role when it acts as the role.

2. Create a permission policy.
   a. Log on to the *RAM console*.
   b. In the RAM page, click Policies.
   c. Click Create Policy.
   d. In the Create Custom Policy page, enter the Policy Name and Note, and select Visualized or Script for Configure Mode.

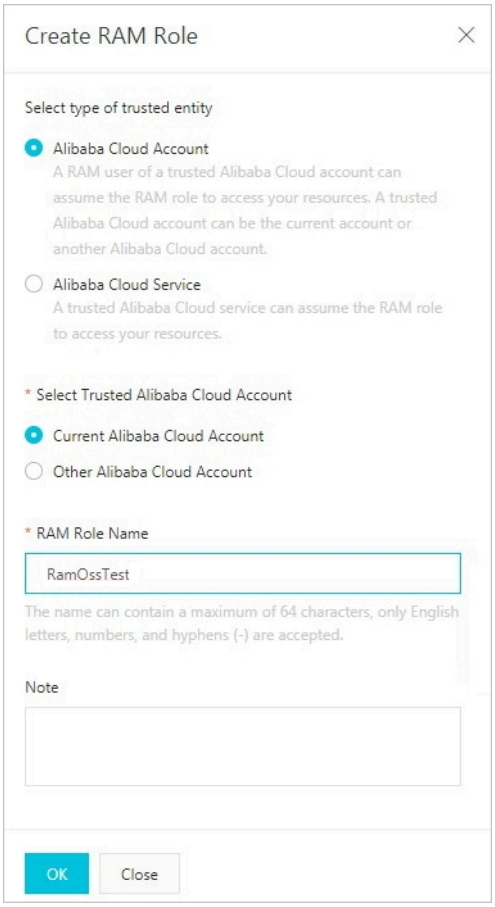
      For example, if you select Script and want to grant read only permissions, such as ListObjects and GetObject, to a RAM user named ram-test, add the following script in the Policy Document.

```
{
 " Version ": " 1 ",
 " Statement ": [
  {
     " Effect ": " Allow ",
     " Action ": [
       " oss : ListObject  s ",
       " oss : GetObject "
     ],
     " Resource ": [
       " acs : oss :*:*: ram - test ",
       " acs : oss :*:*: ram - test /*"
     ]
```

```
    }
  ]
}
```

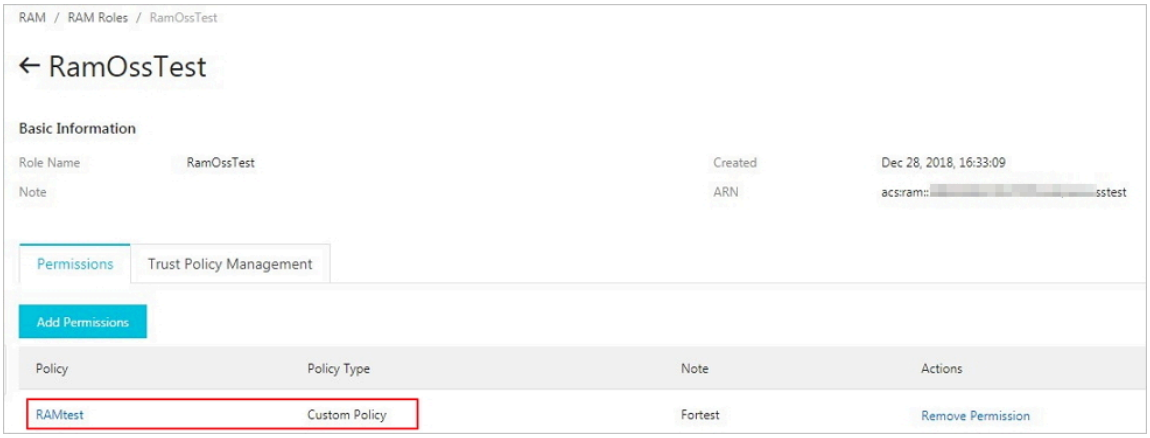

Create Custom Policy
Policy Name
Ramtest
Note
Fortest
Configuration Mode
Visualized
Script
Policy Document
Import an existing system policy
"oss:ListObjects",
"oss:GetObject"
],
"Resource": [
"acs:oss:*:*:ram-test",
"acs:oss:*:*:ram-test/*"
]
}
]
}
OK
Back

3. Create a role.
   a. Log on to the *RAM console*.
   b. In the RAM page, click RAM Roles.
   c. In the RAM Roles page, click Create RAM Role.
   d. In the Create RAM Role page, enter the RAM Role Name (RamOssTest in this example), select the type of trusted entities and keep the default selection for Select Trusted Alibaba Cloud Account.



   e. Click Add Permissions on the right of the created role RamOssTest.
   f. In the Add Permissions page, select Custom Policy and add the policy Ramtest that you created in step 2.

      After the policy is added, the page is shown as follows.



Note:

ARN indicates the ID of the role that the RAM user acts.

4. **Obtain an STS AK and security token through STS APIs**

   You can request STS through STS SDKs to obtain a security token. For more information about the installation and usage of STS SDKs, see *Installation*.

   The following code is described as an example to obtain a security token through STS Java SDK.

```
public class StsService Sample {
    public static void main ( String [] args ) {
        String endpoint = " sts . aliyuncs . com ";
        String accessKeyI d = "< access - key - id >";
        String accessKeyS ecret = "< access - key - secret >";
        String roleArn = "< role - arn >";
        String roleSessio nName = " session - name ";
        String policy = "{\ n " +
                "    \" Version \": \" 1 \", \ n " +
                "    \" Statement \": [\ n " +
                "        {\ n " +
                "            \" Action \": [\ n " +
                "                \" oss :*\"\ n " +
                "            ], \ n " +
                "            \" Resource \": [\ n " +
                "                \" acs : oss :*:*:*\" \ n " +
                "            ], \ n " +
                "            \" Effect \": \" Allow \"\ n " +
                "        }\ n " +
                "    ]\ n " +
                "}";
        try {
            // Adds an endpoint . ( The STS endpoint
is directly used . The first two parameters are
left blank , indicating that the region ID is not
required .)
            DefaultPro file . addEndpoin t ("", "", " Sts ",
endpoint );
            // Constructs a default profile . ( The
parameter is left blank , indicating that the region
ID is not required .)
```

```
            IClientPro  file   profile  =  DefaultPro  file .
 getProfile ("",  accessKeyI  d ,  accessKeyS  ecret );
           //  Uses   the   constructe  d   profile   to
 construct   a   client .
            DefaultAcs  Client   client  =  new   DefaultAcs
 Client ( profile );
            final   AssumeRole  Request   request  =  new
 AssumeRole  Request ();
            request . setMethod ( MethodType . POST );
            request . setRoleArn ( roleArn );
            request . setRoleSes  sionName ( roleSessio  nName );
            Request .  setpolicy  ( policy ); //  If   the   policy
  is   empty ,  the   user   obtains   all   permission  s   of
 the   role .
            Request .  setduratio  nseconds  ( 1000l ); //  Sets
 the   valid   period   of   a   credential .
            final   AssumeRole  Response   response  =  client .
 getAcsResp  onse ( request );
            System . out . println (" Expiration : " +  response .
 getCredent  ials (). getExpirat  ion ());
            System . out . println (" Access   Key   Id : " +
 response . getCredent  ials (). getAccessK  eyId ());
            System . out . println (" Access   Key   Secret : " +
 response . getCredent  ials (). getAccessK  eySecret ());
            System . out . println (" Security   Token : " +
 response . getCredent  ials (). getSecurit  yToken ());
            System . out . println (" RequestId : " +  response .
 getRequest  Id ());
       }  catch  ( ClientExce  ption   e ) {
            System . out . println (" Failed : ");
            System . out . println (" Error   code : " +  e .
 getErrCode ());
            System . out . println (" Error   message : " +  e .
 getErrMsg ());
            System . out . println (" RequestId : " +  e .
 getRequest  Id ());
       }
   }
}
```

The parameters are described as follows:

- AccessKeyId and AccessKey Secret: Indicates the AK information about the RAM user.
- RoleArn: Indicates the ID of the role that the user acts.
- RoleSessionName: Indicates the name used to identify a temporary credential. We recommend you use different application user names to identify different credentials.
- Policy: Indicates the permission limits added to a user when the user acts as a role.

Note:

Policies are used to control the permissions of a temporary credential after the user acts as a role. The permission of a temporary credential is the intersection

of the role permissions and the policies. Policies are passed in to adjust the permissions more flexibly. For example, you can use policies to set different limits on the path where a file is upload for different users.

- DurationSeconds: Indicates the valid period (in seconds) of a temporary credential. The value of the parameter ranges from 900 to 3,600.

5. Access OSS using the STS AK and security token.

   After obtaining the STS AK and security token, you can use the STS credential to construct a signed request.

```
// This example uses the endpoint China East 1 (
 Hangzhou ). Specify the actual endpoint based on your
   requiremen ts .
 String endpoint = " http :// oss - cn - hangzhou . aliyuncs .
 com ";
// It is highly risky to log on with the
 AccessKey of an Alibaba Cloud account because the
 account has permission s on all the APIs in OSS
 . We recommend that you log on as a RAM user
 to access APIs or perform routine operations and
 maintenanc e . To create a RAM account , log on to
 https :// ram . console . aliyun . com .
 String accessKeyI d = "< yourAccess KeyId >";
 String accessKeyS ecret = "< yourAccess KeySecret >";
 String securityTo ken = "< yourSecuri tyToken >";

// After a user obtains a temporary STS credential ,
 the OSSClient is generated with the security token
 and temporary access key ( AccessKeyI d and AccessKeyS
 ecret ) in the credential .
// Creates an OSSClient instance . Note that the STS
 credential generated in the preceding step is used .
 OSSClient ossClient = new OSSClient ( endpoint , accessKeyI
 d , accessKeyS ecret , securityTo ken );

// Performs OSS operations .

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

14 Access and control

14.1 Overview

OSS provides multiple access control methods, including ACLs, RAM policies, and bucket policies, for users who access objects stored in buckets.

- *ACL*: OSS provides Access Control List (ACL) for access control. An ACL is set based on resources. You can set ACLs for buckets or objects. You can set an ACL for a bucket when you create the bucket or for an object when you upload the object to OSS. You can also modify the ACL for a created bucket or an uploaded object at anytime.
- *RAM Policy*: Resource Access Management (RAM) is a service provided by Alibaba Cloud for resource access control. RAM policies are configured based on users. By configuring RAM policies, you can manage multiple users in a centralized manner and control the resources that can be accessed by the users. For example, you can control the permission of a user so that the user can only read a specified bucket. A RAM user belongs to the Alibaba Cloud account under which it was created, and does not own any actual resources. That is, all resources belong to the corresponding Alibaba Cloud account.
- *Bucket Policy*: Bucket policies are configured based on resources. Compared with RAM policies, bucket policies can be directly configured on the graphical console. By configuring bucket policies, you can authorize users to access your bucket even you do not have permissions for RAM operations. By configuring bucket policies, you can grant access permissions to RAM users under other Alibaba Cloud accounts, and to anonymous users who access your resources from specified IP addresses or IP ranges.

14.2 ACL

OSS provides Access Control Lists (ACLs) for permission control. ACLs are access policies that grant bucket and object access permissions to users. You can configure

an ACL for a bucket when you create it or for an object when you upload it. You can also modify the ACL for a created bucket or an uploaded object at any time.

Bucket ACL

Bucket ACLs are configured to control the accesses on buckets. You can configure any of the following three types of ACLs for a bucket: public-read-write, public-read, and private, which are described in the following table.

Permission	Description	Access restrictions
public-read-write	Public read and write	All users (including anonymous users) can read, write, and delete objects in the bucket. The fees incurred by these operations are charged to the bucket owner. Exercise caution when you configure this ACL.
public-read	Public read and private write	Only the bucket owner can write or delete the objects in the bucket. All other users (including anonymous users) can only read objects in the bucket.
private	Private read and write	Only the bucket owner can read, write, and delete objects in the bucket. All other users are prohibited from accessing objects in the bucket without authorizat ion.

You can configure and view bucket ACLs through the console, APIs and SDKs.

- Console: See *Create a bucket* and *Change bucket ACL*.
- API: See *PutBucketACL* and *GetBucketACL*.
- SDK: See *Bucket*.

Object ACL

Object ACLs are configured to control the accesses on objects. You can configure any of the following four types of ACLs for an object: private, public-read, public-read-write, and default. You can use the `x-oss-object-acl` header included in the PUT request to perform the PutObjectACL operation. Only the owner of a bucket can perform the PutObjectACL operation on objects in the bucket.

The following table describes the four types of ACLs for objects.

Permission	Description	Access restrictions
public-read-write	Public read and write	All users can read from and write to the object.
public-read	Public read and private write	Only the object owner can read from and write to the object. All other users can only read from the object.
private	Private read and write	Only the object owner can read from and write to the object. All other users are prohibited from accessing the object without authorization.
default	Default ACL	The object inherits the ACL from the bucket, that is, the ACL of an object is the same as the ACL of the bucket where the object is stored.

Note:

- If no ACL is configured for an object, the object uses the default ACL, indicating that the object has the same ACL as the bucket where the object is stored.
- If an ACL is configured for an object, the object ACL takes precedence over the ACL of the bucket where the object is stored. For example, if the ACL of an object is set to public-read, all users can read from the object regardless of the bucket ACL.

You can configure and view object ACLs through the console, APIs and SDKs.

- Console: See *Create a bucket* and *Change object ACL*.
- API: See *PutObjectACL* and *GetObjectACL*.
- SDK: See *Manage ACL for an object*.

14.3 Access control based on RAM Policy

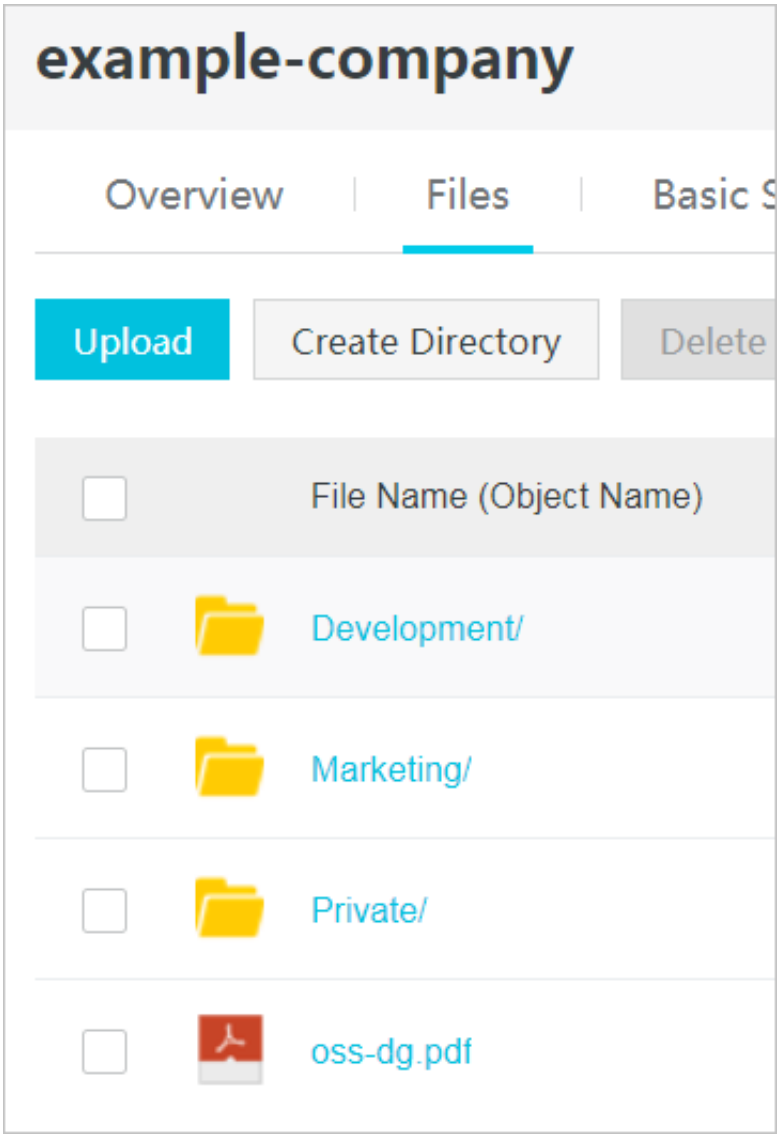
14.3.1 Tutorial: Use RAM Policy to control access to buckets and folders

This tutorial explains how to use RAM policies to grant and control user access to OSS buckets and folders.

In the example, we first create a storage space and folder, and then create access management (RAM) users using the Alibaba Cloud account, and grant these users incremental permissions to the created OSS storage space and folders by creating different RAM policies.

Buckets and folders

The data model structure of Alibaba Cloud OSS is flat instead of hierarchical. All objects (files) are directly related to their corresponding buckets. Therefore, OSS lacks the hierarchical structure of directories and subfolders as in a file system. However, you can emulate a folder hierarchy in the OSS console, where you can arrange and manage files by folders, as shown in the following figure.

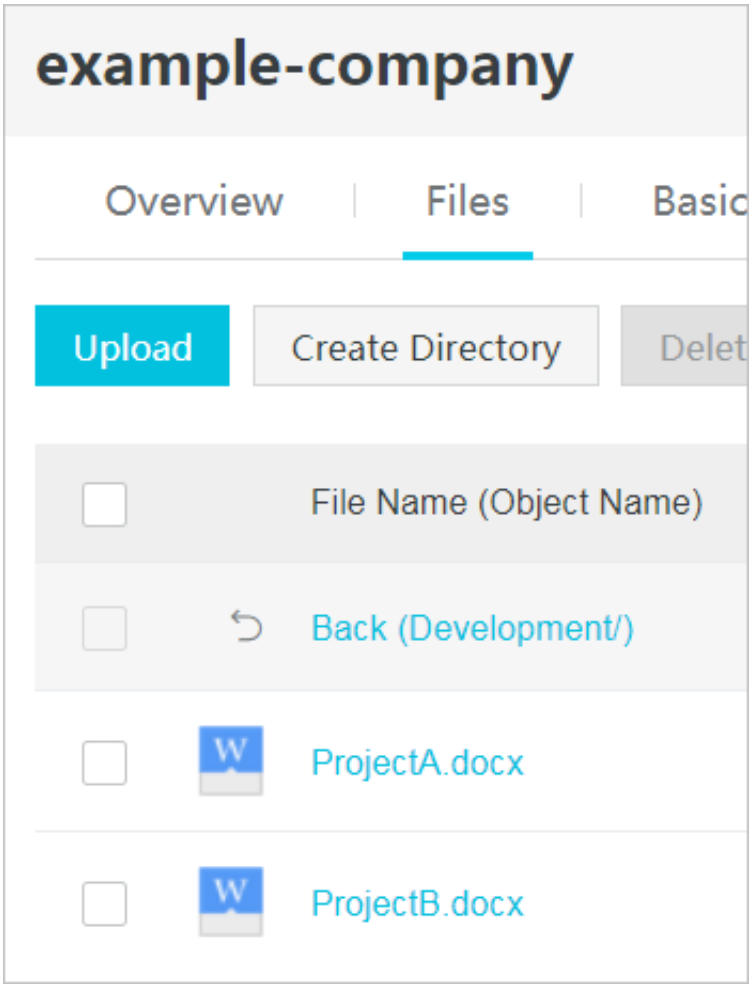


OSS is a distributed object storage service that uses a key-value pair format. Users retrieve the content of an object based on its unique key (object name). For example, the bucket named example-company has three folders: `Developmen t`, `Marketing` and `Private`. This bucket also has one object `oss - dg . pdf`.

- When you create the `Developmen t` folder, the console creates an object with the key `Developmen t /`. Note that the key of a folder includes the delimiter `/`.
- When you upload an object named `ProjectA . docx` into the `Developmen t` folder, the console uploads the object and sets its key to `Developmen t / ProjectA . docx`.

  In the key, `Developmen t` is the prefix and `/` is the delimiter. You can retrieve a list of all objects with a specific prefix and delimiter from a bucket. In the console,

you click the `Developmen t` folder, and the console lists the objects in the folder, as shown in the following figure.



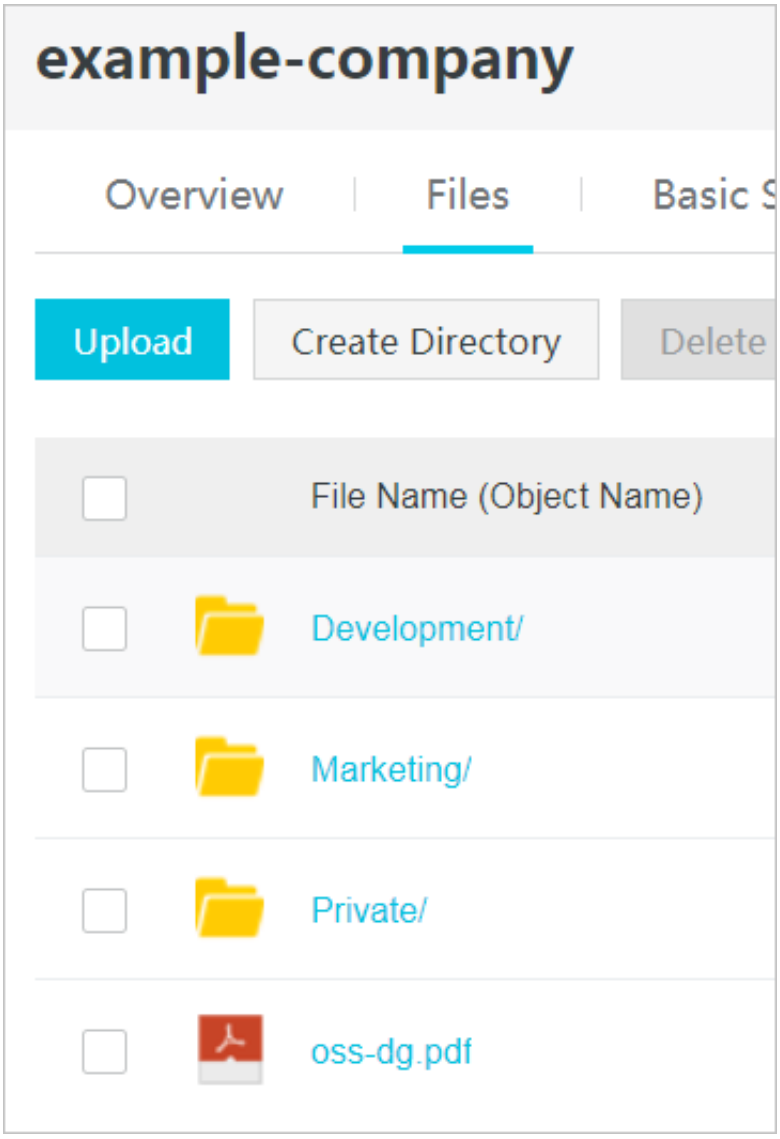
Note:
When the console lists the `Developmen t` folder in the example-company bucket, it sends OSS a request which specifies the prefix `Developmen t` and the delimiter `/`. The console responds with a folder list the same as that in the file system. The preceding example shows that the bucket example-company has two objects with the key `Developmen t / Alibaba Cloud . pdf` , `Developmen t / ProjectA . docx` , and `Developmen t / ProjectB . docx` .

The console uses object keys to resemble a logical hierarchy. When you create a logical hierarchy of objects, you can manage access to individual folders, as described later in this tutorial.

Before going into the tutorial, you also need to know the concept: root-level bucket content. Suppose the example-company bucket has the following objects:

- Development/Alibaba Cloud.pdf
- Development/ProjectA.docx
- Development/ProjectB.docx
- Marketing/data2016.xlsx
- Marketing/data2016.xlsx
- Private/2017/images.zip
- Private/2017/promote.pptx
- oss-dg.pdf

These object keys resemble a logical hierarchy with `Developmen t ,` `Marketing` , and `Private` as root-level folders and oss-dg.pdf as a root-level object. When you click the bucket name in the OSS console, the console shows the first-level prefixes and a delimiter (`Developmen t /,` `Marketing /`, and `Private /`) as root-level folders. The object key `oss - dg . pdf` does not have a prefix, so it appears as a root-level item.



Request and response of OSS

Before granting permissions, we need to understand what request the console sends to OSS when a user clicks a bucket name, the response OSS returns, and how the console interprets the response.

When a user clicks a bucket name, the console sends the *GetBucket* request to OSS. This request includes the following parameters:

- `prefix` with an empty string as its value.
- `delimiter` with `/` as its value.

An example request is as follows:

```
GET  /?  prefix =& delimiter =/  HTTP / 1 . 1
Host :  example - company . oss - cn - hangzhou . aliyuncs . com
Date :  Fri ,  24   Feb   2012   08 : 43 : 27   GMT
```

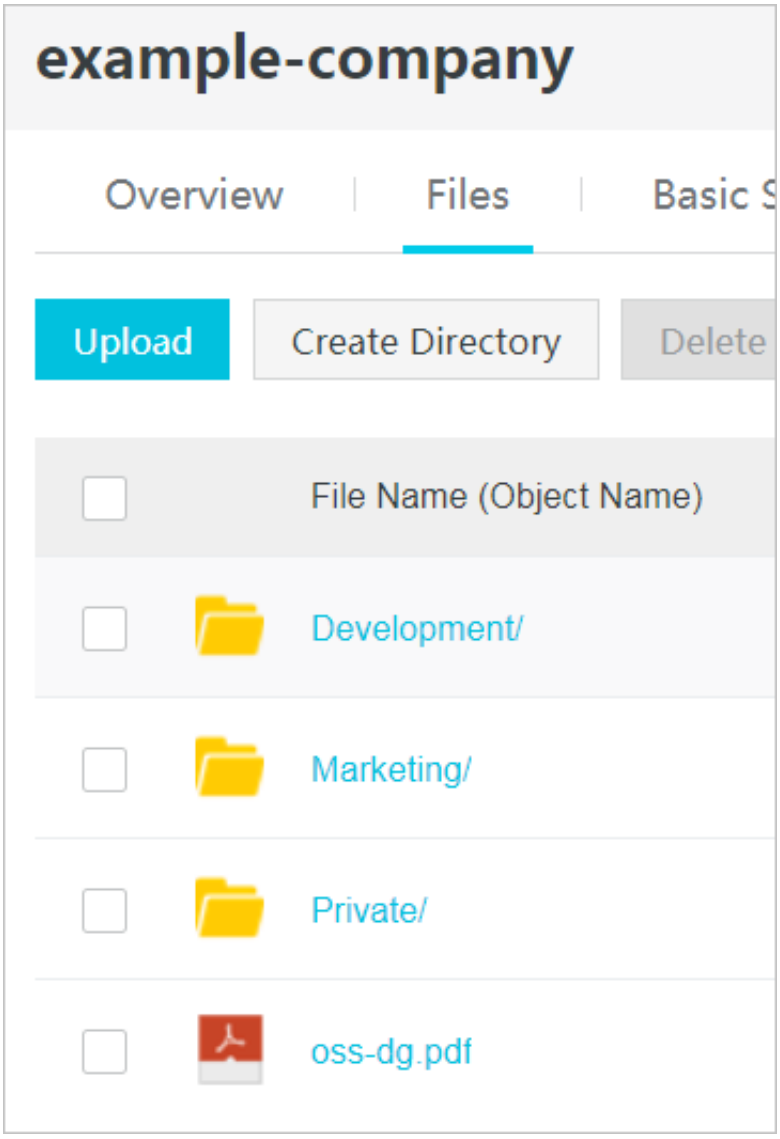
```
Authorizat ion : OSS qn6qrrqxo2 oawuk53otf jbyc : DNrnx7xHk3
sgysx7I8U9 I9IY1vY =
```

OSS returns a response that includes the `ListBucket Result` element:

```
HTTP / 1 . 1 200 OK
x - oss - request - id : 534B371674 E88A4D8906 008B
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Content - Type : applicatio n / xml
Content - Length : 712
Connection : keep - alive
Server : AliyunOSS
<? xml version =" 1 . 0 " encoding =" UTF - 8 "? >
< ListBucket Result xmlns =¡± http :// doc . oss - cn - hangzhou .
aliyuncs . com ¡±>
< Name > example - company </ Name >
< Prefix ></ Prefix >
< Marker ></ Marker >
< MaxKeys > 100 </ MaxKeys >
< Delimiter >/</ Delimiter >
    < IsTruncate d > false </ IsTruncate d >
    < Contents >
        < Key > oss - dg . pdf </ Key >
        ...
    </ Contents >
   < CommonPref ixes >
        < Prefix > Developmen t </ Prefix >
   </ CommonPref ixes >
      < CommonPref ixes >
        < Prefix > Marketing </ Prefix >
   </ CommonPref ixes >
      < CommonPref ixes >
        < Prefix > Private </ Prefix >
   </ CommonPref ixes >
</ ListBucket Result >
```

The key *oss - dg . pdf* does not contain the `/` delimiter, so OSS returns the key in the `< Contents />` element. All other keys in the bucket example-company contain the `/` delimiter, so OSS groups these keys and returns a `CommonPref ixes /` element for each of the prefix values *Developmen t /*, *Marketing /*, and *Private /*. The CommonPrefixes/ element includes a substring of the key name, which starts from the beginning of the key name and ends with (but does not include) the first occurrence of the specified `/` delimiter.

The console interprets this result and displays the root-level items as follows:



Now, if a user clicks the `Developmen t` folder, the console sends the *GetBucket* request to OSS. This request includes the following parameters:

- `prefix` with `Developmen t /` as its value
- `delimiter` with `/` as its value.

An example request is as follows:

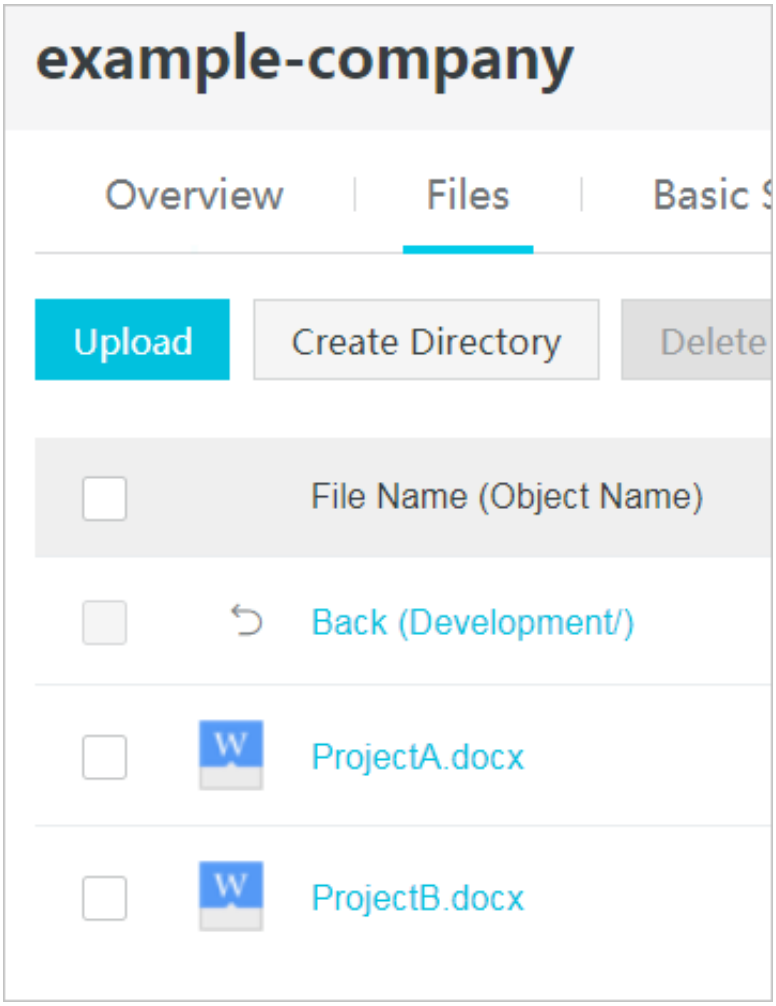
```
GET /? prefix = Developmen t /& delimiter =/ HTTP / 1 . 1
Host : oss - example . oss - cn - hangzhou . aliyuncs . com
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Authorizat ion : OSS qn6qrrqxo2 oawuk53otf jbyc : DNrnx7xHk3
sgysx7I8U9 I9IY1vY =
```

In response, OSS returns the object keys that begin with the specified prefix:

```
HTTP / 1 . 1 200 OK
x - oss - request - id : 534B371674 E88A4D8906 008B
Date : Fri , 24 Feb 2012 08 : 43 : 27 GMT
Content - Type : applicatio n / xml
```

```
 Content - Length :  712
 Connection :  keep - alive
 Server :  AliyunOSS
<?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
< ListBucket  Result   xmlns =¡± http :// doc . oss - cn - hangzhou .
 aliyuncs . com ¡±>
< Name > example - company </ Name >
< Prefix > Developmen  t /</ Prefix >
< Marker ></ Marker >
< MaxKeys > 100 </ MaxKeys >
< Delimiter >/</ Delimiter >
    < IsTruncate  d > false </ IsTruncate  d >
    < Contents >
        < Key > ProjectA . docx </ Key >
        ...
    </ Contents >
    < Contents >
        < Key > ProjectB . docx </ Key >
        ...
    </ Contents >
</ ListBucket  Result >
```

The console interprets this result and displays the object keys as follows:



Tutorial example

The tutorial example is as follows:

- You create a bucket example-company and then add three folders (`Developmen t` , `Marketing` , and `Private`) into it.
- You have two users, Anne and Leo. You want Anne to access only the `Developmen t` folder and Leo to access only the `Marketing` folder, and you want to keep the `Private` folder private. In the tutorial example, you manage access by creating Alibaba Cloud identity and Resource Access Management (RAM) users (Anne and Leo) and granting them the necessary permissions.
- RAM also supports the creation of user groups and granting of group-level permissions that apply to all users in the group. This helps you better manage and control permissions. For this example, both Anne and Leo need some common permissions. You also create a group named Staff and then add both Anne and Leo to the Staff group. You first grant permissions by attaching a group policy to the Staff group. Then you add user-specific permissions by attaching policies to specific users.

 Note:
The tutorial uses example-company as the bucket name, Anne and Leo as the RAM users, and Staff as the group name. Because Alibaba Cloud OSS requires that bucket names be globally unique, you must replace the bucket name with your own unique bucket name.

Prepare for the tutorial

In this example, you use your Alibaba Cloud account to create RAM users. Initially , these users have no permissions. You incrementally grant these users permission s to perform specific OSS operations. To test these permissions, you log on to the console with each user’ s credentials. As you incrementally grant permissions as an Alibaba Cloud account owner and test permissions as a RAM user, you have to log on and log off, each time using different credentials. You can perform this testing with one browser, but the process is more efficient if you can use two different browsers : one browser to connect to the Alibaba Cloud console with your primary account credentials and the other to connect with the RAM user credentials.

To log on to the Alibaba Cloud console with your Alibaba Cloud account credentials , go to https://account.alibabacloud.com/login/login.htm. RAM users cannot log on by using the same link. They must use the RAM user logon link. As the owner of the Alibaba Cloud account, you can provide this logon link to your users.

Note:

For more information about RAM, see *Log on with a RAM user account*.

Provide a logon link for RAM users

1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
2. In the left-side navigation pane, click Dashboard.
3. Find the URL after RAM User Logon Link: You will provide this URL to RAM users to log on to the console with their RAM user name and password.

Step 1. Create a bucket

In this step, you log on to the OSS console with your primary account credentials, create a bucket, add folders (`Developmen t` , `Marketing` , `Private`) to the bucket, and upload one or two sample documents in each folder.

1. Log on to the *OSS console*.
2. Create a bucket named example-company.

   For detailed procedures, see *Create a bucket* in the OSS Console User Guide.
3. Upload a file to the bucket.

   This example assumes that you upload the file `oss - dg . pdf` at the root level of the bucket. You can upload your own file with a different file name.

   For detailed procedures, see *Upload files* in the OSS Console User Guide.
4. Create three folders named `Developmen t` , `Marketing` , and `Private` .

   For detailed procedures, see *Create a folder* in the OSS Console User Guide.

5. Upload one or two files to each folder.

   This example assumes that you upload objects in the bucket with the following object keys:

   - Development/Alibaba Cloud.pdf
   - Development/ProjectA.docx
   - Development/ProjectB.docx
   - Marketing/data2016.xlsx
   - Marketing/data2016.xlsx
   - Private/2017/images.zip
   - Private/2017/promote.pptx
   - oss-dg.pdf

Step 2. Create RAM users and a group

In this step, you use the RAM console to add two RAM users, Anne and Leo, to your Alibaba Cloud account. You also create a group named Staff, and then add both users to the group.

 Note:
In this step, do not attach any policies that grant permissions to these users. In the following steps, you will incrementally grant permissions.

For detailed procedures on creating a RAM user, see *Create a RAM user* in the RAM Quick Start. Remember to create a logon password for each RAM user.

For detailed procedures on creating a group, see the Create a group section of *Groups* in the RAM User Guide.

Step 3: Verify that RAM users have no permissions

If you use two browsers, now you can use the other one to log on to the console by using one of the RAM user credentials.

1. Open the RAM user logon page, and log on to the RAM console with Anne's or Leo's credentials.
2. Open the OSS console.

   You find no buckets in the console, which means that Anne does not have any permissions on the bucket example-company.

Step 4: Grant group-level permissions

We want both Anne and Leo to have the access and ability to perform the following tasks:

- List all buckets owned by the Alibaba Cloud account.

  To do this, Anne and Leo must have permission for the `oss : ListBucket s` action.
- List root-level items, folders, and objects, in the example-company bucket.

  To do this, Anne and Leo must have permission for the `oss : ListObject s` action on the example-company bucket.

Step 4.1: Grant permissions to list all buckets

In this step, you create a policy that grants users minimum permissions. With the minimum permissions, users can list all buckets owned by the Alibaba Cloud account . You also attach the policy to the Staff group, so you grant the group permission to get a list of buckets owned by the primary account.

1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
2. Create a policy `AllowGroupToSeeBucketListInConsole`.
   a. From the left-side navigation pane, click Policies, and then click Create Authorization Policy.
   b. Click Blank Template.
   c. In the Authorization Policy Name field, enter `AllowGroupToSeeBucketListInConsole`.
   d. In the `Policy Content` field, copy and paste the following policy.

```
{
" Version ": " 1 ",
" Statement ": [
 {
   " Effect ": " Allow ",
   " Action ": [
     " oss : ListBucket  s "
   ],
   " Resource ": [
     " acs : oss :*:*:*"
   ]
 }
]
```

```
}
```

 Note:

- A policy is in JSON format. The fields in the policy are described as follows:
  - Statement: This attribute is an array of objects, and each object describes a permission using a collection of name value pairs.
  - Effect: This attribute value determines whether a specific permission is allowed or denied.
  - Action: This attribute specifies the type of access. In the policy, the `oss:ListBuckets` is a predefined OSS action, which returns a list of all buckets owned by the authenticated sender.
- We recommend you use *RAM policy editor* to generate a RAM policy quickly. For more information about RAM policies, see *How to create a RAM policy*.

3. Attach the `AllowGroupToSeeBucketListInConsole` policy to the Staff group.

   For detailed procedures on attaching a policy, see the *Attach policies to a RAM group* section of Attach policies to a RAM user in the RAM Quick Start.

   You can attach policies to RAM users and groups in the RAM console. In this example, we attach the policy to the group, because we want both Anne and Leo to be able to list the buckets.

4. Test the permission.

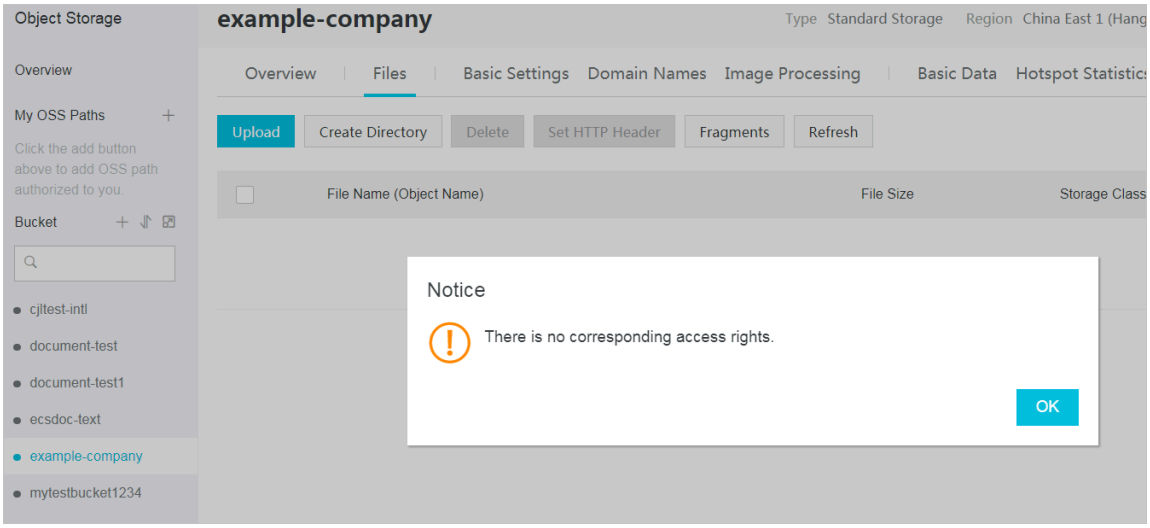
   a. Open the RAM user logon page, and log on to the RAM console with Anne’s or Leo’s credentials.

   b. Open the OSS console.

      The console lists all of the buckets.

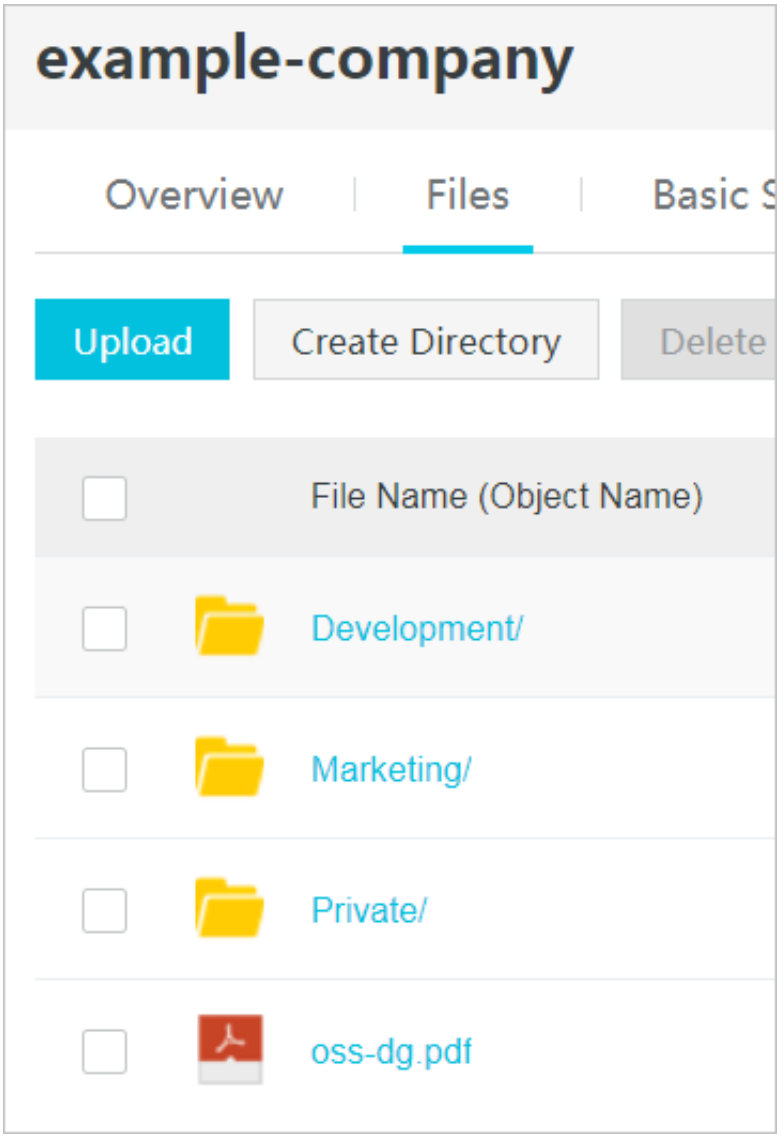
   c. Click the example-company bucket, and then click the Files tab.

      A message box is displayed, indicating that you have no corresponding access rights.



Step 4.2: Grant permissions to list root-level content of a bucket

In this step, you grant permissions to allow all users to list all the items in the bucket example-company. When users click the example-company in the OSS console, they can see the root-level items in the bucket.



1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
2. Replace the existing policy `AllowGroup ToSeeBucke tListInCon sole` that is attached to the Staff group with the following policy. The following policy also allows the `oss : ListObject s` operation. Remember to replace example-company in the policy Resource with the name of your bucket.

   For detailed procedures, see the *Modify a custom authorization policy* section of Authorization policies in the RAM User Guide. Note that you can modify a RAM policy a maximum of five times. If this is exceeded, you must delete the policy, created a new one, and then attach the policy to the Staff group again.

```
{
   " Version ": " 1 ",
   " Statement ": [
     {
       " Effect ": " Allow ",
       " Action ": [
```

```
          " oss : ListBucket  s ",
          " oss : GetBucketA  cl "
        ],
        " Resource ": [
          " acs : oss :*:*:*"
        ],
        " Condition ": {}
      },
      {
        " Effect ": " Allow ",
        " Action ": [
          " oss : ListObject  s "
        ],
        " Resource ": [
          " acs : oss :*:*: example - company "
        ],
        " Condition ": {
          " StringLike ": {
            " oss : Prefix ": [
              ""
            ],
            " oss : Delimiter ": [
              "/"
            ]
          }
        }
      }
    ]
}
```

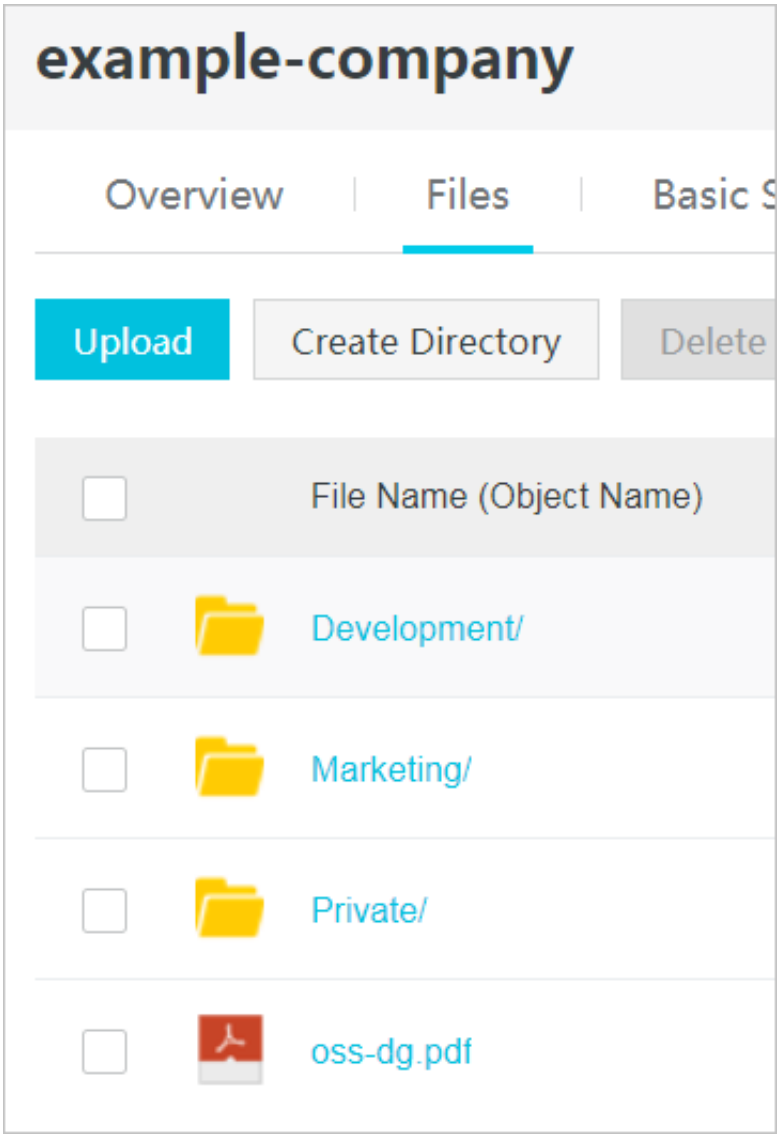
Note:

- To list bucket content, users need permission to call the `oss : ListObject s` operation. To make sure that they see only the root-level content, we add a condition that users must specify an empty prefix in the request, that is, they cannot click any of the root-level folders. We also add a condition to require folder-style access by requiring user requests to include the delimiter parameter with the value `/`.
- When a user logs on to the OSS console, the console checks the user's identities for access to OSS. To support bucket operations in the console, we also need to add the `oss : GetBucketA  cl` operation.

3. Test the updated permissions.
   a. Open the RAM user logon page, and log on to the RAM console with Anne’s or Leo’s credentials.
   b. Open the OSS console.

      The console lists all of the buckets.
   c. Click the example-company bucket, and then click the Files tab.

      The console lists all the root-level items.



   d. Click any of the folders or the `object oss-dg.pdf`.

      A message box is displayed, indicating that you have no corresponding access rights.

Summary of the group policy

The result of the group policy that you have added is to grant the RAM users Anne and Leo the following minimum permissions:

- The ability to list all buckets owned by the primary account.
- The ability to see all root-level items in the example-company bucket.

However, they still have limited access. In the following section, we grant further user-specific permissions including:

- Provide Anne the ability to get and put objects in the *Developmen t* folder.
- Provide Bob the ability to get and put objects in the *Finance* folder.

For user-specific permissions, you attach a policy to the specific user, not to the entire group. In the following section, you grant Anne permission to work within the *Developmen t* folder. You can repeat the steps to grant similar permission to Leo to work in the *Finance* folder.

Step 5: Grant RAM user Anne specific permissions

In this step, we grant additional permissions to Anne so that she can see the content of the *Developmen t* folder and get and put objects in the folder.

Step 5.1: Grant RAM user Anne permission to list the *Developmen t* folder content

For Anne to list the *Developmen t* folder content, you must attach a policy to her that grants permission for the `oss : ListObject s` action on the example-company bucket, and includes the condition that user must specify the prefix *Developmen t /* in the request.

1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
2. Create a policy `AllowListB ucketsIfSp ecificPref ixIsInclud ed` that grants the RAM user Anne permission to list the *Developmen t* folder content.
   a. From the left-side navigation pane, click Policies, and then click Create Authorization Policy.
   b. Click Blank Template.
   c. In the `Authorizat ion Policy Name` field, enter `AllowListBucketsIfSpecificPrefixIsIncluded`.
   d. In the `Policy Content` field, copy and paste the following policy.

```
{
" Version ": " 1 ",
```

```
" Statement ": [
 {
   " Effect ": " Allow ",
   " Action ": [
     " oss : ListObject  s "
   ],
   " Resource ": [
     " acs : oss :*:*: example - company "
   ],
   " Condition ": {
     " StringLike ": {
       " oss : Prefix ": [
         " Developmen  t /*"
       ]
     }
   }
 }
]
}
```

3. Attach the policy to the RAM user Anne.

   For detailed procedures on attaching a policy, see *Attach policies to a RAM user* in the RAM Quick Start.

4. Test Anne's permissions.

   a. Open the RAM user logon page, and log on to the RAM console with Anne's credentials.

   b. Open the OSS console. The console lists all of the buckets.

   c. Click the example-company bucket, and then click the Files tab. The console lists all the root-level items.

   d. Click the *Developmen t /* folder. The console lists the objects in the folder.

Step 5.2 Grant RAM User Anne permissions to get and put objects in the *Developmen t* folder

For Anne to get and put objects in the *Developmen t* folder, you must grant her permission to call the `oss : GetObject` and `oss : PutObject` actions, and includes the condition that user must specify the prefix *Developmen t /* in the request.

1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
2. Replace the policy `AllowListB ucketsIfSp ecificPref ixIsInclud ed` you created in the previous step with the following policy.

   For detailed procedures, see the *Modify a custom authorization policy* section of Authorization policies in the RAM User Guide. Note that you can modify a RAM

policy a maximum of five times. If this is exceeded, you must delete the policy, created a new one, and then attach the policy to the user again.

```
{
   " Version ": " 1 ",
   " Statement ": [
     {
       " Effect ": " Allow ",
       " Action ": [
         " oss : ListObject  s "
       ],
       " Resource ": [
         " acs : oss :*:*: example - company "
       ],
       " Condition ": {
         " StringLike ": {
           " oss : Prefix ": [
             " Developmen  t /*"
           ]
         }
       }
     },
     {
       " Effect ": " Allow ",
       " Action ": [
         " oss : GetObject ",
         " oss : PutObject ",
         " oss : GetObjectA  cl "
       ],
       " Resource ": [
         " acs : oss :*:*: example - company / Developmen  t /*"
       ],
       " Condition ": {}
     }
   ]
 }
```

Note:
When a user logs on to the OSS console, the console checks the user's identities for access to the OSS service. To support bucket operations in the console, we also need to add the `oss : GetObjectA  cl` action.

3. Test the updated policy.
   a. Open the RAM user logon page, and log on to the RAM console with Anne's credentials.
   b. Open the OSS console.

      The console lists all of the buckets.
   c. In the OSS console, verify that Anne can now add an object and download an object in the *Developmen t* folder.

Step 5.3 Explicitly deny RAM user Anne permissions to any other folders in the bucket

RAM user Anne can now list the root-level content in the example-company bucket, and get and put objects in the *Developmen t* folder. If you want to strictly restrict the access permissions, you can explicitly deny Anne's access to any other folders in the bucket. If other policies grant Anne's access to any other folders in the bucket, this explicit policy overrides those permissions.

You can add the following statement to the RAM user Anne's policy `AllowListB ucketsIfSp ecificPref ixIsInclud ed`. The following statement requires all requests that Anne sends to OSS to include the prefix parameter, and the parameter value can be either *Developmen t /** or an empty string.

```
{
      " Effect ": " Deny ",
      " Action ": [
        " oss : ListObject  s "
      ],
      " Resource ": [
        " acs : oss :*:*: example - company "
      ],
      " Condition ": {
        " StringNotL  ike ": {
          " oss : Prefix ": [
            " Developmen  t /*",
            ""
          ]
        }
      }
    }
```

Follow the preceding step to update the policy `AllowListB ucketsIfSp ecificPref ixIsInclud ed` that you created for RAM user Anne. Copy and paste the following policy to replace the existing one.

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss : ListObject  s "
      ],
      " Resource ": [
        " acs : oss :*:*: example - company "
      ],
      " Condition ": {
        " StringLike ": {
          " oss : Prefix ": [
            " Developmen  t /*"
          ]
        }
      }
    },
    {
      " Effect ": " Allow ",
```

```
      " Action ": [
        " oss : GetObject ",
        " oss : PutObject ",
        " oss : GetObjectA  cl "
      ],
      " Resource ": [
        " acs : oss :*:*: example - company / Developmen  t /*"
      ],
      " Condition ": {}
    },
    {
      " Effect ": " Deny ",
      " Action ": [
        " oss : ListObject  s "
      ],
      " Resource ": [
        " acs : oss :*:*: example - company "
      ],
      " Condition ": {
        " StringNotL  ike ": {
          " oss : Prefix ": [
            " Developmen  t /*",
            ""
          ]
        }
      }
    }
  ]
}
```

Step 6: Grant RAM user Leo specific permissions

Now you want to grant Leo permission to the *Marketing* folder. Follow the steps you used earlier to grant permissions to Anne, but replace the *Developmen t* folder with the *Marketing* folder. For detailed procedures, see Step 5: Grant RAM user Anne specific permissions.

Step 7: Secure the Private folder

In this example, you have only two users. In this example, you have only two users. You have granted all the minimum required permissions at the group level. In addition, you have granted user-level permissions only when you really need permissions at the individual user level. This approach helps minimize the effort of managing permissions. As the number of users increases, we want to make sure that we do not accidentally grant a user permission to the *Private* folder. Therefore we need to add a policy that explicitly denies access to the *Private* folder. An explicit denial overrides any other permissions. To make sure that the *Private* folder remains private, you can add the following two deny statements to the group policy:

- Add the following statement to explicitly deny any action on resources in the *Private* folder (*example - company / Private /*).

```
{
    " Effect ": " Deny ",
    " Action ": [
      " oss :*"
    ],
    " Resource ": [
      " acs : oss :*:*: example - company / Private /*"
    ],
    " Condition ": {}
  }
```

- You also deny permission for the `ListObject s` action when the request specifies the *Private / prefix* . In the console, if Anne or Leo clicks the *Private* folder, this policy causes OSS to return an error response.

```
{
    " Effect ": " Deny ",
    " Action ": [
      " oss : ListObject  s "
    ],
    " Resource ": [
      " acs : oss :*:*:*"
    ],
    " Condition ": {
      " StringLike ": {
        " oss : Prefix ": [
          " Private /"
        ]
      }
    }
  }
```

- Replace the Staff group policy `AllowGroup ToSeeBucke tListInCon sole` with an updated policy that includes the preceding deny statements. After the updated policy is applied, none of the users in the group can access the Private folder in your bucket.

  1. Log on to the *RAM console* with your Alibaba Cloud account credentials.
  2. Replace the existing policy `AllowGroup ToSeeBucke tListInCon sole` that is attached to the Staff group with the following policy. Remember to replace example-company in the policy Resource with the name of your bucket.

```
{
" Version ": " 1 ",
" Statement ": [
 {
   " Effect ": " Allow ",
   " Action ": [
     " oss : ListBucket  s ",
     " oss : GetBucketA  cl "
```

```
    ],
    " Resource ": [
      " acs : oss :*:*:*"
    ],
    " Condition ": {}
  },
  {
    " Effect ": " Allow ",
    " Action ": [
      " oss : ListObject  s "
    ],
    " Resource ": [
      " acs : oss :*:*: example - company "
    ],
    " Condition ": {
      " StringLike ": {
        " oss : Prefix ": [
           ""
         ],
        " oss : Delimiter ": [
          "/"
        ]
      }
    }
  },
  {
    " Effect ": " Deny ",
    " Action ": [
      " oss :*"
    ],
    " Resource ": [
      " acs : oss :*:*: example - company / Private /*"
    ],
    " Condition ": {}
  },
  {
    " Effect ": " Deny ",
    " Action ": [
      " oss : ListObject  s "
    ],
    " Resource ": [
      " acs : oss :*:*:*"
    ],
    " Condition ": {
      " StringLike ": {
        " oss : Prefix ": [
          " Private /"
        ]
      }
    }
  }
]
}
```

Cleanup

After you finish the tutorial, remove the users Anne and Leo in the RAM console.

For detailed procedures, see *Delete a RAM user* section of Users in the RAM User Guide.

To avoid any unnecessary charges, delete the objects and the bucket that you created for this tutorial.

14.3.2 RAM policy

Resource Access Management (RAM) is a service provided by Alibaba Cloud for resource access control. RAM policies are configured based on users. By configuring RAM policies, you can manage multiple users in a centralized manner and control the resources that can be accessed by the users. For example, you can control the permission of a user so that the user can only read a specified bucket. A RAM user belongs to the Alibaba Cloud account under which it was created, and does not own any actual resources. That is, all resources belong to the corresponding Alibaba Cloud account.

Policy examples

- Policies that grant full permissions

  A policy that grants full permissions allows applications to perform all operations on OSS.

   Warning:
  We recommend that you do not use a policy that grants full permissions for mobile applications because it is not secure.

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

Operations on OSS	Result
List all created buckets.	Success
Upload an object without a prefix, such as text.txt.	Success
Download an object without a prefix, such as text.txt.	Success

Operations on OSS	Result
Upload an object with a prefix, such as user1/test.txt.	Success
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

- Read-only policies for all objects

  A read-only policy indicates that an application can list and download all objects in the bucket `app - base - oss`.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss /*", " acs
 : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed
Upload an object without a prefix, such as text.txt.	Failed
Download an object without a prefix, such as test.txt.	Success
Upload an object with a prefix, such as user1/test.txt.	Failed
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

- Read-only policies for objects with a specified prefix

  This kind of policy indicates that an application can list and download objects with the `user1 /` prefix in the bucket `app - base - oss` but cannot download objects with other prefixes. Using this kind of policy, you can isolate applications with different prefixes in a bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed
Upload an object without a prefix, such as text.txt.	Failed
Download an object without a prefix, such as text.txt.	Failed
Upload an object with a prefix, such as user1/test.txt.	Failed
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

- Write-only policies for all objects

  A write-only policy for all objects indicates that an application can upload objects to the bucket `app - base - oss` .

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : PutObject "
```

```
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss /*", " acs
 : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed
Upload an object without a prefix, such as text.txt.	Success
Download an object without a prefix, such as text.txt.	Failed
Upload an object with a prefix, such as user1/test.txt.	Success
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

- Write-only policies for objects with a specified prefix

  This kind of policy indicates that an application can upload objects with the prefix `user1 /` to the bucket `app - base - oss` . However, the application cannot upload objects with other prefixes. Using this kind of policy, you can isolate applications with different prefixes in a bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : PutObject "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed

Operations on OSS	Result
Upload an object without a prefix, such as text.txt.	Failed
Download an object without a prefix, such as test.txt.	Failed
Upload an object with a prefix, such as user1/test.txt.	Success
Download an object with a prefix, such as user1/test.txt.	Failed
List objects without prefixes, such as test.txt.	Failed
List objects with prefixes, such as user1 /test.txt.	Failed

- Read/write policies for all objects

  A read/write policy for all objects indicates that an application can upload objects to the bucket `app - base - oss` and list, download, and delete all objects in the bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : PutObject ",
          " oss : DeleteObje  ct ",
          " oss : ListParts ",
          " oss : AbortMulti  partUpload ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss /*", " acs
 : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed
Upload an object without a prefix, such as text.txt.	Success
Download an object without a prefix, such as text.txt.	Success

Operations on OSS	Result
Upload an object with a prefix, such as user1/test.txt.	Success
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

- Read/write policies for objects with a specified prefix

  This kind of policy indicates that an application can upload objects with the prefix `user1 /` to the bucket `app - base - oss` and list, download, and delete all objects with the prefix in the bucket. However, the application cannot perform read or write operations on objects with other prefixes. Using this kind of policy, you can isolate applications with different prefixes in a bucket.

```
{
    " Statement ": [
      {
        " Action ": [
          " oss : GetObject ",
          " oss : PutObject ",
          " oss : DeleteObje  ct ",
          " oss : ListParts ",
          " oss : AbortMulti  partUpload ",
          " oss : ListObject  s "
        ],
        " Effect ": " Allow ",
        " Resource ": [" acs : oss :*:*: app - base - oss / user1 /
*", " acs : oss :*:*: app - base - oss "]
      }
    ],
    " Version ": " 1 "
  }
```

Operations on OSS	Result
List all created buckets.	Failed
Upload an object without a prefix, such as text.txt.	Failed
Download an object without a prefix, such as text.txt.	Failed
Upload an object with a prefix, such as user1/test.txt.	Success

Operations on OSS	Result
Download an object with a prefix, such as user1/test.txt.	Success
List objects without prefixes, such as test.txt.	Success
List objects with prefixes, such as user1 /test.txt.	Success

Complex policy examples

```
{
    " Version ": " 1 ",
    " Statement ": [
        {
            " Action ": [
                " oss : GetBucketA  cl ",
                " oss : ListObject  s "
            ],
            " Resource ": [
                " acs : oss :*: 1775305056  529849 : mybucket "
            ],
            " Effect ": " Allow ",
            " Condition ": {
                " StringEqua  ls ": {
                    " acs : UserAgent ": " java - sdk ",
                    " oss : Prefix ": " foo "
                },
                " IpAddress ": {
                    " acs : SourceIp ": " 192 . 168 . 0 . 1 "
                }
            }
        },
        {
            " Action ": [
                " oss : PutObject ",
                " oss : GetObject ",
                " oss : DeleteObje  ct "
            ],
            " Resource ": [
                " acs : oss :*: 1775305056  529849 : mybucket / file
 *"
            ],
            " Effect ": " Allow ",
            " Condition ": {
                " IpAddress ": {
                    " acs : SourceIp ": " 192 . 168 . 0 . 1 "
                }
            }
        }
    ]
}
```

The preceding example describes a complex authorization policy. By using this policy , a user can authorize other users through RAM or STS. In the policy, a statement is

included (a policy can include multiple statements), in which Action, Resource, Effect , and Condition are specified.

This policy grant permissions to authorized users so that they can access your resources, such as `mybucket` and `mybucket / file *`. In addition, this policy supports the following operations: GetBucketAcl, GetBucket, PutObject, GetObject, and DeleteObject. Conditions included in Condition indicate that authentication is successful and authorized users can access related resources only when UserAgent is java-sdk and the source IP address is 192.168.0.1. The Prefix condition is used when the GetBucket (ListObjects) action is performed. For more information about the field, see OSS API documentation.

Version

The Version field specifies the version of the policy. For the configuration method in this document, it is set to `1` .

Statement

A statement describes the authorization semantics. According to different scenarios , a statement can include multiple semantics which include Action, Effect, Resource , and Condition individually. When receiving a request, the system checks all Statements in the policy. All Statements that match the request are classified into two categories based on their Effect settings: Allow or Deny, in which Deny statements have higher priority when the system determines whether the authentication is successful. If all matched statements are classified into Allow, the request passes the authentication. If a matched statement is classified into Deny, or no statement matches the request, the request is rejected.

Action

Actions can be classified into three categories:

- Service-level actions: include the GetService action used to list the buckets owned by a user.
- Bucket-level actions: indicate actions performed on buckets, such as oss: PutBucketAcl and oss:GetBucketLocation. The name of each action corresponds to an API.
- Object-level actions: indicate actions performed on objects, such as oss:GetObject, oss:PutObject, oss:DeleteObject, and oss:AbortMultipartUpload.

To authorize a type of actions on objects, you can select one or more of the preceding actions. In addition, all action names must be prefixed with `oss :`, as shown in the preceding example. The Action field is a list that can include multiple actions. The following tables show the mapping relationship between actions and APIs.

- Service-level actions

API	Action
GetService（ListBuckets）	oss:ListBuckets

- Bucket-level actions

API	Action
PutBucket	oss:PutBucket
GetBucket (ListObjects)	oss:ListObjects
PutBucketAcl	oss:PutBucketAcl
DeleteBucket	oss:DeleteBucket
GetBucketLocation	oss:GetBucketLocation
GetBucketAcl	oss:GetBucketAcl
GetBucketLogging	oss:GetBucketLogging
PutBucketLogging	oss:PutBucketLogging
DeleteBucketLogging	oss:DeleteBucketLogging
GetBucketWebsite	oss:GetBucketWebsite
PutBucketWebsite	oss:PutBucketWebsite
DeleteBucketWebsite	oss:DeleteBucketWebsite
GetBucketReferer	oss:GetBucketReferer
PutBucketReferer	oss:PutBucketReferer
GetBucketLifecycle	oss:GetBucketLifecycle
PutBucketLifecycle	oss:PutBucketLifecycle
DeleteBucketLifecycle	oss:DeleteBucketLifecycle
ListMultipartUploads	oss:ListMultipartUploads
PutBucketCors	oss:PutBucketCors
GetBucketCors	oss:GetBucketCors
DeleteBucketCors	oss:DeleteBucketCors

API	Action
PutBucketReplication	oss:PutBucketReplication
GetBucketReplication	oss:GetBucketReplication
DeleteBucketReplication	oss:DeleteBucketReplication
GetBucketReplicationLocation	oss:GetBucketReplicationLocation
GetBucketReplicationProgress	oss:GetBucketReplicationProgress

- Actions on objects

API	Action
GetObject	oss:GetObject
HeadObject	oss:GetObject
PutObject	oss:PutObject
PostObject	oss:PutObject
InitiateMultipartUpload	oss:PutObject
UploadPart	oss:PutObject
CompleteMultipart	oss:PutObject
DeleteObject	oss:DeleteObject
DeleteMultipartObjects	oss:DeleteObject
AbortMultipartUpload	oss:AbortMultipartUpload
ListParts	oss:ListParts
CopyObject	oss:GetObject,oss:PutObject
UploadPartCopy	oss:GetObject,oss:PutObject
AppendObject	oss:PutObject
GetObjectAcl	oss:GetObjectAcl
PutObjectAcl	oss:PutObjectAcl
RestoreObject	oss:RestoreObject

Resource

The resource field indicates specified resources or a kind of resources (which can be represented by a wildcard *). The format of a resource is as follows: `acs : oss :{ region }:{ bucket_own er }:{ bucket_nam e }/{ object_nam e }`. The "/{object_name}" part is not required for the names of bucket-level actions. The

format of a resource for a bucket-level action is as follows: `acs : oss :{ region }:{ bucket_own er }:{ bucket_nam e }`. The Resource field is a list that can include multiple resources. The region field is not supported currently and is set to * in the preceding example.

Effect

The Effect field indicates the authorization result of this statement and has two values :Allow and Deny. When multiple statements match a request, statements in which the value of Effect is Deny has higher priority.

For example, the following policy prohibits users from deleting a specified directory but allows them to perform all operations on other objects.

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss :*"
      ],
      " Resource ": [
        " acs : oss :*:*: bucketname "
      ]
    },
    {
      " Effect ": " Deny ",
      " Action ": [
        " oss : DeleteObje  ct "
      ],
      " Resource ": [
        " acs : oss :*:*: bucketname / index /*",
      ]
    }
  ]
}
```

Condition

The Condition field indicates the conditions for the authorization policy. In the preceding example, you can set checking conditions for acs:UserAgent and acs: SourceIp, and use oss:Prefix as a condition to restrict resources when the GetBucket action is performed.

OSS supports the following conditions

Condition	Function	Valid value
acs:SourceIp	Specifies the source IP address or IP range.	IP address or IP range, wildcards (*) supported

Condition	Function	Valid value
acs:UserAgent	Specifies the http useragent header.	String
acs:CurrentTime	Specifies a valid access time.	Time in the ISO8601 format
acs:SecureTransport	Indicates whether the HTTPS protocol is used.	"true" or "false"
oss:Prefix	Indicates the prefix used when the ListObjects action is performed.	Valid object name

Best practice

OSS provides *Ram Policy Editor* that can help you generate a RAM policy quickly.

You can also *Grant permissions with a simple policy* by using ossbrowser, a graphical management tool to authorize a RAM user so that it can access specified buckets or directories.

For more examples of configuring authorization policies in different scenarios, see *Tutorial: control access to buckets and objects* and *Authorization for OSS*.

14.4 Bucket policy

You can configure bucket policies to authorize users to access your OSS buckets. Compared with a RAM policy, bucket policies can be directly configured by the bucket owner on the console for access authorization.

Bucket policies are suitable for the following scenarios:

- Authorize RAM users of other accounts to access your OSS resources.

  You can authorize RAM users of other accounts to access your OSS resources.
- Authorize anonymous users to access your OSS resources using specific IP addresses or IP ranges.

  In some cases, you must authorize anonymous users to access OSS resources using specific IP addresses or IP ranges. For example, confidential documents of an enterprise are only allowed to be accessed within the enterprise but not in other regions. Previously, configuring RAM policies for every user was a tedious and complex task because of the potential for a large number of internal users. To

resolve this issue, you can configure access policies with IP restrictions based on bucket policies to authorize a large number of users easily and efficiently.

For more information about the configuration methods of bucket policies and video tutorials, see *Use bucket policies to authorize other users to access OSS resources*.

14.5 Cross-account authorization

14.5.1 Overview

OSS provides multiple cross-account authorization methods to allow users using different accounts to access OSS resources.

The ACL for all OSS resources is private by default. The owner of a OSS resource can grant permissions to users using different accounts so that they can access the OSS resource. The following cross-account authorization methods can be used to allow other users to access OSS resources.

- *Authorize a RAM user under another Alibaba Cloud account by adding a bucket policy*: A bucket policy authorize users based on resources. Compared with RAM policies, bucket policies can be easily configured in the graphical console. By configuring bucket policies, you can directly authorize other users so that they can access your bucket even you do not have permissions for RAM operations. You can configure bucket policies to grant bucket access permissions with IP address restrictions to anonymous users and RAM users under other accounts.
- *Tutorial: Grant cross-account bucket permissions*: The RAM administrator can configure a RAM role and add the ID of another Alibaba Cloud account as trusted ID. After that, the RAM administrator can grant OSS access configuration permissions to the RAM role to share OSS resources to users under the Alibaba Cloud account.

14.5.2 Tutorial:Authorize a RAM user under another Alibaba Cloud account by adding a bucket policy

The ACL for an OSS resource is private by default. To allow another user to access your OSS resources, you can grant permissions for the user to access your bucket by adding a bucket policy.

For example: Company A wants its partner, company B, to access its OSS resources, but company A does not want to create a RAM user under its Alibaba Cloud account for this requirement. In this case, company A can grant permissions for company B

to access the bucket of company A by adding a bucket policy. After being authorized , company B can access an OSS resource owned by company A by adding the path of the resource in the OSS console.

Add a bucket policy for the RAM user of company B

- Follow these steps by using the Alibaba Cloud account of company B:
  1. Log on to the *RAM console* and create a RAM user. For more information, see *Create a RAM user*.
  2. In the RAM console, click Users.
  3. Click the created RAM user and record its UID.
- Follow these steps using the Alibaba Cloud account of company A:
  1. Log on to the *OSS console*.
  2. In the left-side bucket list, click the name of the bucket that you want to grant permissions for company B.
  3. Click Filles > Authorize > Authorize.
  4. In the Authorize dialog box, enter the policy information. Select Other Account for Accounts, and enter the UID of the RAM user created by company B. For more information about other parameters, see *Use bucket policies to authorize other users to access OSS resources*.
  5. Click OK.

Log on to OSS with the RAM user of company B and add the resource path

After a bucket policy is added, you must log on to the OSS console with the RAM user of company B and add the access path of the OSS resource of company A. To add the access path, follow these steps:

1. Log on to Alibaba Cloud console with the RAM user of company B through the *RAM user logon link*.
2. Open the *OSS console*.

3. In the left-side menu, click "+" on the right of My OSS Paths. In the displayed Add Authorized OSS Path dialog box, add the following information:
   - Region: Select the region of the bucket that company A allows company B to access.
   - OSS path: Add the resource path that company A allows company B to access. The format of an OSS path is as follows: bucket/object-prefix. For example, if company A allows company B to access only the `abc` folder in the `aliyun` bucket, the OSS path is `aliyun / abc` .

   You can also *Create an AccessKey* for the RAM user, and use *ossutil* or *ossbrowser* with the AccessKey to access the authorized bucket.

References

You can also grant permissions for other users to access your OSS resources in the following methods:

- *Tutorial: Authorize a RAM user under another Alibaba Cloud account by creating a RAM role*

15 Manage logs

15.1 Set access logging

A large number of access logs are generated when an object is accessed. After the access logging function is enabled for a bucket, OSS automatically generates an object in the specified bucket (target bucket) to store the access logs on an hourly basis. The generated object complies with the OSS naming conventions. You can analyze the access logs by using Alibaba Cloud Data Lake Analytics or establishing a Spark cluster. You can also configure lifecycle rules for the target bucket to convert the storage class of the log object to Archive for archiving.

Note:

For more information about the APIs related to the logging function, see the following documents:

- Set access logging: *PutBucketLogging*
- Delete logging settings: *DeleteBucketLogging*
- View logging settings: *GetBucketLogging*

Setting methods

Methods	Description
OSS console	The OSS console is an intuitive web application that allows you to set access logging with ease.
Java SDK	OSS provides varies of SDK demos in different languages.
Python SDK	
PHP SDK	
Go SDK	
C SDK	
.NET SDK	
Node.js SDK	
Ruby SDK	

Naming convention for objects that store access logs

```
< TargetPref  ix >< SourceBuck  et > YYYY - mm - DD - HH - MM - SS -
 UniqueStri  ng
```

Fields included in the naming convention are described as follows:

- `TargetPref ix` indicates the name prefix of the object that stores access logs. This field is defined by the user and can be left blank.
- `YYYY - mm - DD - HH - MM - SS` indicates the year, month, day, hour, minute, and second when the object is created (the number of digits must be noted).
- `UniqueStri ng` is a string (UUID) generated by OSS, which is used to uniquely identify the log object.

An example name of an object that stores OSS access logs is as follows:

```
MyLog - oss - example201  7 - 09 - 10 - 04 - 00 - 00 - 0000
```

In the preceding example:

- `MyLog -` is the object prefix specified by the user.
- `oss - example` is the name of the source bucket.
- `2017 - 09 - 10 - 04 - 00 - 00` is the time when the object is created.
- `0000` is a string generated by OSS.

Log file format

The following table describes the fields that compose a log file. In a log file, these fields are combined in order from left to right and are separated by spaces.

Field	Example	Description
Remote IP	119.xx.xx.11	Indicates the IP address from which the request is initiated (note that a proxy or firewall may block this field).
Reserved	-	Indicates that this is a reserved field.
Reserved	-	Indicates that this is a reserved field.
Time	[02/May/2012:00:00:04 + 0800]	Indicates the time when OSS receives a request.

Field	Example	Description
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	Indicates the URI of a user request (including query-string).
HTTP Status	200	Indicates the HTTP status code returned by OSS.
SentBytes	5576	Indicates the amount of traffic downloaded by the user from OSS.
RequestTime (ms)	71	Indicates the amount of time used to complete the request (in ms).
Referer	`http :// www . aliyun . com / product / oss`	Indicates the requested HTTP referer.
User-Agent	curl/7.15.5	Indicates the HTTP User-Agent header.
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	Indicates the domain name that the request accesses.
Request ID	505B016950xxxxxx032593A4	Indicates the UUID used to uniquely identify the request.
LoggingFlag	true	Indicates whether the access logging function is enabled
Requester Alibaba Cloud ID	16571xxxxxx83691	Indicates the ID of the RAM user used by the requester, which is “-“ for anonymous access.
Operation	GetObject	Indicates the request type.
Bucket	oss-example	Indicates the name of the request bucket.
Key	/aliyun-logo.png	Indicated the key that the user requests.
ObjectSize	5576	Indicates the object size.

Field	Example	Description
Server Cost Time (ms)	17	Indicates the amount of time that the OSS server used to process the request (in ms).
Error Code	NoSuchBucket	Indicates the error code returned by OSS.
Request Length	302	Indicates the length of the user request (in bytes).
UserID	16571xxxxxx83691	Indicates the ID of the bucket owner.
Delta DataSize	280	Indicates the bucket size variation, which is `-` if the bucket size has not changed.
Sync Request	-	Indicates whether the request is a CDN back-to-origin request, which is `-` if the request is not a back-to-origin request.
StorageClass	Standard	Indicates the storage class of the requested object, which can be `Standard`, `IA`, or `Archive`. If the value of this field is `-`, it indicates that the access is not performed on an object so that the storage class cannot be obtained.

Detail analysis

- The source bucket and target bucket can be the same bucket, or different buckets that are owned by the same account and belong to the same region. You can also store the logs of multiple source buckets in the same target bucket. In this case, we recommend that you specify different TargetPrefix values for logs of different source buckets.

- OSS generates an object that stores bucket access logs on an hourly basis. However, requests in the last hour may be recorded in the object generated for the previous hour or the next hour.
- Each time OSS generates an object that stores bucket access logs, a PUT operation and the storage space that the operation occupies are recorded. However, traffic generated by the PUT operation is not recorded. You can perform common OSS operations on a generated object that stores access logs.
- OSS ignores all query-string parameters prefixed by `x-`. However, these parameters are recorded in the access logs. To easily identify a specific request from a large amount of access logs, add a query-string parameter prefixed by `x-` to the URL of the request. For example:

  ```
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png
  ```

  ```
  http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png?x-user=admin
  ```

  OSS returns the same result for the preceding two requests. However, you can easily locate the request with `x-user=admin` by searching this parameter.
- A `-` may appear in any field in OSS logs. It indicates that data is unknown or the field is invalid for the current request.
- More fields will be added to the end of OSS logs in the future. We recommend that developers take note of potential compatibility issues when developing log processing tools.

16 Data encryption

16.1 Server-side encryption

This topic describes how to use the server-side encryption feature provided by OSS to encrypt and protect persistent data stored in OSS.

OSS supports server-side encryption for uploaded data. This means that when user data is uploaded, OSS encrypts the data and permanently stores the data. Then, when the data is downloaded by a user, OSS automatically decrypts the data, returns the original data to the user, and declares in the header of the returned HTTP request that the data has been encrypted on the server.

Scenarios

The following server-side encryption methods are available for different application scenarios:

- Server-side encryption that uses CMKs managed by KMS for encryption and decryption (SSE-KMS)

  When uploading an object, you can use a specified CMK ID or the default CMK managed by KMS to encrypt and decrypt a large amount of data. This method is cost-effective because you do not need to send user data to the KMS service side through networks for encryption and decryption.

  > Note:
  >
  > - Fees for API calls are incurred if you use a CMK to encrypt an object.

- Server-side encryption fully managed by OSS (SSE-OSS)

  This encryption method is a property of an object. When sending a request to upload an object or modify the metadata of an object, you can include the `X-OSS-server-side-encrpytion` header in the request and specify its value as AES256. In this method, OSS uses AES256 to encrypt each object with an individual key. Furthermore, the individual keys are encrypted by a customer master key (CMK) that is updated periodically for higher security. This method applies to encrypt or decrypt bulk data.

> **Notice:**
> Only one server-side encryption method can be used for an object at one time.

Configuration

For detailed information about server-side encryption configuration, see *Protect data by performing server-side encryption*.

Server-side encryption that uses CMKs managed by KMS for encryption and decryption

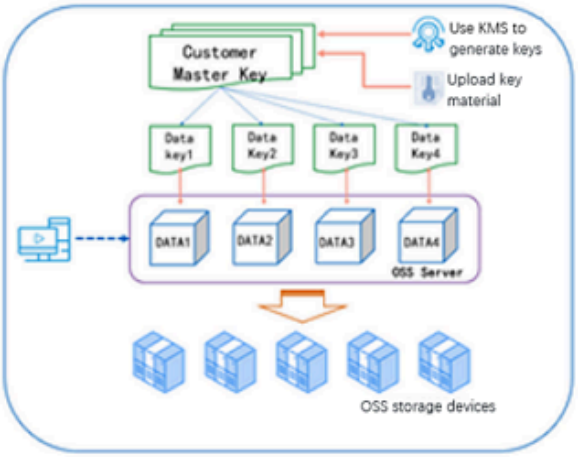
Key Management Service (KMS) is a secure and easy-to-use key protection and management service provided by Alibaba Cloud. KMS allows you to use keys in a secure manner, at minimal cost. You can view and manage keys in the KMS console.

To encrypt an object when creating it, you can include the `x-oss-server-side-encryption` header in the request and specify its value to `KMS` (which indicates that KMS is used for key management).

In addition to the usage of AES256 encryption algorithm, KMS stores the customer master key (CMK) used to encrypt data keys, generate data keys, and uses the envelope encryption mechanism to protect data from unauthorized access.

The following table shows the logic of SSE-KMS.

Server-side encryption (SSE-KMS): Supports using BYOK material for encryption



- Multi-level key envelope encryption
- Supports using BYOK material for encryption and decryption
- Keys are managed by KMS
- A random key is generated for each object

A CMK can be generated in the following methods:

- Use the default CMK managed by KMS.

  When sending a request to upload an object or modify the metadata of an object, you can include the `X-OSS-server-side-encrpytion` header in the request and specify its value as KMS without a specified CMK ID. In this method, OSS generates an individual key to encrypt each object by using the default managed CMK, and automatically decrypts the object when it is downloaded.
- Use a CMK specified by the user.

  When sending a request to upload an object or modify the metadata of an object, you can include the `X-OSS-server-side-encrpytion` header in the request, specify its value as KMS, and specify the value of `X-oss-server-side-encrpytion-key-id` to a specified CMK ID. In this method. OSS generates an individual key to encrypt each object by using the specified CMK, and adds the CMK ID used to encrypt an object into the metadata of the object so that the object is automatically decrypted when it is downloaded by an authorized user.
- Use the BYOK material of the user as the CMK.

  You can import your BYOK material into KMS as the CMK as follows:

  1. Create a CMK without key material.
  2. Import the key material from an external source.

  For more information about how to import key material, see *Import key material*.

> **Note:**
>
> - If you use a CMK to encrypt an object, the data key used in the encryption is also encrypted and is stored as the metadata of the object.
> - In server-side encryption that uses the default CMK managed by KMS, only the data in the object is encrypted. The metadata of the object is not encrypted.
> - To use a RAM user to encrypt objects with a specified CMK, you must grant the relevant permissions to the RAM user. For more information, see *Use RAM for KMS resource authorization*.

Server-side encryption fully managed by OSS

In this server-side encryption method, OSS generates and manages the keys used for data encryption, and provides strong multi-factor security measures to protect data

. AES256 (256-bit advanced encryption standard), a strong encryption algorithm, is used to encrypt data.

In this way, the encryption method becomes a property of an object. To perform server-side encryption on an object, you can include the `X-OSS-server-side-encrpytion` header in the PutObject request and specify its value as `AES256`.

APIs that support server-side encryption

Notice:
This function is in the beta testing phase. To join the testing group, contact Alibaba Cloud technical support or open a ticket.

- APIs that support server-side encryption in requests

  The `x-oss-server-side-encryption` header is supported in requests initiated by the following APIs:

  - PutObject
  - CopyObject
  - InitiateMultipartUpload

  The following table describes the HTTP headers that can be included in requests.

Header	Description	Example
x-oss-server-side-encryption	Specifies the server-side encryption method. Valid values: `AES256` and `KMS`	`x-oss-server-side-encryption: KMS` indicates that the server-side encryption uses CMKs managed by KMS.
x-oss-server-side-encryption-key-id	Specifies the ID of the CMK used to encrypt the object. This header must be specified when you use a specified CMK ID for encryption.	`x-oss-server-side-encryption-key-id: 72779642-7d88-4a0f-8d1f-1081a9cc7a fb`

Note:

- If the `x-oss-server-side-encryption` header is included in requests initiated by APIs except for PutObject, CopyObject, and InitiateMultipartUpload, OSS returns HTTP status code 400 and includes `InvalidArgument` in the error message.
- If an invalid value is specified for the `x-oss-server-side-encryption` header, OSS returns HTTP status code 400 and includes `InvalidEncryptionAlgorithmError` in the error message.

- APIs that support server-side encryption in responses

  OSS includes the `x-oss-server-side-encryption` header in responses to requests initiated by the following APIs to access objects encrypted at the server side.

  - PutObject
  - CopyObject
  - InitiateMultipartUpload
  - UploadPart
  - CompleteMultipartUpload
  - GetObject
  - HeadObject

17 Static website hosting

17.1 Configure static website hosting

In the *OSS console*, you can set up your buckets to work in static website hosting mode.

If your selected bucket is located in Hangzhou, after the configuration takes effect, the endpoint of the static website is as follows:

```
http ://< Bucket >. oss - cn - hangzhou . aliyuncs . com /
```

Note:

When you use an OSS endpoint in Mainland China regions or the Hongkong region to access a web file through the Internet , the Content-Disposition: 'attachment=filename;' is automatically added to the Response Header, and the web file is downloaded as an attachment. If you access OSS with a user domain, the Content-Disposition: 'attachment=filename;' will not be added to the Response Header. For more information about using the user domain to access OSS, see *Bind a custom domain name*.

For users to manage static websites hosted on the OSS more easily, the OSS provides two functions:

- Index Document Support

  The index document refers to the default index document (such as index.html) that is returned by the OSS when a user directly accesses the root domain name of the static website. If static website hosting mode is set for a bucket, you must specify the index document as an object in that bucket. This setting is required.

- Error Document Support

  The error document refers to the error page the OSS returns to a user if the HTTP 4XX error (such as 404 "NOT FOUND") occurs when the user attempts to access the static website but fails. If static website hosting mode is set for a bucket, you must specify the error document as an object in that bucket. This setting is optional.

For example, if a user sets:

- The index document support as index.html

- The error document support as error.html
- The bucket as oss-sample
- The endpoint as `oss-cn-hangzhou.aliyuncs.com`

Then:

- When a user accesses `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/` and `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/directory/`, it is the same as accessing `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/index.html`.
- When a user accesses `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/object`, and the object does not exist, OSS returns `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/error.html`.

Detail analysis

- Static websites are websites with web pages composed of static content, including scripts such as JavaScript executed on the client. OSS does not support content that needs to be processed by the server, such as PHP, JSP, and ASP.NET content.
- For access to a bucket-based static website through a user-defined domain name, see *Bind custom domain names*.
- When you set a bucket to static website hosting mode, you must specify an index page, the error page is optional.
- When you set a bucket to static website hosting mode, the specified index page and error page must be an object in the bucket.
- After a bucket is set to static website hosting mode, the OSS returns the index page for anonymous access to the root domain name of the static website, and returns Get Bucket results for signed access to the root domain name of the static website.
- After a bucket is set to static website hosting mode, and the user accesses the root domain name of a static website or a nonexistent object, the OSS returns a specified object to the user and bills the return traffic and requests to the bucket owner.

Reference

- API: *PutBucketWebsite*
- Console: *Static website hosting*

- Java SDK: *Static website hosting*

17.2 Tutorial: Host a static website using a custom domain name

Suppose that you want to host a static website on Alibaba Cloud Object Storage Service (OSS). You have registered a domain (for example, examplewebsite.com), and you want the requests for `http :// exampleweb site . com` and `http :// www . exampleweb site . com` to be serviced from your OSS content. Whether you have an existing static website that you want to host on OSS, or you are starting from scratch, you can use this example and learn how to host websites on Alibaba Cloud OSS.

Prerequisites

This tutorial covers the following services:

- Domain name registration

  If you do not have a registered domain name, such as `exampleweb site . com`, select a registrar to register one. Alibaba Cloud also provides domain name registration service. For more information, see *Alibaba Cloud Domain service*.

- Alibaba Cloud OSS

  You use Alibaba Cloud OSS to create buckets, upload a sample website page, configure permissions to let others see the content, and then configure the buckets for website hosting. In this example, because you want to allow requests for `http :// exampleweb site . com` and `http :// www . exampleweb site . com`, you create two buckets. You host content in only one bucket, and configure the other bucket to redirect requests to the bucket that hosts the content.

- Alibaba Cloud DNS

  As your Domain Name System (DNS) provider, you configure Alibaba Cloud DNS. In this example, you add your domain name to Alibaba Cloud DNS and define a CNAME record so that you can use your domain name instead of the OSS assigned access domain name to access your OSS buckets.

  In this example, we use Alibaba Cloud DNS. We recommend that you use Alibaba Cloud DNS. However, you can use various registrars to define a CNAME record that points to an OSS bucket.

Note:

This tutorial uses `exampleweb site . com` as a domain name. Replace this domain name with the one that you have registered.

Step 1: Register a domain

If you already have a registered domain, you can skip this step. If you have never hosted a website, your first step is to register a domain, such as `exampleweb site . com`. You can use Alibaba Cloud Domain service to register a domain.

For more information, see *Buy a domain name* in the Alibaba Cloud Domain Quick Start.

Step 2: Create and configure buckets and upload data

You create two buckets to support requests from both the root domain such as `exampleweb site . com` and subdomain such as `http :// www . exampleweb site . com`. One bucket is used to store the content, and the other bucket is used to redirect requests to the bucket that stores the content.

Step 2.1: Create two buckets

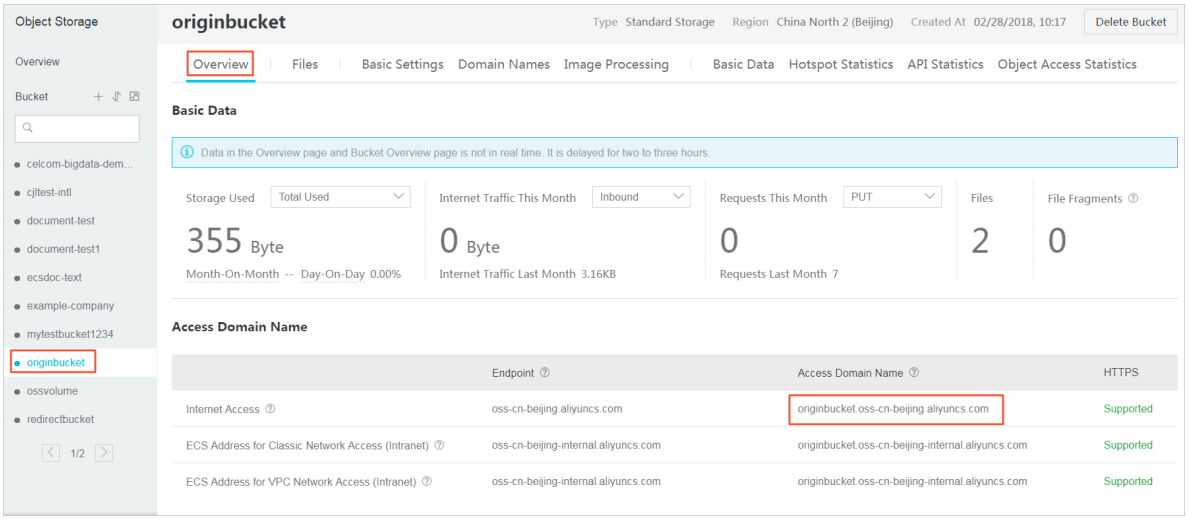
In this step, you log on to Alibaba Cloud OSS console with your Alibaba Cloud account credentials and create the following two buckets:

- originbucket: to store the content
- redirectbucket: to redirect requests to originbucket

1. Log on to the *OSS console*.
2. Create two buckets, for example, originbucket and redirectbucket, one to store the content, and the other to redirect requests to the bucket that stores the content. Set the access control list (ACL) of the two buckets to Public Read so that everyone can see the content of the buckets.

   For detailed procedures, see *Create a bucket*.
3. Make yourself a note of the access domain name of the originbucket and redirectbucket. You will use them in later steps. You can find the access domain

name of a bucket on the Overview tab page of the bucket, as shown in the following figure.



4. Upload your website data to originbucket.

   You will host your content out of the root domain bucket originbucket, and you will redirect requests for the subdomain bucket redirectbucket to the root domain bucket originbucket. You can store content in either bucket.

   For this example, you host content in the originbucket bucket. The content can be any type of files, such as text files, photos, and videos. If you have not yet created a website, then you only need two files for this example. One file is used as the homepage of the website, and the other file is used as the error page of the website.

   For example, you can create one file named index.html using the following HTML and upload it to the bucket. In a later step, you use this file name as the default homepage for your website.

```
< html >
   < head >
       < title > Hello   OSS ! </ title >
       < meta   charset =" utf - 8 ">
   </ head >
   < body >
       < p > Now   host   on   Alibaba   Cloud   OSS </ p >
       < p > This   is   the   index   page </ p >
   </ body >
 </ html >
```

   You create another file named error.html using the following HTML and upload it to the bucket. This file is used as the 404 error page of a website. In a later step, you use this file name as the default 404 page for your website.

```
< html >
< head >
```

```
   < title > Hello   OSS ! </ title >
   < meta   charset =" utf - 8 ">
</ head >
< body >
   < p > This   is   the   404   error   page </ p >
</ body >
</ html >
```

Step 2.2: Configure buckets for website hosting

When you configure a bucket for website hosting, you can access the website using the OSS assigned access domain name.

In this step, you configure originbucket as a website.

1. Log on to the *OSS console*.
2. From the bucket name list, select `originbuck et .`
3. Click the Basic Settings tab and find the `Static Page` area.
4. Click Edit, and then enter the following information:
   - Default Homepage: The index page (equivalent to index.html of the website). Only HTML files that have been stored in the bucket can be used. For this example, enter *`index . html`* .
   - Default 404 Page: The default 404 page returned when an incorrect path is accessed. Only HTML and image files that have been stored in the bucket can be used. If this field is left empty, the default 404 page is disabled. For this example, enter *`error . html`* .
5. Click Save.

Step 2.3: Configure the index page for redirect

Now that you have configured the default homepage and error page of the originbuck et, you also need to configure the default homepage of redirectbucket.

To configure the index page for redirect, follow these steps:

1. Log on to the *OSS console*.
2. From the bucket name list, select `redirectbu cket .`
3. Click the Basic Settings tab and find the `Static Page` area.
4. Click Edit, and then enter *`index . html`* in the `Default Homepage` text box.
5. Click Save.

Step 3: Bind your domain name to your OSS buckets

Now that you have your root domain `exampleweb site . com` and your OSS bucket originbucket, bind your domain to your OSS buckets so that you can access the OSS buckets using your own domain name instead of the domain name assigned by OSS.

In this example, before you bind your domain `exampleweb site . com` to your OSS bucket originbucket, you have to use the OSS assigned domain name originbucket.oss-cn-beijing.aliyuncs.com to access your bucket originbucket. After you bind your domain `exampleweb site . com` , you can use this `exampleweb site . com` to access your OSS bucket.

Similarly, you also need to bind your subdomain `www . exampleweb site . com` to your OSS bucket redirectbucket, so that you can use `www . exampleweb site . com` instead of the OSS assigned domain name originbucket.oss-cn-beijing.aliyuncs.com to access your OSS bucket.

To bind your root domain `exampleweb site . com` to your OSS bucket originbuck et, follow these steps:

1. Log on to the *OSS console*.
2. From the bucket name list, select `originbuck et .`
3. Click the Domain Names tab.
4. Click Bind User Domain to open the Bind User Domain dialog box.
5. In the `User Domain` text box, enter the root domain `examplewebsite.com`.

6. Define a CNAME record to originbucket.
   - If your domain name has been resolved under your Alibaba Cloud account, you can open the Add CNAME Record Automatically switch. Then click Submit.
   - If your domain name has not been resolved under your Alibaba Cloud primary account, the Add CNAME Record Automatically switch is disabled. Follow these steps to add a CNAME record manually, and then click Submit.
     a. Add your domain name in Alibaba Cloud DNS.
        - If your domain name is registered with Alibaba Cloud, it is automatically added to the Alibaba Cloud DNS list. You can skip this step.
     b. In the Alibaba Cloud DNS console, find your domain name.
     c. Click the domain name or click the Configure link.
     d. Click Add Record.
     e. In the `Add Record` dialog box, select `CNAME` from the `Type` drop-down box, and enter the OSS domain name of the bucket in the `Value` text box. In this example, enter originbucket.oss-cn-beijing.aliyuncs.com.
     f. Click Confirm.
7. Follow the preceding steps to bind your sub domain `www.exampleweb site.com` to your OSS bucket redirectbucket.

Step 4: Configure your website redirect

Now that you have configured your bucket for website hosting and bound your custom domain to your OSS bucket, configure the redirectbucket to redirect all requests for `http://www.exampleweb site.com` to `http://exampleweb site.com`.

To configure your website redirect, follow these steps:

1. Log on to the *OSS console*.
2. From the bucket name list, select `redirectbu cket`.
3. Click the Basic Settings tab and find the `Back-to-Origin` area.
4. Click Edit, and then click Create Rule.

5. Create the 404 redirect rule as follows:
   a. In the Back-to-Origin Type area, select `Redirect`.
   b. In the `Back-to-Origin When` area, set `HTTP Status` Code to 404.
   c. In the `Back-to-Origin URL` area, select `Add a prefix or suffix`, enter your domain name of the originbucket. In this example, enter examplewebsite.com.
   d. Click OK.

Step 5: (Optional) Speed up your website with Alibaba Cloud CDN

You can use Alibaba Cloud Content Delivery Network (CDN) to improve the performance of your website. CDN makes your website files (such as HTML, images, and video) available from data centers around the world. These are called edge nodes. When a visitor requests a file from your website, Alibaba Cloud CDN automatically redirects the request to a copy of the file at the nearest edge node. This results in faster download times than if the visitor had requested the content from a data center that is located farther away.

Alibaba Cloud CDN caches content at edge nodes for a period of time that you specify. If a visitor requests content that has been cached for longer than the expiration date, CDN checks the origin server to see if a later version of the content is available. If a later version is available, CDN copies the new version to the edge node. Changes that you make to the original content are replicated to edge nodes as visitors request the content. However, before the expiration date, the content is still in the earlier version. We recommend that you open the Auto Refresh CDN Cache switch, so that changes you make to the original content are automatically refreshed in CDN cache in real time.

To speed up originbucket with CDN, follow these steps:

1. Add a CDN domain in the CDN console. For detailed procedures, see Step 2. Add a CDN domain in *CDN quick start*.
2. Define a CNAME record in the Alibaba Cloud DNS console. For detailed procedures, see *Configure Alibaba Cloud DNS*.

3. Open the Auto Refresh CDN Cache switch in the OSS console.
   a. Log on to the *OSS console*.
   b. From the bucket name list, select `originbuck et`.
   c. Click the Domain Names tab.
   d. Open the Auto Refresh CDN Cache switch.
4. Follow the preceding steps to speed up `redirectbu cket` with CDN.

Step 6: Test the website

To verify that the website is working correctly, in your browser, try the following URLs:

URL	Result
`http :// exampleweb site . com`	Displays the index document in the originbucket.
The URL of a file that does not exist in the originbucket, for example, `http :// exampleweb site . com / abc`	Displays the error document in the originbucket.
`http :// www . exampleweb site . com`	Redirects your request to `http :// exampleweb site . com`.
`http :// www . exampleweb site . com / abc`	Redirects your request to `http :// exampleweb site . com / abc`.

Note:
In some cases, you may need to clear the cache of your web browser to see the expected behavior.

Cleanup

If you created your static website as a learning exercise only, remember to delete the Alibaba Cloud resources that you allocated to avoid unnecessary fees charged to your account. After you delete your Alibaba Cloud resources, your website is no longer available.

To clean up, follow these steps:

1. Disable and then remove the domain name in the Alibaba Cloud CDN console.
2. Delete the CNAME records in the Alibaba Cloud DNS console.
3. Delete the OSS files and buckets in the Alibaba Cloud OSS console.

18 Monitoring service

18.1 Monitoring service overview

The OSS monitoring service details metric data, including basic system operation statuses, performance, and metering. It also provides a custom alarm service to track requests, analyze usage, collect statistics on business trends, and promptly discover and diagnose system problems.

OSS metric indicators are classified into indicator groups such as basic service indicators, performance indicators, and metering indicators. For more information, see *Monitoring indicators reference*.

High real-time performance

Real-time performance monitoring can expose potential peak/valley problems, display actual fluctuations, and provide insights into the analysis and evaluation of business scenarios. OSS real-time metric indicators (excluding the metering indicator) enable minute-level collection and aggregation of metric data with an output delay of less than one minute.

Metering indicator description

The metering indicator uses the following features:

- Metering entries are collected, aggregated, and output at the hour-level.
- However, the output delay can be up to thirty minutes.
- The time of metering refers to the start time of the relevant statistical period.
- The metering acquisition cutoff time is the end time of the last metering data statistical period for the current month. If no metering data is produced during the current month, the metering data acquisition cutoff time is 00:00 on the first day of the current month.
- A maximum amount of metering entries is pushed for presentation. For precise metering data, go to Billing Management and click *Usage Records*.

For example, suppose that the user just uses the request to upload data, an average of 10 times a minute. So at 2016-05-10 08:00:00 to 2016-05-10 09:00:00 this hour of time , the measured data value of the user's number of put class requests is 600 times (10

*60 minutes), data time 2016-05-10 At 08:00:00, the data will be output at about 09: 30:00. If this data is from 2016-05-01 From 00:00:00 to the present, the last measure monitoring data, then the cut-off time of the month for the measure data acquisition is 2016-05-10 09:00:00. If the user did not produce any measurement data in May 2016 , the cut off time for the measurement collection is 2016-05-01 00:00:00.

OSS alarm service

You can set up to 1,000 alarm rules.

Alarm rules can be configured for other metric indicators, which can then be added to alarm monitoring. Additionally, multiple alarm rules may be configured for a single metric indicator.

- For information about the alarm service, see *Alarm service overview*.
- For instructions about how to use the OSS alarm service, see *Alarm service user guide*.
- For more information about OSS metric indicators, see *Monitoring indicators reference*.

Metric data retention policy

Metric data is retained for 31 days and is automatically cleared upon expiration. To analyze metric data offline, or download and store historical metric data for longer than 31 days, use the appropriate tool or input code to read the data storage of CloudMonitor. For more information, see *Metric data access through the API*.

The console displays metric data up until the past seven days. To view historical metric data earlier than seven days, use the CloudMonitor SDK. For more information, see *Metric data access through the API*.

Metric data access through the API

The API of CloudMonitor allows you to access OSS metric data. For usage information , see:

- *CloudMonitor API Reference*
- *Cloud monitoring SDK Reference*
- *Metric item reference*

Monitoring, diagnosis, and troubleshooting

The following documentation provides monitoring, diagnosis, and troubleshooting details related to OSS management:

- *Real-time service monitoring*

  Describes how to use the monitoring service to monitor the running status and performance of OSS.

- *Tracking and diagnosis*

  Describes how to use the OSS monitoring service and logging function to diagnose problems, and how to associate relevant information in log files for tracking and diagnosis.

- *Troubleshooting*

  Describes typical problems and corresponding troubleshooting methods.

Considerations

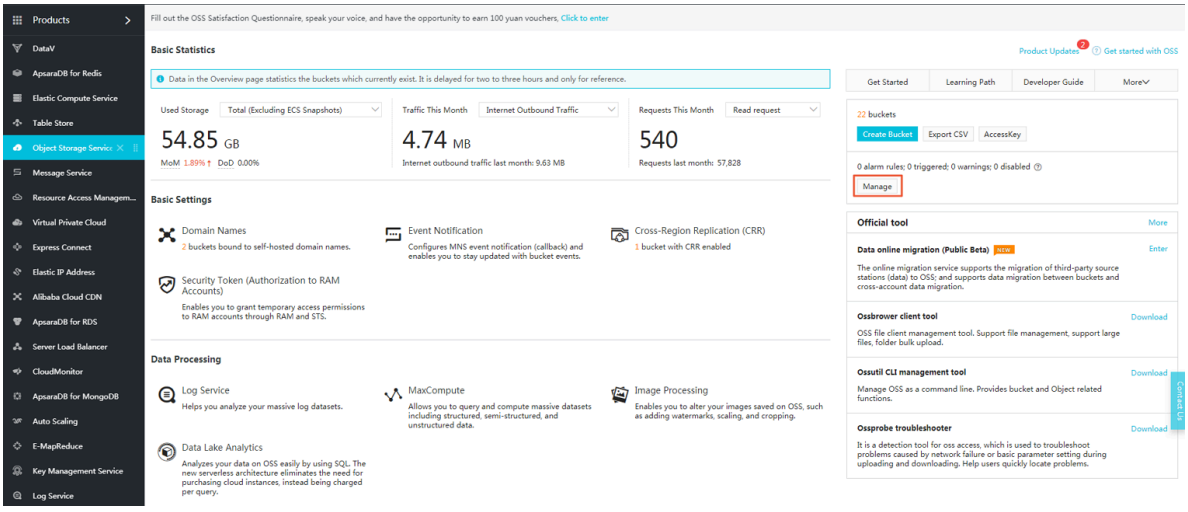
OSS buckets must be globally unique. If, after deleting a bucket, you create another bucket with the same name as the deleted bucket, the monitoring and alarm rules set for the deleted bucket are applied to the new bucket.

18.2 Monitoring service user guide

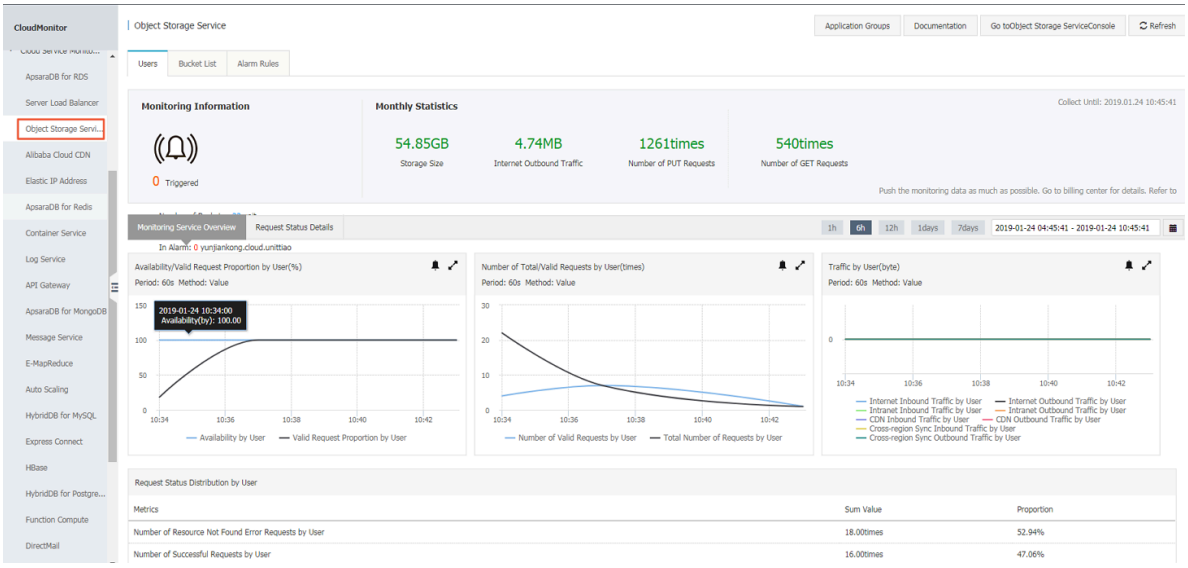
OSS monitoring entry

The OSS monitoring service is available on the Alibaba Cloud Console. You can access the OSS monitoring service in either of the following ways:

- Log on to the *OSS console* and then click Manage on the right side of OSS overview page.



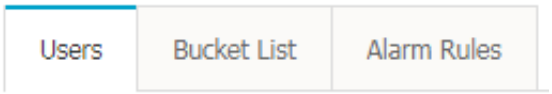
- Log on to the *CloudMonitor console* to view the OSS monitoring service.



OSS monitoring page

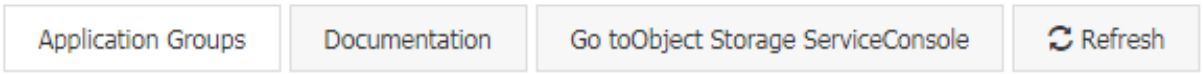
The OSS monitoring page consists of the following three tabs:

- Users
- Bucket list
- Alarm rules



The OSS monitoring page does not support automatic refresh. You can click Refresh in the upper-right corner to display the latest data.

Click Go to Object Storage Service Console to log on to the OSS console.

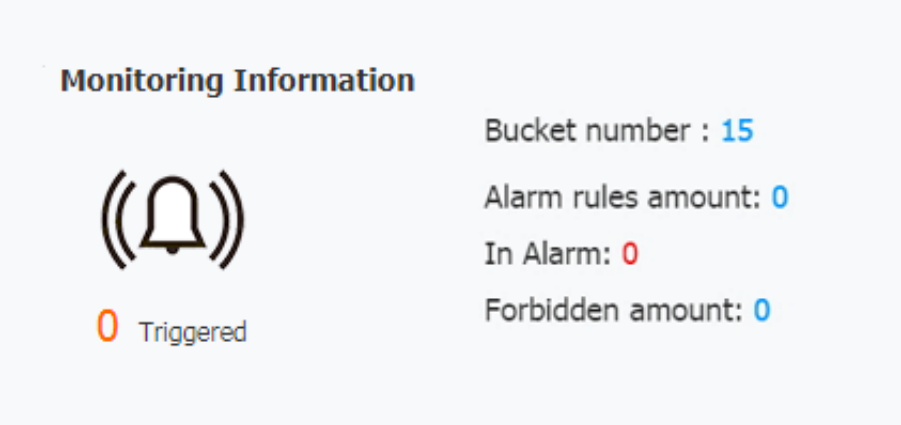


Users

The Users page displays monitoring information at the user level, including User monitoring information, Latest month statistics and User-level metric indicators.

- User monitoring information

  This module shows the total number of your buckets and related alarm rules.

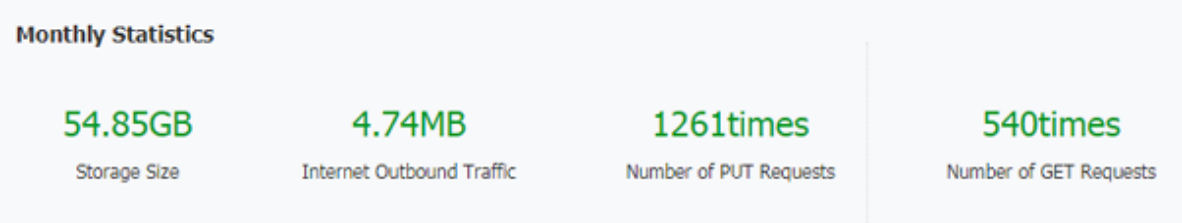


  - The parameters are as follows: Click the number next to Bucket number to display all the buckets you have created.
    - Click the number next to Alarm rules amount, In Alarm, Forbidden amount, or Alerted to display the following information: Alarm rules amount refers to total number of alarm rules you have set.
    - In Alarm refers to alarms in alarm state.
    - Forbidden amount refers to alarms that are currently disabled.
    - Alerted refers to alarms recently changed to alarm state

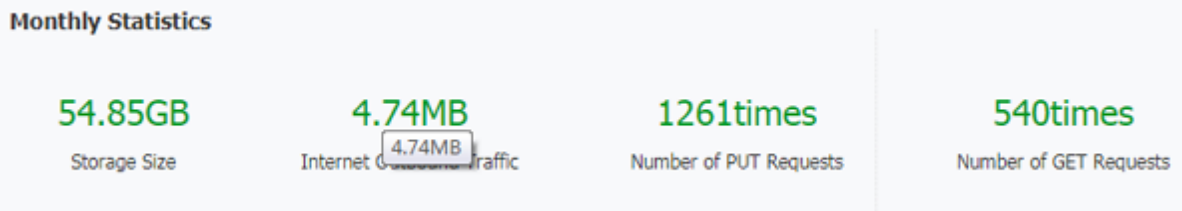
- Monthly Statistics

  This module displays information about charged OSS resources that you have used during the period from 00:00 on the first day of the current month, to the metering acquisition cutoff time. The following indicators are displayed:

  - Storage Size
  - Internet Outbound Traffic
  - Number of PUT Requests
  - Number of GET Requests

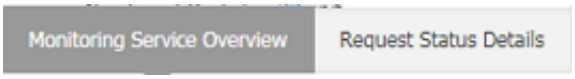


The unit of each value is automatically adjusted by the order of magnitude. The exact value is displayed when you hover the cursor over the selected value.



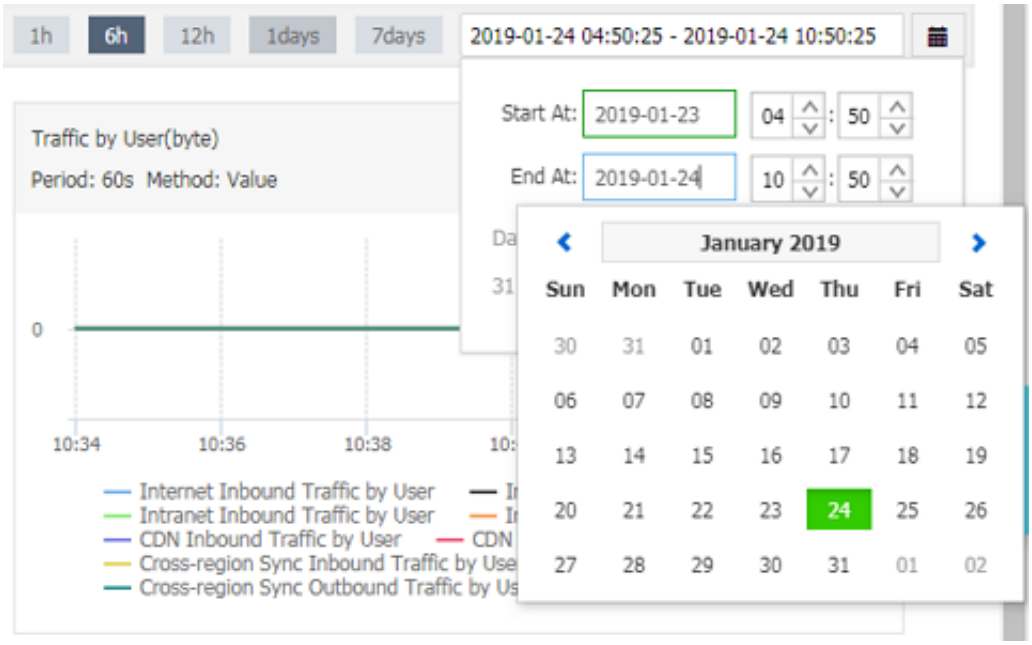
- User-level metric indicators

  This module displays user-level metric charts and tables and consists of `Monitoring Service Overview` and `Request Status Detalls .`



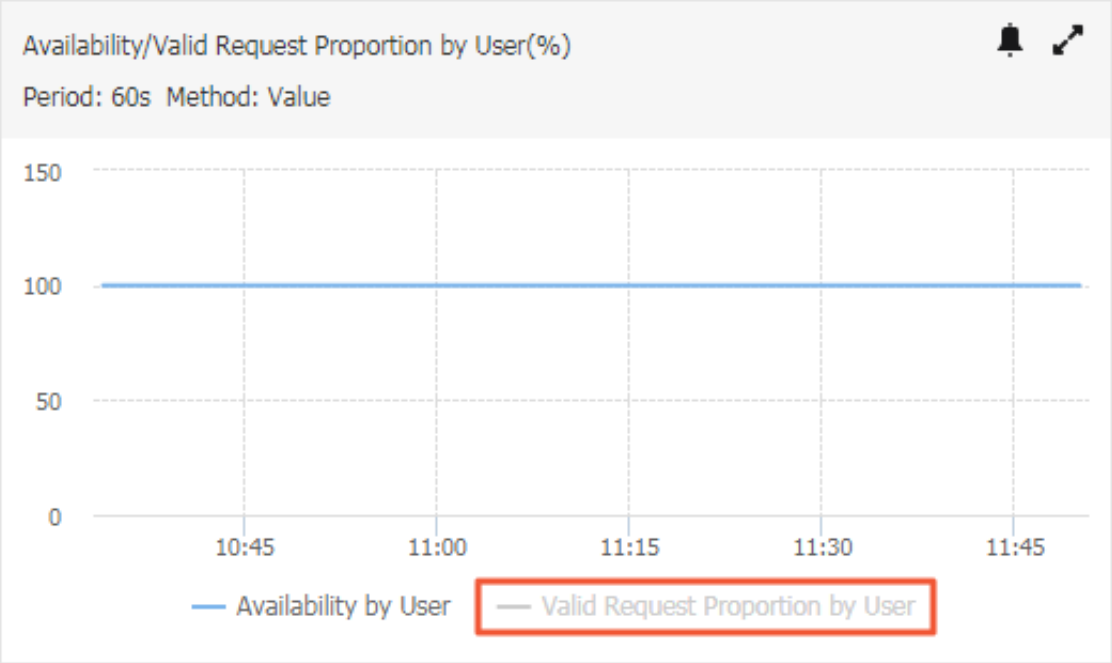
  You can select a pre-determined time range, or define a time range in the custom time boxes, to display the corresponding metric chart or table.

  - The following time range options are available: 1 hour, 6 hours, 12 hours, 1 day, and 7 days. The default option is 1 hour.
  - The custom time boxes allow the start time and the end time to be defined at the minute-level.



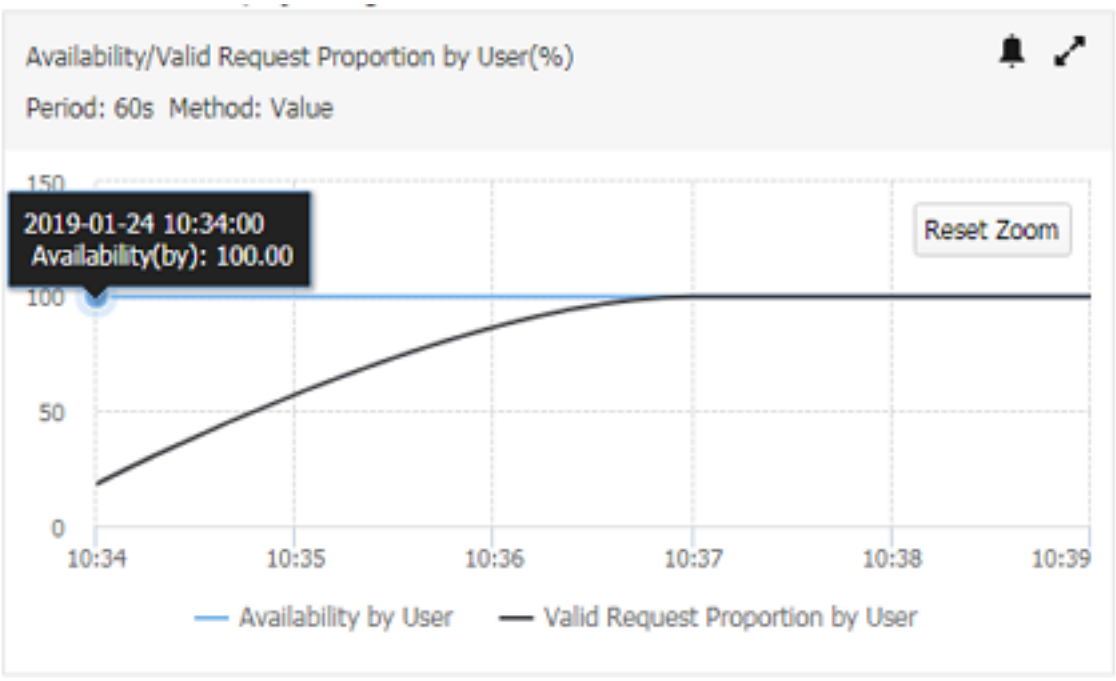
  Metric charts/tables support the following display modes:

  - Legend hiding: You can click a legend to hide the corresponding indicator curve, as shown in the following figure:



- Click the icon in the upper-right corner of a metric chart to zoom in on the chart. Be note that tables cannot be zoomed in.
- Click the icon in the upper-right corner of a metric chart to configure alarm rules for the displayed metric indicators. For more information, see the *Alarm Service User Guide*. Be note that you cannot set alarm rules for tables and metering reference indicators.
- Place the cursor inside the curve area of a chart, and press and hold the left button on the mouse while dragging the mouse to

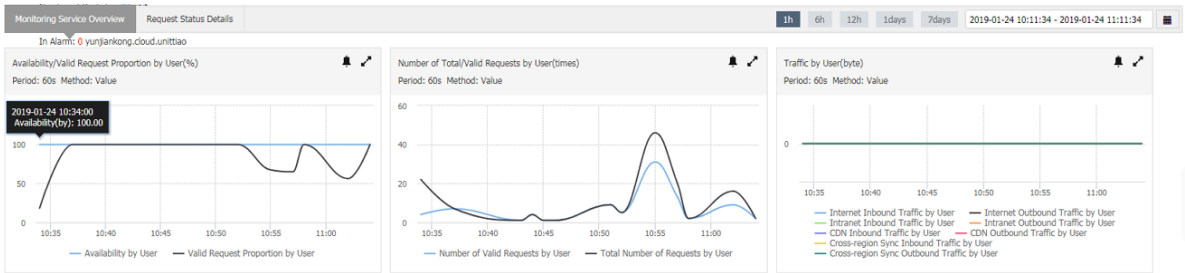
extend the time range. Click Reset Zoom to restore the original time range.



- Monitoring Service Overview

  The Monitoring Service Overview page displays the following main metric charts:

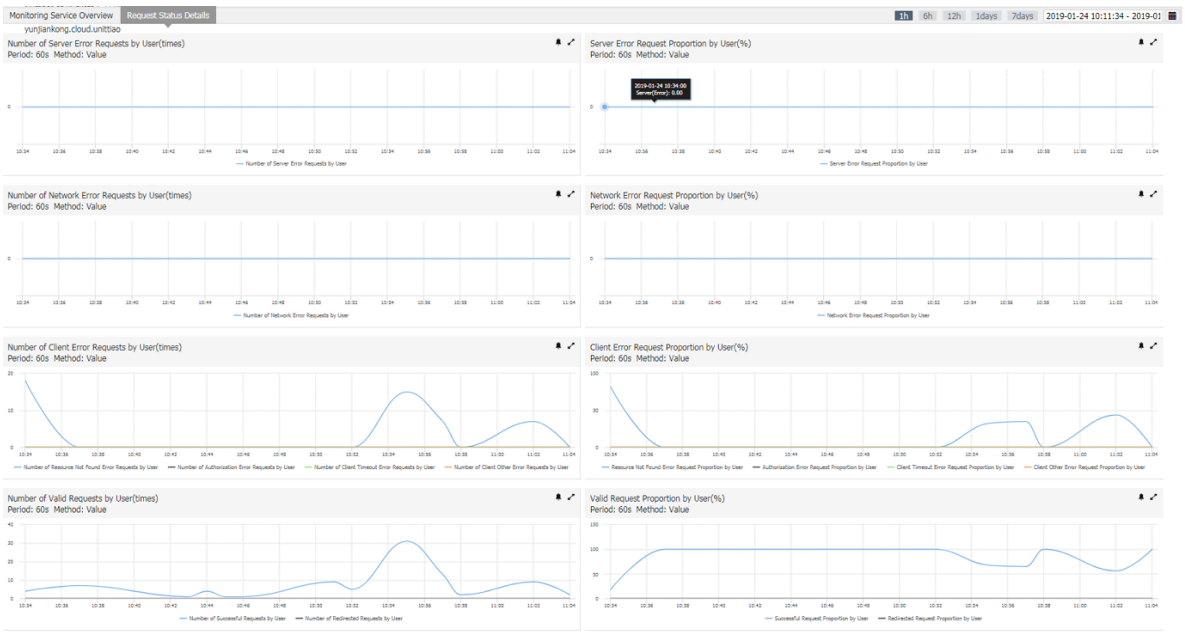
  - User-level availability/valid request rate, which includes two metric indicators: availability and percentage of valid requests.
  - User-level requests/valid requests, which includes two metric indicators: total number of requests and number of valid requests.
  - User-level traffic, which includes eight metric indicators: Internet outbound traffic, Internet inbound traffic, Intranet outbound traffic, Intranet inbound traffic, CDN outbound traffic, CDN inbound traffic, outbound traffic of cross-region replication, and inbound traffic of cross-region replication.
  - User-level request state distribution, which is a table that displays the number and percentage of each type of requests within the selected time range.



- Request Status Detalls

  The "Request State Details" page shows the metric data of request state distribution through the following main metric charts:

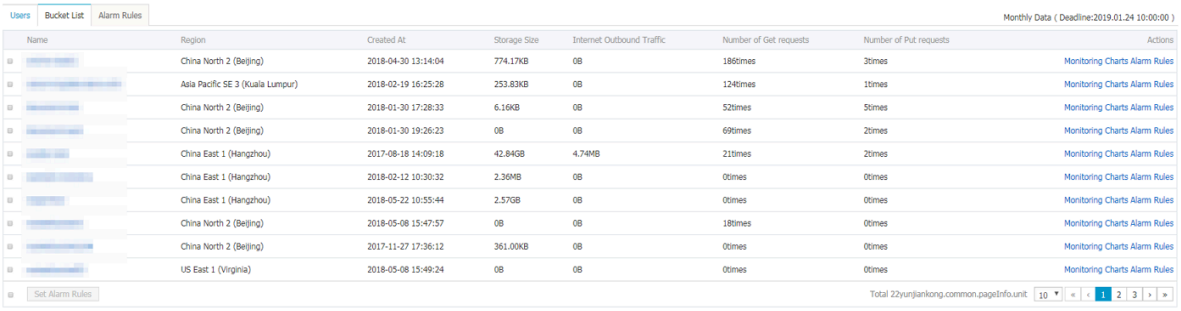
  - Number of Server Error Requests by User
  - Server Error Request Proportion by User
  - Number of Network Error Requests by User
  - Network Error Request Proportion by User
  - Number of Client Error Requests by User, which includes four metric indicators : number of error requests indicating resource not found, number of authorizat ion error requests, number of client-site time-out error requests, and number of other client-site error requests
  - Client Error Request Proportion by User, which includes four metric indicators : percentage of error requests indicating resource not found, percentage of authorization error requests, percentage of client-site time-out error requests, and percentage of other client-site error requests
  - Number of Valid Requests by User, which includes two metric indicators: number of successful requests and number of redirect requests
  - Valid Request Proportion by User, which includes two metric indicators: percentage of successful requests and percentage of redirect requests



Bucket List

- **Bucket list information**

  The Bucket list tab page displays the information including bucket name, region, creation time, metering statistics of the current month, and related operations.



  - Display parameters are as follows: The metering statistics of the current month display the storage size, Internet outbound traffic, Put request count, and Get request count for each bucket.
  - Click Monitoring chart or the corresponding bucket name to go to the bucket monitoring view page.
  - Click Alarm rules next to your expected bucket, or go to the Alarm rules tab to display all alarm rules of the bucket.
  - Enter the expected bucket name in the search box in the upper left-corner to display the bucket (fuzzy match is supported).
  - Select the check boxes before the expected bucket names and click Setting custom monitor alarm rules to batch set alarm rules. For more information, see the *Alarm Service User Guide*.

- Bucket-level monitoring view

  Click Monitoring chart next to the expected bucket name in the bucket list to go to the bucket monitoring view.

  The bucket monitoring view displays metric charts based on the following six indicator groups:

  - ■ Monitoring Service Overview
    ■ Request Status Details
    ■ Measurement Reference
    ■ Average Latency
    ■ Maximum Latency
    ■ Successful Request Category

  Except measurement reference, other indicators are displayed with an aggregation granularity of 60s. The default time range for bucket-level metric charts is of the previous six hours, whereas that for user-level metric charts is of the previous hour. Click Back to bucket list in the upper-left corner to return to the Bucket list tab.

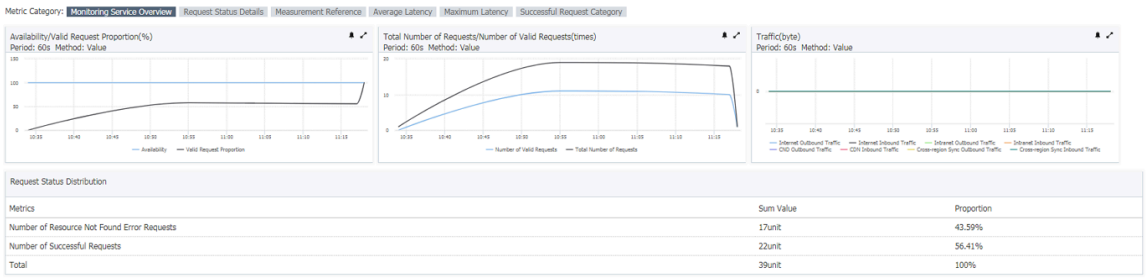
  - Monitoring Service Overview

    This indicator group is similar to the service monitoring overview at the user level, but the former displays metric data at the bucket level. The main metric charts include:

    ■ Request Valid Availability, which includes two metric indicators: availability and percentage of valid requests
    ■ Total/Valid request, which includes two metric indicators: total number of requests and number of valid requests
    ■ Overflow, which includes eight metric indicators: Internet outbound traffic, Internet inbound traffic, intranet outbound traffic, intranet inbound traffic,

CDN outbound traffic, CDN inbound traffic, outbound traffic of cross-region replication, and inbound traffic of cross-region replication

- Request status count, which is a table that displays the number and percentage of each type of requests within the selected time range.



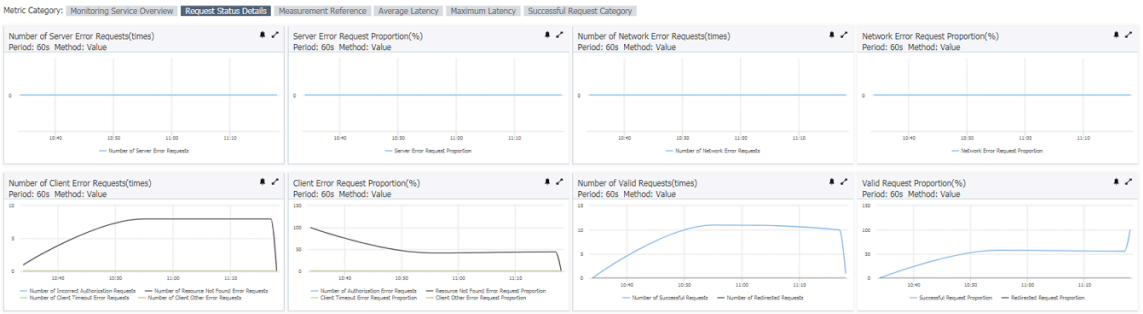
- Request Status Details

This indicator group is similar to the request state details at the user level, but the former displays metric data at the bucket level. The main metric charts include:

- Server error count
- Server error rate
- Network error count
- Network error request rate
- Client error request count, which includes four metric indicators: number of error requests indicating resource not found, number of authorization error requests, number of client-site time-out error requests, and number of other client-site error requests
- Client error request percent, which includes four metric indicators: percentage of error requests indicating resource not found, percentage of

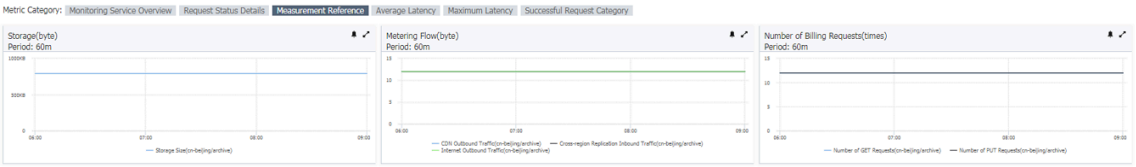
authorization error requests, percentage of client-site time-out error requests , and percentage of other client-site error requests

- Redirect request count, which includes two metric indicators: number of successful requests and number of redirect requests
- Success redirect rate, which includes two metric indicators: percentage of successful requests and percentage of redirect requests



- Measurement Reference

The metering reference group shows metering indicators with an hourly collection and representation granularity, as shown in the following figure:



The metering metric charts include:

- Quota size
- Overflow
- Billing requests, which includes the Get request count and Put request count.

After a bucket is created, new data is collected in the next hour on the hour following the current time point, and the collected data will be displayed within 30 minutes.

- Average Latency

This indicator group contains the average latency indicators of API monitoring. The metric charts include:

- getObject Average Latency
- headObject Average Latency
- putObject Average Latency

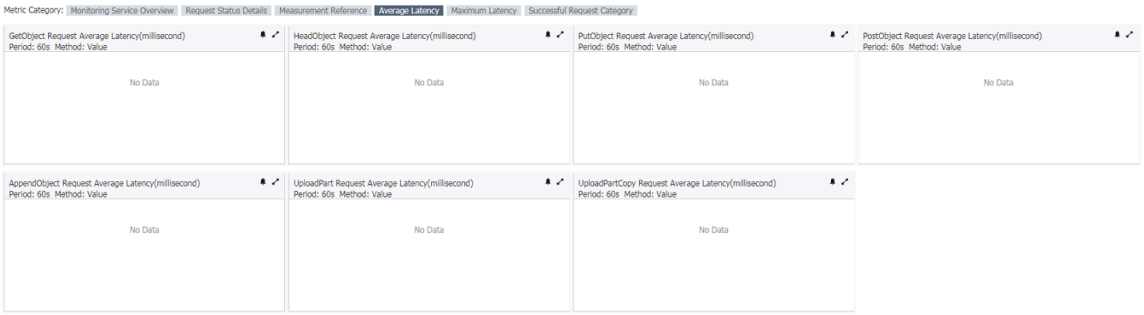
■ postObject Average Latency

■ append Object Average Latency

■ upload Part Average Latency

■ upload Part Copy Average Latency

Each metric chart shows the corresponding average E2E latency and average server latency. See the figure below:



- Maximum Latency

This indicator group contains the maximum latency indicators of API monitoring. The metric charts include:

■ getObject Max Latency(Millisecond)

■ headObject Max Latency

■ putObject Max Latency

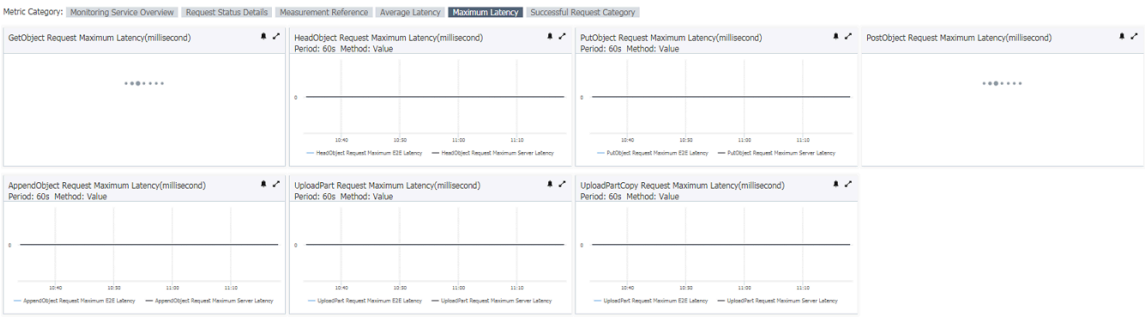
■ postObject Max Latency

■ append Object Max Latency

■ upload Part Max Latency

■ upload Part Copy Max Latency

Each metric chart shows the corresponding maximum E2E latency and maximum server latency. See the following figure:

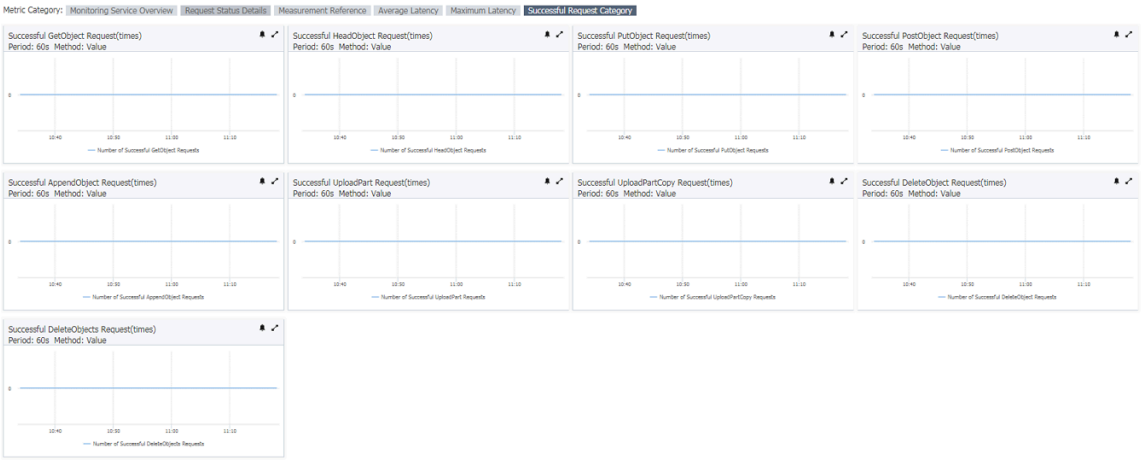


- Successful Request Category

This indicator group contains the successful request count indicators of API monitoring. The metric charts include:

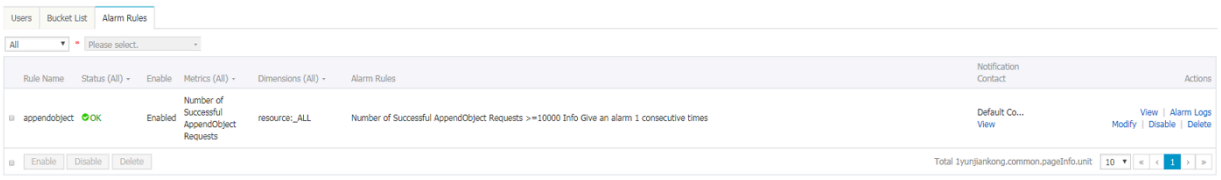
- getObject Success Count
- headObject Success Count
- putObject Success Count
- post Object Success Count
- append Object Success Count
- upload Part Success Count
- upload Part Copy Success Count
- delete Object Success Count
- deleteObjects Success Count

See the following figure:



Alarm Rules

The Alarm rules tab page allows you to view and manage all your alarm rules, as shown in the following figure:



For the description and usage of the "Alarm Rules" tab page, see the *Alarm Service User Guide*.

Additional links

For more information regarding the important points and user guide of the monitoring service, see the related chapter in *Monitoring, diagnosis, and troubleshooting*.

18.3 Alarm service user guide

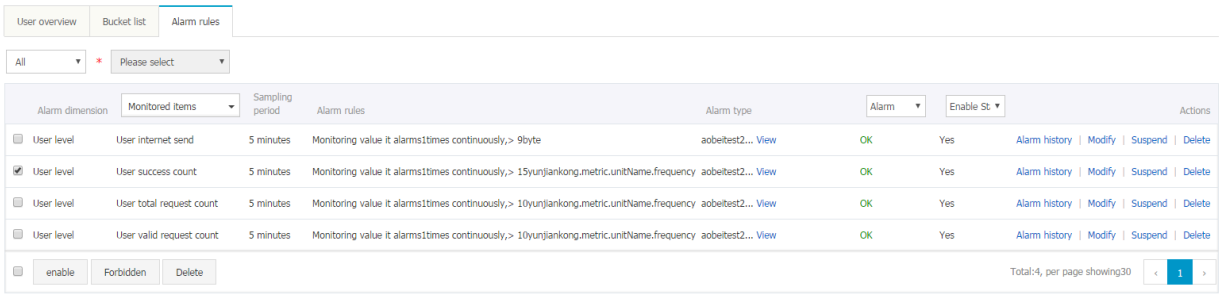
To help familiarize yourself with the basic concepts and configurations of alarm contacts and alarm contact groups, we recommend that the following documents are read before this user guide:

- *Alarm service overview*
- *Manage alarm contact*

Additionally, OSS alarm rules are developed in accordance with OSS metric items. This means they are categorized by dimensions similar to those of OSS metric items. Two alarm dimensions are available: user-level and bucket-level.

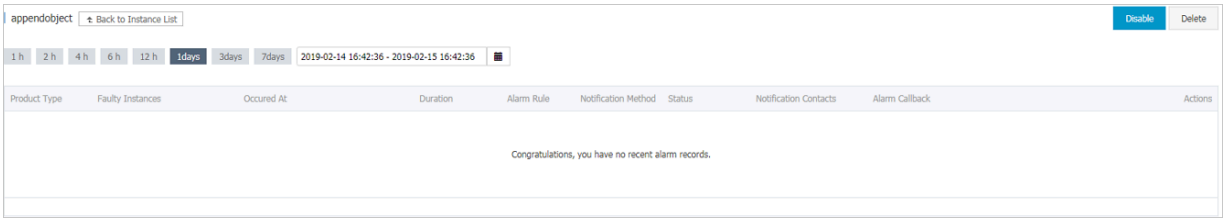
Alarm rule page

The alarm rule page is where you can view, modify, activate, deactivate, and delete alarm rules related to OSS monitoring alarms. You can also view historical alarms of the different alarm rules. An example screenshot is as follows:



- Click Modify next to the expected alarm rule to modify it.
- Click Delete next to the expected alarm rule to delete it. You can also select multiple alarm rules and then click Delete at the bottom of the table to delete alarm rules in batches.
- If an alarm rule is in the `Enable` status, click Suspend next to the expected alarm rule to deactivate it. Once the alarm rule is suspended, you no longer receive alarm information for this rule. You can also select multiple alarm rules and then click Forbidden at the bottom of the table to deactivate alarm rules in batches.

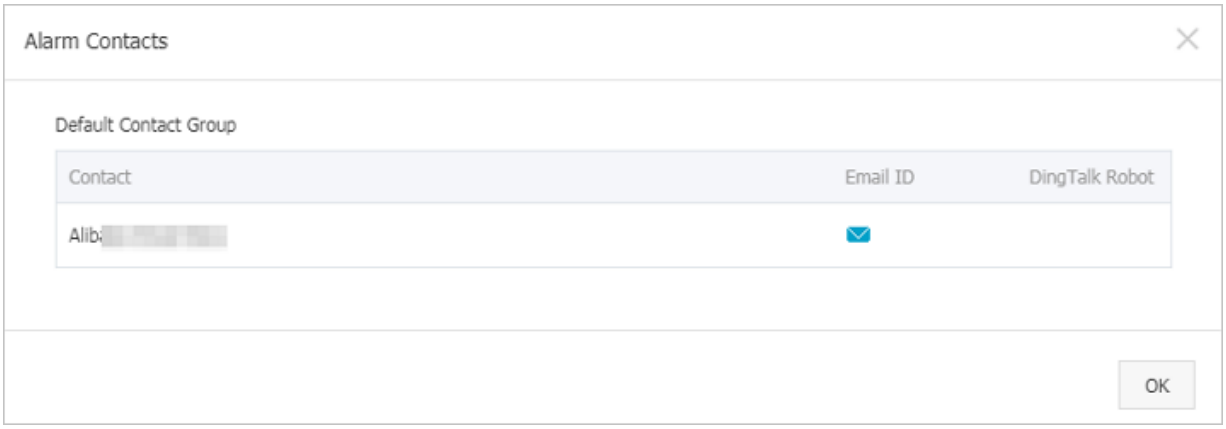
- If an alarm rule is in the `Forbidden` status, click Enable next to the expected alarm rule to activate it. The rule is then be resumed to detect exceptions and send alarm information. You can also select multiple alarm rules and then click Enable at the bottom of the table to activate alarm rules in batches.
- Click Alarm history next to the expected alarm rule to view information on past alarms corresponding to this rule.



Alarm history concepts

- Alarm history refers to past status changes of a selected alarm rule. Operations such as switching from normal status to alarm status, or switching from alarm status to normal status, are considered status changes. Additionally, a status change called channel silence is also available.
- Channel silence occurs when a triggered alarm has remained active for 24 hours and has not returned to a normal status. In this case, no new alarm notifications are sent for 24 hours.
- Historical alarm information is retained for one month, and can be queried at a maximum of three days' data at one time within this time period. Alarm information older than one month is automatically deleted, and cannot be queried.

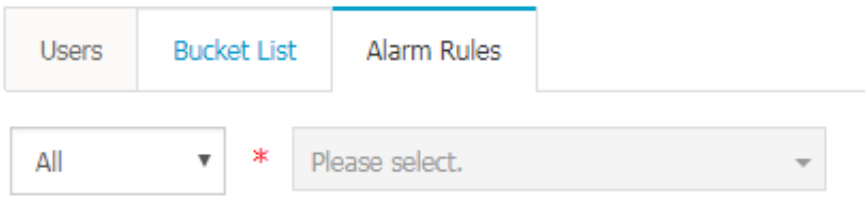
To view details about an alarm, such as the alarm contact list and contact details, click View next to the expected alarm. An example screenshot displaying specific details is as follows:



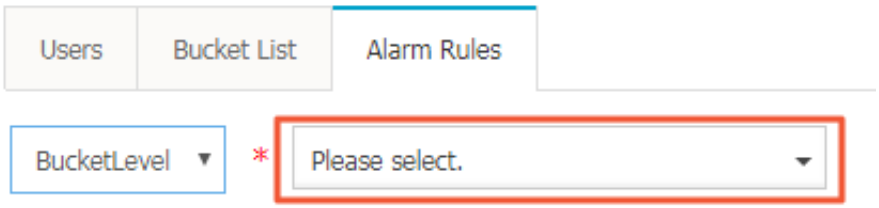
Search for alarm rules

Based on the control information at the bottom of the alarm rule page, you can quickly find alarm rules you have searched for:

- Alarm dimension drop-down box: All and Bucket Level. If you select `All`, all user-level and bucket-level alarm rules are displayed.



- Bucket drop-down box: If you select `Bucket Level` in the alarm dimension drop-down box, this box lists the buckets of the current user. Select a bucket to display all the alarm rules for this bucket:



- Monitored items drop-down box lists all OSS metric items, including user-level and bucket level metric items. If you select `Monitored items`, user-level and bucket-level alarm rules for all monitored items are displayed.
- Alarm status drop-down box lists alarm status, including OK and Alarm .
- Enable state drop-down box lists the enable status, including Enabled and Forbidden .
- View alarm rules

  Click the Alarm rules tab to display all alarm rules. You can also select `Bucket Level` in the drop-down box and then select the name of the expected bucket to see alarm rules for that bucket. You can then filter returned information using selections in the drop-down box such as `Metric Item`, `Alarm status` and `Activation status`.

- View alarm rules for a specific bucket

  If you want to view the alarm rules of a specific bucket, select Bucket Level in the alarm dimension drop-down box and then select the name of the target bucket in the bucket drop-down box. Select Alarm Rules for the target bucket in the

Bucket List to go to the alarm tab. This tab displays all the alarm rules for this bucket. With the `Metric item`, `Alarm status` and `Activation status` drop-down boxes, you can better filter the alarm rules that match certain conditions in the current dimension.

- View alarm rules related to a specific metric item

  Select a specific metric item in the metric item drop-down box to display all the alarm rules for this metric item.

- View alarm rules in a certain alarm status

  Choose an alarm status in the alarm status drop-down box, such as Alarm, to display all the alarm rules currently in this status.

- View alarm rules in a certain activation status

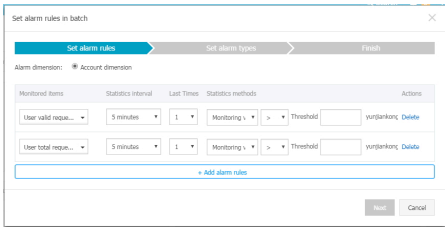
  Choose an activation status in the activation status drop-down box, such as Deactivated, to display all the alarm rules currently in this status.

Add alarm rules

After specifying a bucket in the Bucket List Tab, click Set Alarm Rule to set an alarm rule. Alternatively, click the alarm icon  in a metric chart in the User Overview tab or the Monitoring View tab of a specific bucket to open the Batch Set Alarm Rules page to set multiple alarm rules.

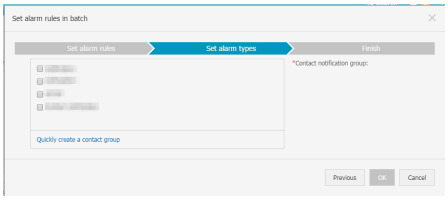
The following example describes how to set alarms at the user-level. To learn more about the terms and concepts used later, see the *Alarm service overview* of CloudMonitor.

1. Set parameters for Alarm rules as follows:



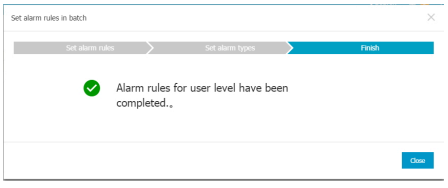
- Alarm dimension specifies the monitoring dimension of the alarm rule to set. If the dimension is set to bucket-level, the expected bucket with which to set the alarm rule for must be specified.
- Monitored items specifies all the metric items for the selected alarm dimension. You can use the quick search box to easily find metric items:
- Statistics interval specifies the length of the interval between statistical measurements. The default setting is 5 minutes.
- Last times specifies the number of statistical cycles for which an alarm which is triggered when the value of the metric item continuously exceeds the threshold value in several consecutive statistical cycles.
- Statistics method: specifies the statistical indicator calculated for this metric item. For the OSS monitoring service, the statistical method is set as `Monitoring Value .`
- Click + Add alarm rules to set additional metric item alarm rules.
- Click Delete next to the expected alarm rule to delete it.

2. Click Next, the page to Set the alarm types is then displayed.



If you have set alarm contract groups following the *Manage alarm contact*, they are displayed on the interface. If you have not set alarm contact groups, click Quickly create a contact group and follow the prompts to create a group.

3. Click OK.



- Add alarm rules in the Bucket list

  Under the Bucket list tab, you can add identical alarm rules for multiple buckets at the same time. Select the expected buckets for which to configure alarm rules and click Set Custom monitor alarm rules to go to the alarm rule settings page previously described in Add alarm rules.

  Note:
  During batch setting, the alarm dimension is bucket-level and the metric item must be a bucket-level metric item.

- Add alarm rules in a metric chart

  In the User overview or Monitoring chart tab, for the expected bucket, click 

  in the upper-right corner of a metric chart to set alarm rules for the metric item associated with this metric chart.

  Note:
  If you click the alarm icon in a metric chart, the alarm dimension displayed on the alarms rule page is pre-determined and you can only set alarm rules for the metric item corresponding to the metric chart.

Considerations

Currently, alarm rules can be created without requiring prior association to a bucket . If you delete a bucket, any associated alarm rules are not deleted. Before deleting a bucket, we recommend that you delete any corresponding alarm rules first.

18.4 Metric item reference

This chapter provides parameter references to use with the API, or the CloudMonitor SDK, to access the metric data of the OSS monitoring service.

Project

The OSS monitoring service metric data uses the same project name: acs_oss.

Sample code written by the Java SDK:

```
QueryMetri  cRequest   request  =  new   QueryMetri  cRequest ();
Request .  setproject  (" acs_oss  ");
```

StartTime and EndTime

The value range of the time parameters for CloudMonitor is in the format of [StartTime, EndTime]. The data that is attributed to StartTime is not collected, whereas the data that is attributed to EndTime can be accessed.

The CloudMonitor retention policy specifies that data is retained for 31 days. This means the interval between StartTime and EndTime cannot exceed 31 days, and data outside the 31 day collection period cannot be accessed.

For more information about other time parameters, see *CloudMonitor API Reference*.

Sample code written by the Java SDK:

```
request . setStartTi  me (" 2016 - 05 - 15   08 : 00 : 00 ");
request . setEndTime (" 2015 - 05 - 15   09 : 00 : 00 ");
```

Dimensions

OSS metric items are classified into user level bucket level based on application scenarios. The value of Dimensions varies with regards to access of metric data at these different levels.

- Dimensions does not need to be set for access to user-level metric data.
- Set Dimensions access to bucket-level metric data as follows:

```
{" BucketName ": " your_bucke  t_name "}
```

your_bucket_name indicates the name of the bucket you want to access.

Note: Dimensions is a JSON string and has only one Key-Value pair for OSS metric indicators.

Sample code written by the Java SDK:

```
request . setDimensi  ons ("{\" BucketName \":\" your_bucke  t_name
\"}");
```

Period

The aggregation granularity of all OSS metric indicators, except metering indicators , is 60s by default. The aggregation granularity of metering indicators is 3,600s by default.

Sample code written by the Java SDK:

```
request . setPeriod (" 60 ");
```

Metric

The *Monitoring indicators reference* describes the following metric items.

Metric	Metric item name	Unit	Level
Useravailability	User-level availability	%	User level
UserRequestValidRate	User-level valid request rate	%	User level
UserTotalRequestCount	User-level requests	Times	User level
UserValidRequestCount	User-level valid requests	Times	User level
UserInternetSend	User-level Internet outbound traffic	Byte	User level
UserInternetRecv	User-level Internet inbound traffic	Byte	User level
UserIntranetSend	User-level intranet outbound traffic	Byte	User level
UserIntranetRecv	User-level intranet inbound traffic	Byte	User level
UserCdnSend	User-level CDN outbound traffic	Byte	User level
UserCdnRecv	User-level CDN inbound traffic	Byte	User level
UserSyncSend	User-level outbound traffic of cross-region replication	Byte	User level
UserSyncRecv	User-level inbound traffic of cross-region replication	Byte	User level
UserServerErrorCount	User-level server-site error requests	Times	User level

Metric	Metric item name	Unit	Level
UserServer ErrorRate	User-level server-site error request rate	%	User level
UserNetwor kErrorCount	User-level network-site error requests	Times	User level
UserNetwor kErrorRate	User-level network -site error request rate	%	User level
UserAuthor izationErrorCount	User-level client-site authorization error requests	Times	User level
UserAuthor izationErrorRate	User-level client-site authorization error request rate	%	User level
UserResour ceNotFound ErrorCount	User-level client-site error requests indicating resource not found	Times	User level
UserResour ceNotFound ErrorRate	User-level client-site error request rate indicating resource not found	%	User level
UserClient TimeoutErrorCount	User-level client-site time-out error request	Times	User level
UserClient OtherErrorRate	User-level client-site time-out error request rate	%	User level
UserClient OtherErrorCount	Other user-level client-site error requests	Times	User level
UserClient OtherErrorRate	Other user-level client-site error request rate	%	User level
UserSuccessCount	Successful user-level requests	Times	User level

Metric	Metric item name	Unit	Level
UserSuccessRate	Successful user-level request rate	%	User level
UserRedirectCount	User-level redirect requests	Times	User level
UserRedirectRate	User-level redirect request rate	%	User level
Availability	Availability	%	Bucket level
RequestValidRate	Valid request rate	%	Bucket level
TotalRequestCount	Requests	Times	Bucket level
ValidRequestCount	Valid requests	Times	Bucket level
InternetSend	Internet outbound traffic	Byte	Bucket level
InternetRecv	Internet inbound traffic	Byte	Bucket level
IntranetSend	Intranet outbound traffic	Byte	Bucket level
IntranetRecv	Intranet inbound traffic	Byte	Bucket level
CdnSend	CDN outbound traffic	Byte	Bucket level
CdnRecv	CDN inbound traffic	Byte	Bucket level
SyncSend	Outbound traffic of cross-region replication	Byte	Bucket level
SyncRecv	Inbound traffic of cross-region replication	Byte	Bucket level
ServerErrorCount	Server-site error requests	Times	Bucket level
ServerErrorRate	Server-site error request rate	%	Bucket level
NetworkErrorCount	Network-site error requests	Times	Bucket level
NetworkErrorRate	Network-site error request rate	%	Bucket level

Metric	Metric item name	Unit	Level
AuthorizationErrorCount	Client-site authorization error requests	Times	Bucket level
AuthorizationErrorRate	Client-site authorization error request rate	%	Bucket level
ResourceNotFoundErrorCount	Client-site error requests indicating resource not found	Times	Bucket level
ResourceNotFoundErrorRate	Client-site error request rate indicating resource not found	%	Bucket level
ClientTimeoutErrorCount	Client-site time-out error requests	Times	Bucket level
ClientOtherErrorRate	Client-site time-out error request rate	%	Bucket level
ClientOtherErrorCount	Other client-site error requests	Times	Bucket level
ClientOtherErrorRate	Other client-site error request rate	%	Bucket level
SuccessCount	Successful requests	Times	Bucket level
SuccessRate	Successful request rate	%	Bucket level
RedirectCount	Redirect requests	Times	Bucket level
RedirectRate	Redirect request rate	%	Bucket level
GetObjectE2eLatency	Average E2E latency of GetObject requests	Millisecond	Bucket level
GetObjectServerLatency	Average server latency of GetObject requests	Millisecond	Bucket level
MaxGetObjectE2eLatency	Maximum E2E latency of GetObject requests	Millisecond	Bucket level

Metric	Metric item name	Unit	Level
MaxGetObjectServerLatency	Maximum server latency of GetObject requests	Millisecond	Bucket level
HeadObjectE2eLatency	Average E2E latency of HeadObject requests	Millisecond	Bucket level
HeadObjectServerLatency	Average server latency of HeadObject requests	Millisecond	Bucket level
MaxHeadObjectE2eLatency	Maximum E2E latency of HeadObject requests	Millisecond	Bucket level
MaxHeadObjectServerLatency	Maximum server latency of HeadObject requests	Millisecond	Bucket level
PutObjectE2eLatency	Average E2E latency of PutObject requests	Millisecond	Bucket level
PutObjectServerLatency	Average server latency of PutObject requests	Millisecond	Bucket level
MaxPutObjectE2eLatency	Maximum E2E latency of PutObject requests	Millisecond	Bucket level
MaxPutObjectServerLatency	Maximum server latency of PutObject requests	Millisecond	Bucket level
PostObjectE2eLatency	Average E2E latency of PostObject requests	Millisecond	Bucket level
PostObjectServerLatency	Average server latency of PostObject requests	Millisecond	Bucket level

Metric	Metric item name	Unit	Level
MaxPostObjectE2eLatency	Maximum E2E latency of PostObject requests	Millisecond	Bucket level
MaxPostObjectServerLatency	Maximum server latency of PostObject requests	Millisecond	Bucket level
AppendObjectE2eLatency	Average E2E latency of AppendObject requests	Millisecond	Bucket level
AppendObjectServerLatency	Average server latency of AppendObject requests	Millisecond	Bucket level
MaxAppendObjectE2eLatency	Maximum E2E latency of AppendObject requests	Millisecond	Bucket level
MaxAppendObjectServerLatency	Maximum server latency of AppendObject requests	Millisecond	Bucket level
UploadPartE2eLatency	Average E2E latency of UploadPart requests	Millisecond	Bucket level
UploadPartServerLatency	Average server latency of UploadPart requests	Millisecond	Bucket level
MaxUploadPartE2eLatency	Maximum E2E latency of UploadPart requests	Millisecond	Bucket level
MaxUploadPartServerLatency	Maximum server latency of UploadPart requests	Millisecond	Bucket level

Metric	Metric item name	Unit	Level
UploadPartCopyE2eLatency	Average E2E latency of UploadPartCopy requests	Millisecond	Bucket level
UploadPartCopyServerLatency	Average server latency of UploadPartCopy requests	Millisecond	Bucket level
MaxUploadPartCopyE2eLatency	Maximum E2E latency of UploadPartCopy requests	Millisecond	Bucket level
MaxUploadPartCopyServerLatency	Maximum server latency of UploadPartCopy requests	Millisecond	Bucket level
GetObjectCount	Successful GetObject requests	Times	Bucket level
HeadObjectCount	Successful HeadObject requests	Times	Bucket level
PutObjectCount	Successful PutObject requests	Times	Bucket level
PostObjectCount	Successful PostObject requests	Times	Bucket level
AppendObjectCount	Successful AppendObject requests	Times	Bucket level
UploadPartCount	Successful UploadPart requests	Times	Bucket level
UploadPartCopyCount	Successful UploadPartCopy requests	Times	Bucket level
DeleteObjectCount	Successful DeleteObject requests	Times	Bucket level

Metric	Metric item name	Unit	Level
DeleteObjectsCount	Successful DeleteObjects requests	Times	Bucket level

The following table lists the metric items of metering indicators with an aggregation granularity of 3,600s.

Metric	Metric item name	Unit	Level
MeteringStorageUtilization	Size of storage	Byte	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.
MeteringGetRequest	Get requests	Times	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.
MeteringPutRequest	Put requests	Times	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.
Meteringinternettx	Volume of Internet outbound traffic	Byte	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.

Metric	Metric item name	Unit	Level
MeteringCdnTX	Volume of CDN outbound traffic	Byte	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.
MeteringSyncRX	Volume of inbound traffic of cross-region replication	Byte	If Dimensions is set, the returned metric data belongs to the bucket level; if Dimensions is not set, the returned metric data belongs to the user level.

Sample code written by the Java SDK:

```
request . setMetric (" UserAvaila  bility ");
```

18.5 Monitoring indicators reference

OSS indicators can be monitored at the user level or the bucket level based on application scenarios.

In addition to common chronological metric indicators, the system analyzes and collects statistics on the existing metric indicators for easy user observation of metric data and matching of billing policy. Statistical indicators over a specified period of time are provided, such as request status distribution and metering statistics of the month. This reference guide describes the indicators in detail.

All indicators are collected at the minute-level (per minute) except for metering and statistical indicators. Metering indicators are collected at the hour-level (per hour).

User-level indicators

The user level indicator refers to the indicator information that monitors the overall situation of the OSS system used from the user's account level, and is a summary of all bucket related monitoring data under the account. User-level indicators consist

of three monitoring indicator details: current-month metering statistics, service monitoring overview, and request state details.

Service monitoring overview

Indicators in service monitoring overview are basic service indicators. Details of service monitoring overview indicators are as follows:

Indicator	Unit	Description
Availability	%	An indicator showing the system availability of using the storage service. It is obtained through the equation: `1 - percentage of requests with server - end errors ( indicated by a return code 5xx ) in all requests` .
Valid requests rate	%	Percentage of valid requests in all requests . For more information about valid requests, see the following description.
Requests	Times	Total number of requests received and processed by the OSS server
Valid requests	Times	Total number of requests whose return code is 2xx or 3xx.
Internet outbound traffic	Byte	Downstream Internet traffic
Internet inbound traffic	Byte	Upstream Internet traffic
Intranet outbound traffic	Byte	Downstream intranet traffic of the service system
Intranet inbound traffic	Byte	Upstream intranet traffic of the service system

Indicator	Unit	Description
CDN outbound traffic	Byte	Downstream CDN traffic when CDN acceleration service is activated, that is , the origin retrieval traffic
CDN inbound traffic	Byte	Upstream CDN traffic when CDN acceleration service is activated
Outbound traffic of cross-region replication	Byte	Downstream traffic generated in the data replication process when the cross-region replicatio n function is activated
Inbound traffic of cross-region replication	Byte	Upstream traffic generated in the data replication process when the cross-region replication function is activated

In addition to the above specific monitoring indicators, statistics are also provided for the distribution of request status over a period of time. The statistics are mainly based on the status codes of the returned status codes or OSS error codes (the total number and the percentage of requests within the observed time period).

Request state details

Request state details indicators are requested monitoring information based on the return status code, or OSS error code, associated with the different requests. Details of request state details indicators are as follows:

Indicator	Unit	Description
Server-site error requests	Times	Total number of requests with system-level errors indicated by a return code 5xx
Server-site error requests rate	%	Percentage of requests with server-end errors in all requests

Indicator	Unit	Description
Network error requests	Times	Total number of requests whose *HTTP status code is 499*
Network error requests rate	%	Percentage of requests with network errors in all requests
Client-end authorization error requests	Times	Total number of requests with a return code 403
Client-end authorization error requests rate	%	Percentage of requests with client-end authorization errors in all requests
Client-end error requests indicating resource not found	Times	Total number of requests with a return code 404
Client-end error requests rate indicating resource not found	%	Percentage of requests with client-end errors indicating resource not found in all requests
Client-end time-out error requests	Times	Total number of requests whose return status code is 408 or return OSS error code is RequestTimeout
Client-end time-out error requests rate	%	Percentage of requests with client-end time-out errors in all requests
Other client-end error requests	Times	Total number of requests with other client-end errors indicated by a return code 4xx
Other client-end error requests rate	%	Percentage of requests with other client-end errors in all requests
Successful requests	Times	Total number of requests whose return code is 2xx.
Successful requests rate	%	Percentage of successful requests in all requests

Indicator	Unit	Description
Redirect requests	Times	Total number of requests whose return code is 3xx.
Redirect requests rate	%	Percentage of redirect requests in all requests

Current-month metering statistics

Metering statistics of the current month are collected from 00:00 on the first day of the month to the metering cutoff time as indicated in the same month.

Details of the metering indicators currently available are as follows:

Indicator	Unit	Description
Storage size	Byte	Size of the total storage occupied by all buckets of a specified user before the metering statistic collection deadline.
Internet outbound traffic	Byte	Total Internet outbound traffic of the user from 00:00 of the first day of the current month to the metering statistic collection deadline.
Put requests	Times	Total number of Put requests of the user from 00:00 of the first day of the current month to the metering statistic collection deadline.
Get requests	Times	Total number of Get requests of the user from 00:00 of the first day of the current month to the metering statistic collection deadline.

Bucket-level indicators

Bucket-level indicators are used to monitor OSS operations of specific buckets and are applicable for business scenarios. In addition to current-month metering statistics

and basic service indicator items such as service monitoring overview and request state details (which can be monitored at the account level), bucket-level indicators include metering indicators and performance indicators such as metering reference, latency, and successful request operation categories.

Service monitoring overview

Similar to the user-level description, the service monitoring overview indicators are basic indicators, but use metric data that is displayed at the bucket-level.

Request state details

Similar to the user-level description, the request state details indicators use metric data that is displayed at the bucket-level.

Current-month metering statistics

Statistical methods are similar to those listed in current-month metering statistics at the user level, but the former collects resource usage statistics at the bucket level. Details of current-month metering statistics at the bucket-level are as follows:

Indicator	Unit	Description
Storage size	Byte	Size of storage occupied by a specified bucket before the metering statistic collection deadline.
Internet outbound traffic	Byte	Total Internet outbound traffic of a specified bucket from 00:00 of the first day of the current month to the metering statistic collection deadline.
Put requests	Times	Total number of Put requests of a specified bucket from 00:00 of the first day of the current month to the metering statistic collection deadline.

Indicator	Unit	Description
Get requests	Times	Total number of Get requests of a specified bucket from 00:00 of the first day of the current month to the metering statistic collection deadline.

Metering indicators

Metering indicators are monitored chronologically, and are collected and aggregated at the hour-level. Details of metering indicators are as follows:

Indicator	Unit	Description
Storage size	Byte	Average size of storage used by a specified bucket in an hour.
Internet outbound traffic	Byte	Total Internet outbound traffic of a specified bucket in an hour.
Put requests	Times	Total number of Put requests of a specified bucket in an hour.
Get requests	Times	Total number of Gut requests of a specified bucket in an hour.

Latency

Latency Request latency directly reflects the system performance and is monitored using two indicators: average latency and maximum latency. The indicators are collected and aggregated at the minute-level. Moreover, indicators can be classified based on the OSS API request operation type to more specifically reflect the performance of the system responding to different operations. Only APIs involving data operations in bucket-related operations (excluding meta operations) are monitored currently.

Besides, in order to facilitate analyzing performance hotspots and environmental problems, latency monitoring indicators are collected from two different links of E2E and the server, in which:

- E2E latency refers to the E2E latency of sending a successful request to OSS, including the processing time OSS requires to read the request, send a response, and receive a response confirmation message.
- Server latency is the latency of OSS processing a successful request, excluding the network delay involved in E2E latency.

Note that performance indicators are used to monitor successful requests (with a return status code 2xx).

The following table lists specific metric indicator items:

Indicator	Unit	Description
Average E2E latency of GetObject requests	Millisecond	Average E2E latency of successful requests whose request API is GetObject
Average server latency of GetObject requests	Millisecond	Average server latency of successful requests whose request API is GetObject
Maximum E2E latency of GetObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is GetObject
Maximum server latency of GetObject requests	Millisecond	Maximum server latency of successful requests whose request API is GetObject
Average E2E latency of HeadObject requests	Millisecond	Average E2E latency of successful requests whose request API is HeadObject
Average server latency of HeadObject requests	Millisecond	Average server latency of successful requests whose request API is HeadObject
Maximum E2E latency of HeadObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is HeadObject
Maximum server latency of HeadObject requests	Millisecond	Maximum server latency of successful requests whose request API is HeadObject

Indicator	Unit	Description
Average E2E latency of PutObject requests	Millisecond	Average E2E latency of successful requests whose request API is PutObject
Average server latency of PutObject requests	Millisecond	Average server latency of successful requests whose request API is PutObject
Maximum E2E latency of PutObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is PutObject
Maximum server latency of PutObject requests	Millisecond	Maximum server latency of successful requests whose request API is PutObject
Average E2E latency of PostObject requests	Millisecond	Average E2E latency of successful requests whose request API is PostObject
Average server latency of PostObject requests	Millisecond	Average server latency of successful requests whose request API is PostObject
Maximum E2E latency of PostObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is PostObject
Maximum server latency of PostObject requests	Millisecond	Maximum server latency of successful requests whose request API is PostObject
Average E2E latency of AppendObject requests	Millisecond	Average E2E latency of successful requests whose request API is AppendObject
Average server latency of AppendObject requests	Millisecond	Average server latency of successful requests whose request API is AppendObject
Maximum E2E latency of AppendObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is AppendObject

Indicator	Unit	Description
Maximum server latency of AppendObject requests	Millisecond	Maximum server latency of successful requests whose request API is AppendObject
Average E2E latency of UploadPart requests	Millisecond	Average E2E latency of successful requests whose request API is UploadPart
Average server latency of UploadPart requests	Millisecond	Average server latency of successful requests whose request API is UploadPart
Maximum E2E latency of UploadPart requests	Millisecond	Maximum E2E latency of successful requests whose request API is UploadPart
Maximum server latency of UploadPart requests	Millisecond	Maximum server latency of successful requests whose request API is UploadPart
Average E2E latency of UploadPartCopy requests	Millisecond	Average E2E latency of successful requests whose request API is UploadPart Copy
Average server latency of UploadPartCopy requests	Millisecond	Average server latency of successful requests whose request API is UploadPart Copy
Maximum E2E latency of UploadPartCopy requests	Millisecond	Maximum E2E latency of successful requests whose request API is UploadPart Copy
Maximum server latency of UploadPartCopy requests	Millisecond	Maximum server latency of successful requests whose request API is UploadPartCopy

Successful request operation categories

In conjunction with latency monitoring, the monitoring of successful requests reflects the system capability of processing access requests to a certain extent.

Similarly, only APIs involving data operations in bucket-related operations are monitored currently. The following lists specific indicator items:

Indicator	Unit	Description
Successful GetObject requests	Times	Number of successful requests whose request API is GetObject
Successful HeadObject requests	Times	Number of successful requests whose request API is HeadObject
Successful PutObject requests	Times	Number of successful requests whose request API is PutObject
Successful PostObject requests	Times	Number of successful requests whose request API is PostObject
Successful AppendObject requests	Times	Number of successful requests whose request API is AppendObject
Successful UploadPart requests	Times	Number of successful requests whose request API is UploadPart
Successful UploadPart Copy requests	Times	Number of successful requests whose request API is UploadPartCopy
Successful DeleteObject requests	Times	Number of successful requests whose request API is DeleteObject
Successful DeleteObjects requests	Times	Number of successful requests whose request API is DeleteObjects

18.6 Service monitoring, diagnosis, and troubleshooting

Despite reducing users' costs of infrastructure construction and O&M cloud applications compared to traditional applications, cloud applications have complicated monitoring, diagnosis, and troubleshooting. The OSS storage service provides a wide array of monitoring and log information, helping you fully understand program behavior and promptly discover and locate problems.

Overview

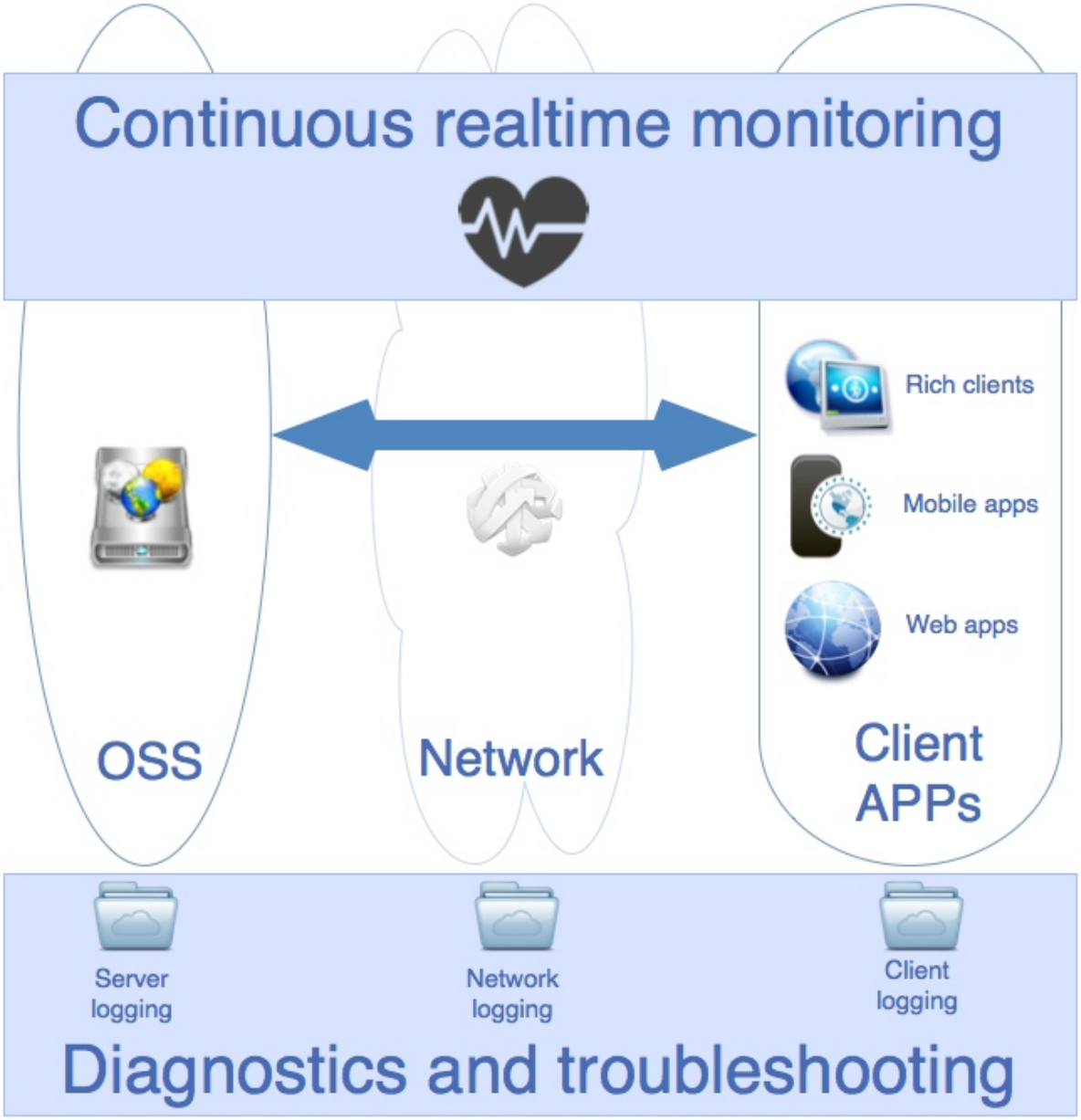
This chapter instructs you how to monitor, diagnose, and troubleshoot OSS problems by using the OSS monitoring service, logging, and other third-party tools, helping you achieve the following goals:

- Monitors in real time the running status and performance of OSS and provides prompt alarm notifications.
- Provides effective methods and tools to help you locate problems.
- Provides methods to help you quickly solve common OSS-related problems.

This chapter is organized as follows:

- *OSS real-time monitoring*: Describes how to use the OSS monitoring service to continuously monitor the running status and performance of OSS.
- *Tracking and diagnosis*: Describes how to use the OSS monitoring service and logging function to diagnose problems, and how to associate the relevant information in log files for tracking and diagnosis.
- *Troubleshooting*: Describes typical problems and corresponding troubleshooting methods.

.

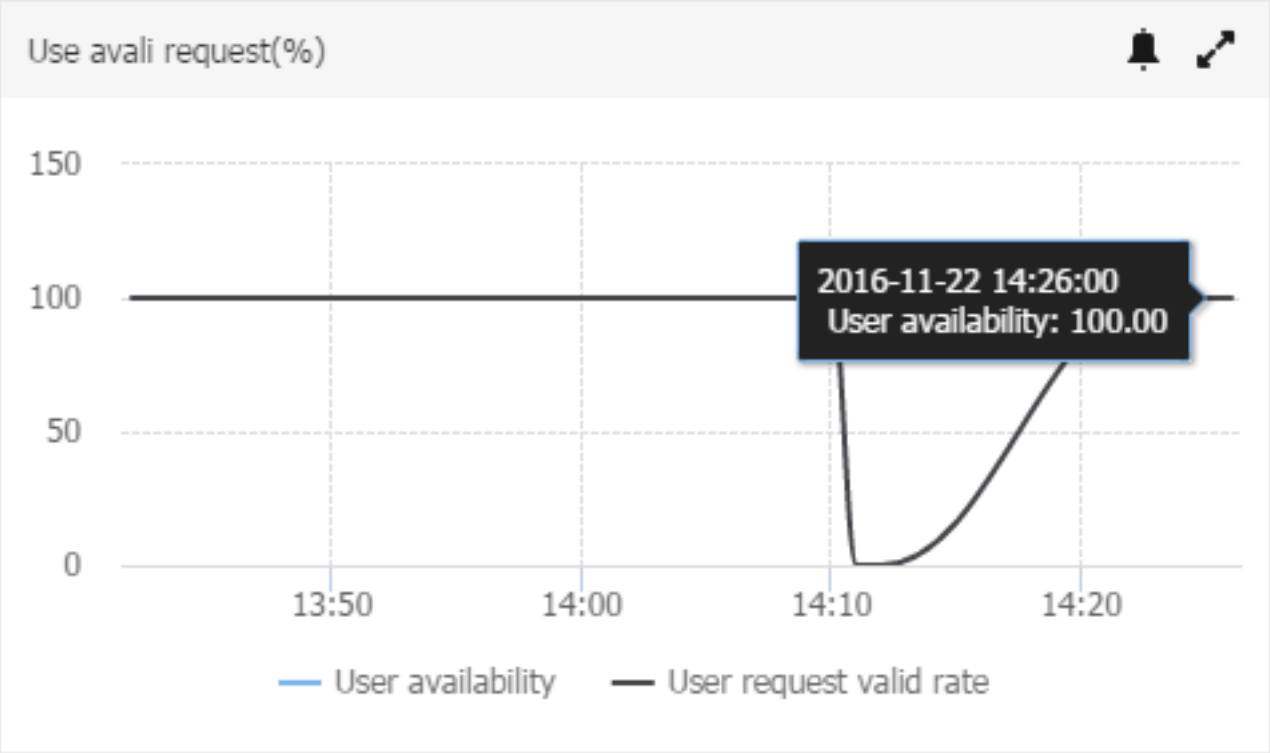


OSS monitoring

Overall operating conditions

- Availability and percentage of valid requests

  This is an important indicator related to system stability and the ability of users to correctly use the system. Any value lower than 100% indicates that some requests have failed.



Availability may also temporarily fall below 100% due to system optimization factors, such as partition migration for load balancing. In these cases, OSS SDKs can provide relevant retry mechanisms to handle this type of intermittent failure, keeping the service end unware.

Also, when the percentage of valid requests falls below 100%, you must analyze the issue based on your own usage. You can use request distribution statistics or request status details to determine the actual types of request errors. Then, you can use *Tracking and Diagnosis* to determine the cause and perform *Troubleshooting*.

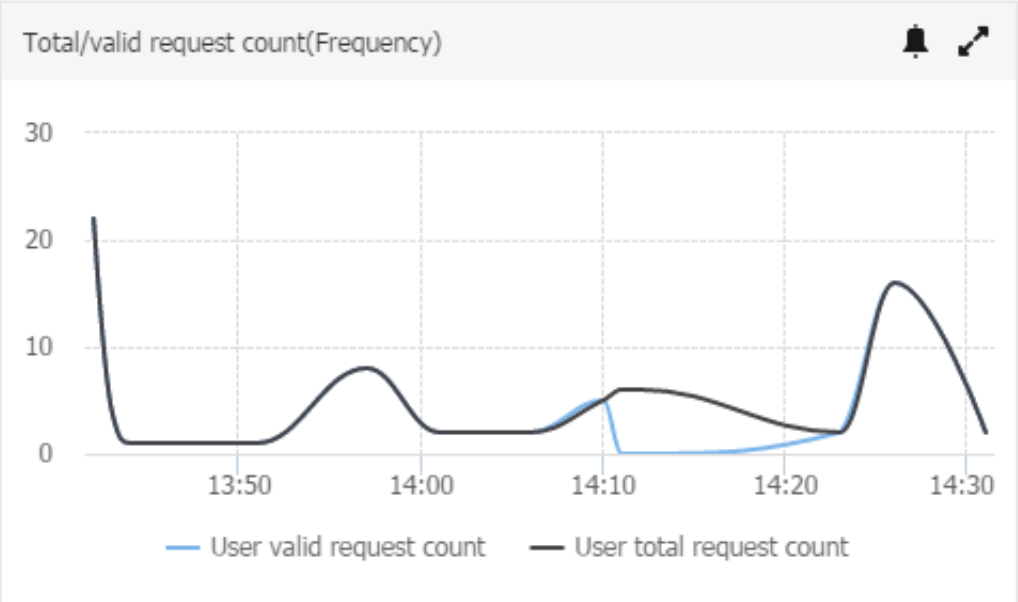
In some business scenarios, a valid request rate is expected to fall below 100%. For example, you may need to first check that an object exists and then perform a certain operation based on the existence of the object. In this case, if the object does not exist, the read request that checks its existence returns a 404 error code

(resource does not exist error). This inevitably produces a valid request rate of less than 100%.

For businesses that require high system availability, you can set an alarm rule that is triggered when the indicator falls below the expected threshold value.

- Total No. of requests and No. of valid requests

  This indicator reflects the system operation status from the perspective of the total traffic volume. When the No. of valid requests is not equal to the total No. of requests, this indicates that some requests have failed.



You can watch the fluctuations in the total No. of requests and No. of valid requests, especially when they sharply increase or decrease. In such cases, follow-up action is required. You can set alarm rules to make sure you receive prompt notifications. For periodic businesses, you can set periodic alarm rules (periodic alarms will be available soon). For more information, see *Alarm Service User Guide*.

- Request status distribution statistics

  When availability or the valid request rate falls below 100% (or the No. of valid requests is not equal to the total No. of requests), you can look at the request status

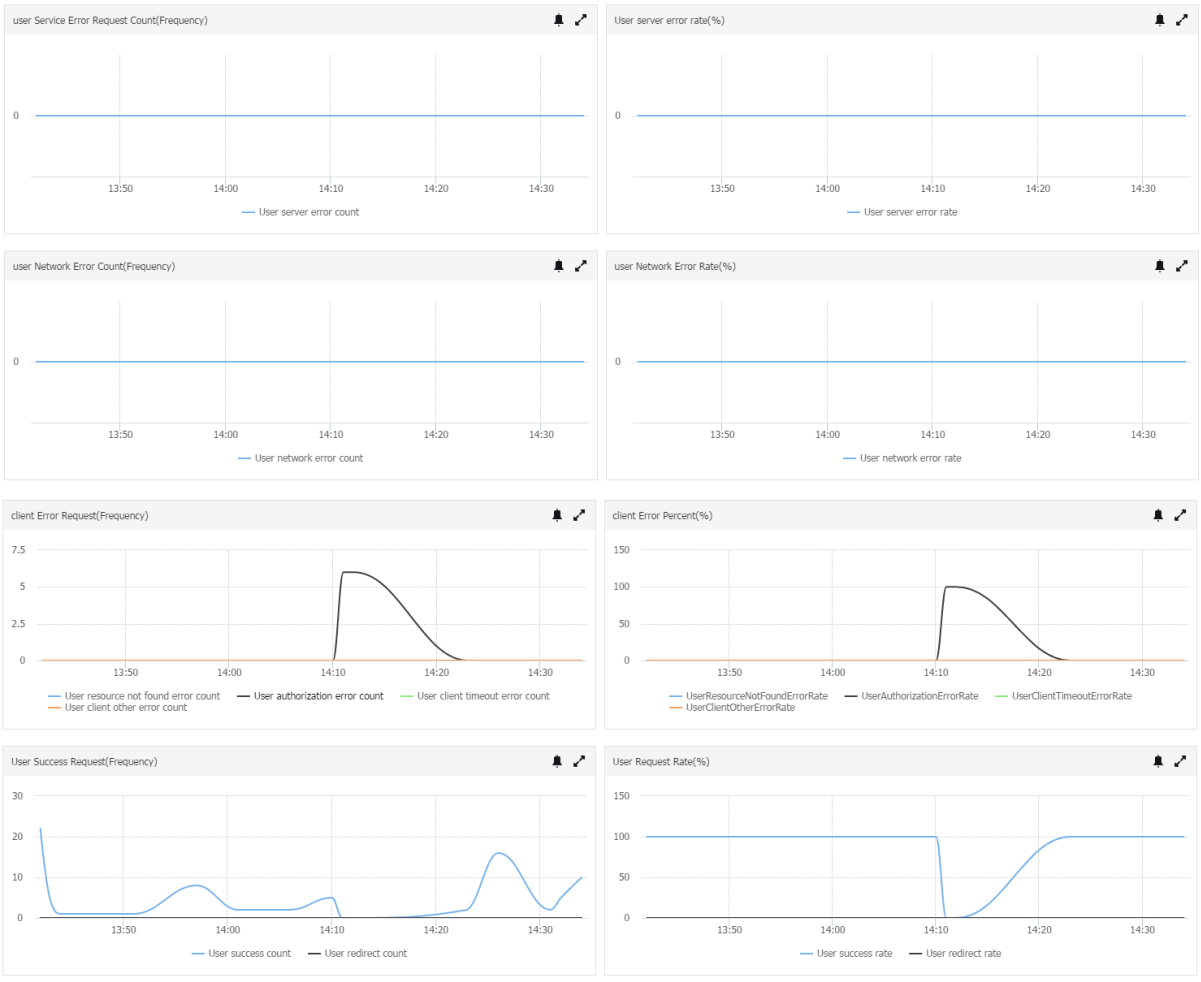
distribution statistics to quickly determine the request error types. For more information about this metric indicator, see *OSS Metric Indicator Reference Manual*.

User level request

Metric	Sum value	Percent
User authorization error count	12yunjiankong.metric.unitName.frequency	14.29%
User success count	72yunjiankong.metric.unitName.frequency	85.71%
Sum	84yunjiankong.metric.unitName.frequency	100%

Request status details monitoring

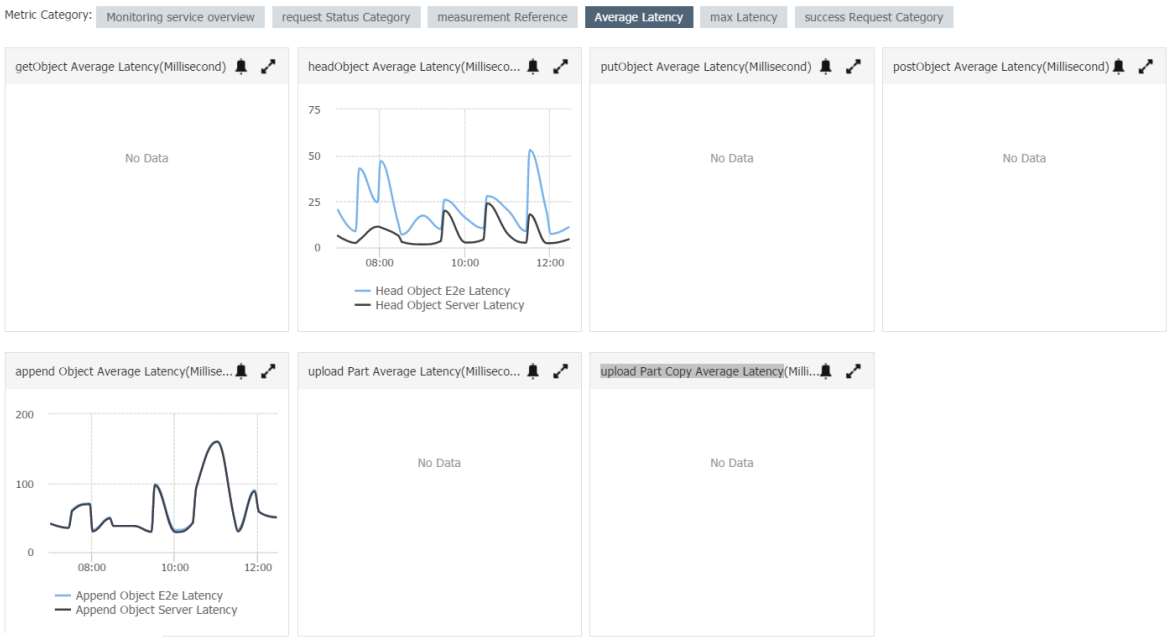
Request status details provides more details about the request monitoring status on the basis of request status distribution statistics. They let you monitor certain types of requests in more detail.



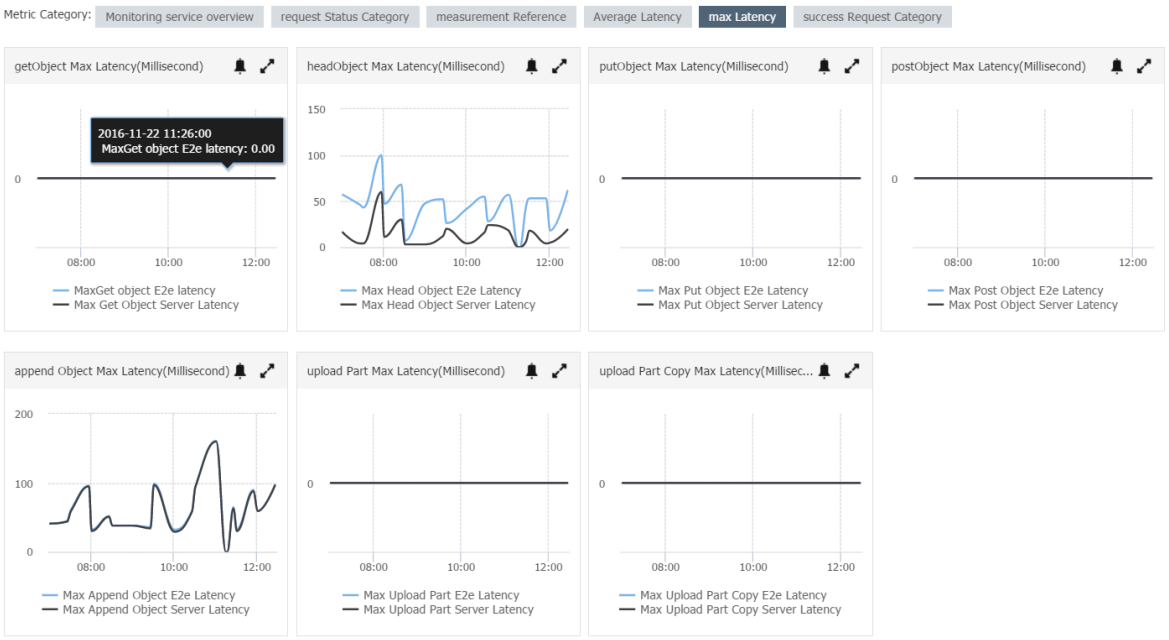
Performance monitoring

The monitoring service provides the following metric items that can be used as indicators for performance monitoring.

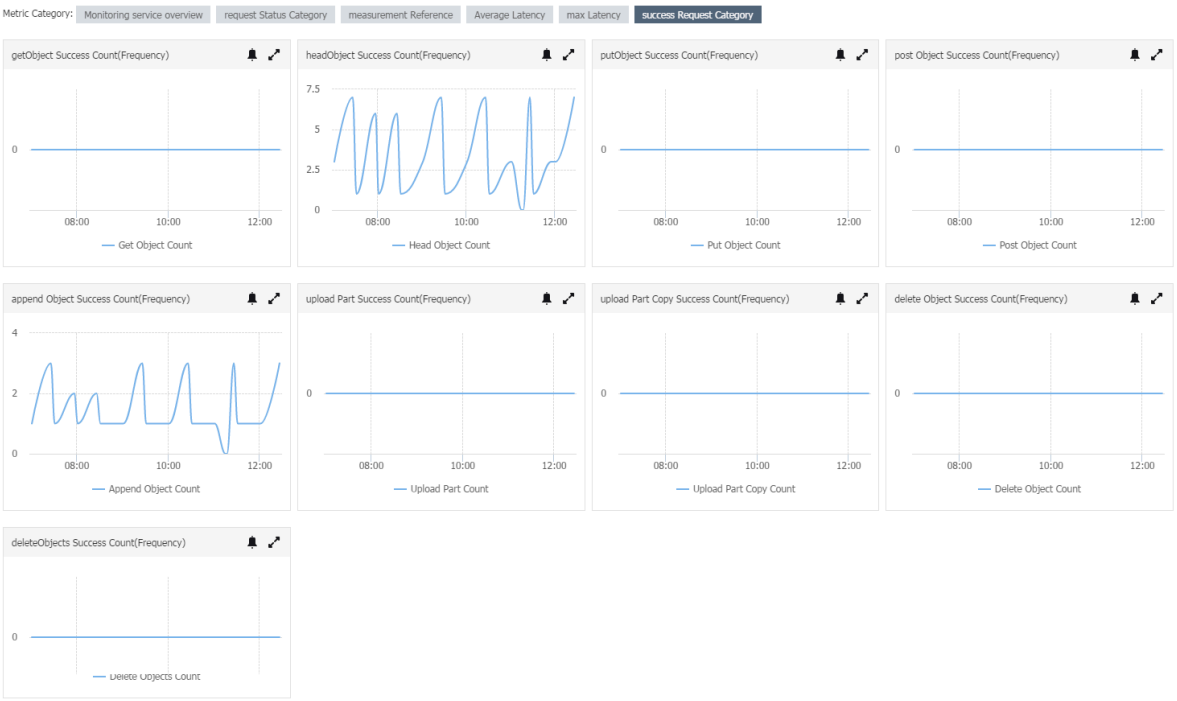
- Average latency: E2E average latency and Server average latency



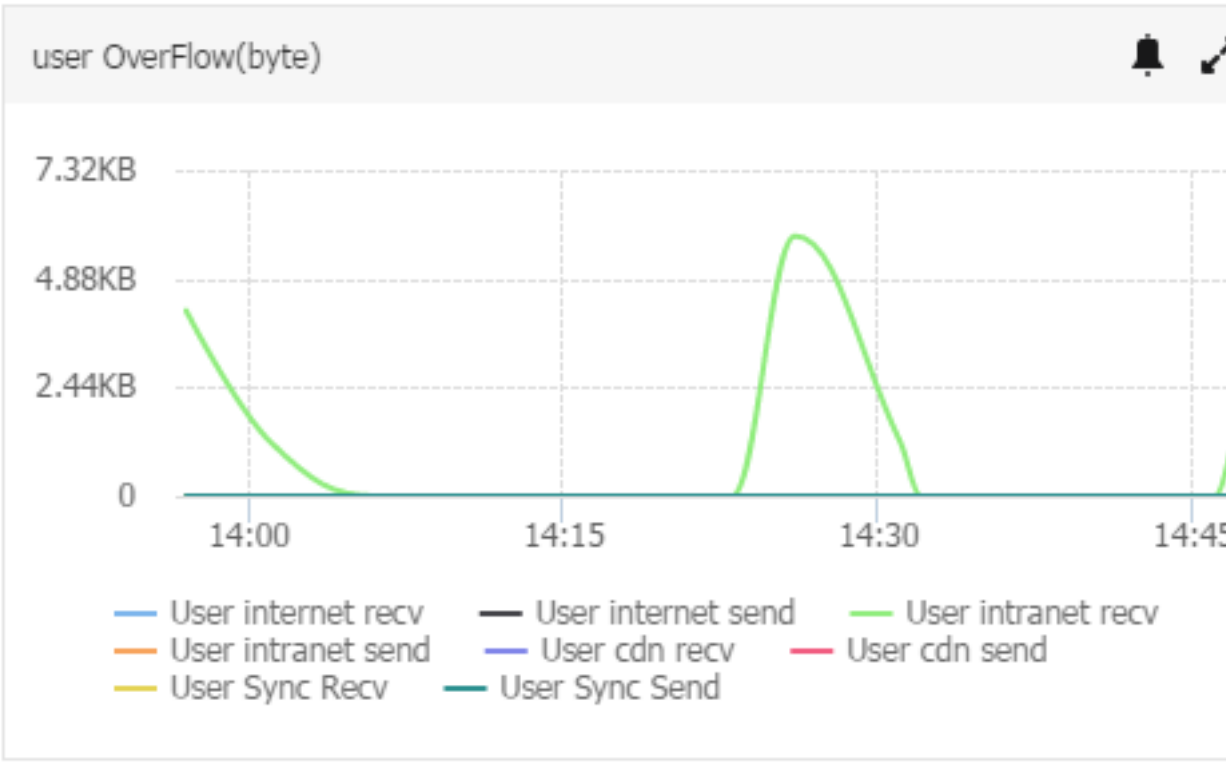
- Maximum latency: E2E maximum latency and Server maximum latency



- **Successful request categories**



- Traffic



The preceding metric items (except for ‘Traffic’) implement categorized monitoring based on API operation types:

- GetObject
- HeadObject
- PutObject
- PostObject
- AppendObject
- UploadPart
- UploadPartCopy

The latency indicators show the average or maximum time needed for API operation types to process requests. E2E latency is the indicator for end-to-end latency. Besides the time needed to process requests, it also includes the time needed to read requests and send responses, and the delay caused by network transmission. Server latency only includes the time needed to process the requests on the server, not the client-side transmission network latency. Therefore, if the E2E latency suddenly increases but the server latency does not change significan

tly, you can determine that the poor performance has been caused by network instability, instead of an OSS system fault.

In addition to the APIs mentioned previously, "successful request operation categories" also monitors the quantity of requests for the following two API operation types:

- DeleteObject
- Deleteobjects

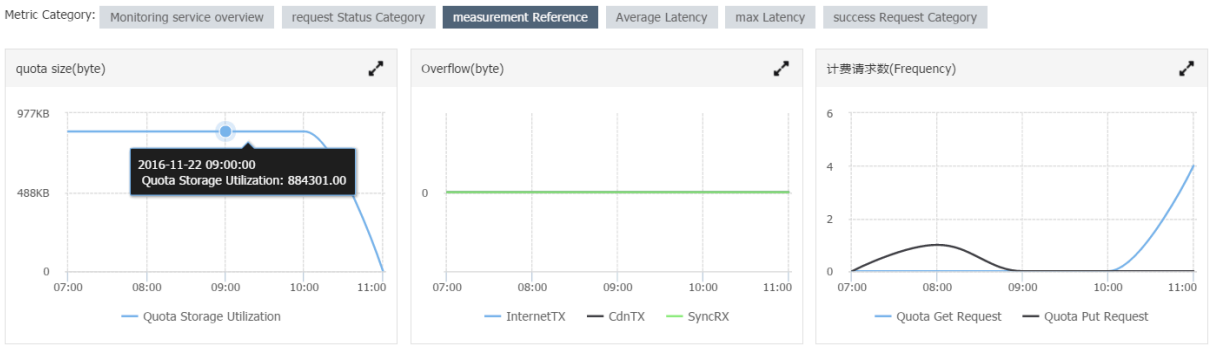
The traffic indicator is used to monitor the overall situation for a user or a specific bucket. It looks at the usage of network resources in Internet, intranet, CDN origin retrieval, cross-domain replication, and other such scenarios.

For performance-type indicators, we must focus on sudden and abnormal changes , such as when the average latency suddenly spikes or remains above the normal request latency baseline for a long period of time. You can set alarm rules that correspond to performance indicators, so that the relevant personnel are immediately notified if an indicator falls below or exceeds a threshold value. For businesses with periodic peaks and troughs, you can set periodic alarm rules for week on week, day on day, or hour on hour comparisons (periodic alarms will be available soon).

Billing monitoring

At press time, the OSS monitoring service can only monitor storage space, outbound Internet traffic, Put requests, and Get requests (not including cross-domain replicatio n outbound traffic and CDN outbound traffic). It does not support alarm setting or API read operations for billing data.

The OSS monitoring service collects bucket-level billing monitoring data on an hourly basis. In the monitoring view for a specific bucket, you can see graphs of continuous monitoring trends. Using the monitoring view, you can analyze your businesses' OSS resource usage trends and estimate future costs. See the following figure:



The OSS monitoring service also provides statistics on the quantity of user and bucket-level resources consumed each month. For example, the total amount of OSS resources consumed by an account or bucket starting from the 1st day of the month. These statistics are updated hourly. This increases your understanding of your resource usage and computation fees for the current month in real time, as shown in the following figure:

as shown in the following figure:

Note:
In the monitoring service, the provided billing data is pushed to the maximum extent possible, but this may cause some discrepancies with the actual bill amount. Please note that the Billing Center data is used in actual billing applications.

Tracking and diagnosis

Problem diagnosis

- Performance diagnosis

  Many subjective factors are involved in the determination of application performance. You must use the satisfaction of your business needs in your specific business scenario as a baseline, to determine if a performance problem occurs. Also, when a client initiates a request, factors that may cause performance problems may come from anywhere in the request chain. For example, problems may be caused by OSS overloads, client TCP configuration problems, or traffic bottlenecks in the basic network architecture.

  Therefore, when diagnosing performance problems, you must first set a reasonable baseline. Then, you use the performance indicators provided by the monitoring service to determine the potential root cause of any performance

problem. Next, you find detailed information in the relevant logs to help you further diagnose and troubleshoot any faults.

In the *Troubleshooting* section, we give examples of many common performance problems and troubleshooting measures. This can be used as a reference.

- Error diagnosis

  When requests from client applications are at fault, the clients receive error information from the server. The monitoring service records these errors and shows statistics for the various types of errors that may affect requests. You can also retrieve detailed information for individual requests from the server log, client log, and network log. Generally, the returned HTTP status code, OSS error code, and OSS error information can indicate the cause of the request failure.

  For error response information details, see *OSS error responses*.

- Using the logging function

  OSS provides a server logging function for user requests. This helps you track end-to-end detailed request logs.

  For instructions on the activation and use of the logging function, see *Set logging*.

  For more information on Log Service naming rules and record formats, see *Set access logging*.

- Using network logging tools

  In many situations, you can diagnose problems by using the logging function to record storage log and client application log data. However, in certain situations, you may need more details by using network logging tools.

  This is because capturing traffic exchanged between clients and the server can give you more detailed information on the data exchanged between clients and server and the underlying network conditions, which can help you investigate problems. For example, in some situations, user requests may report an error, but no request can be seen in the server log. In such cases, you can use the records logged by the OSS logging function to see if the cause of the problem lies with the client, or you can use network monitoring tools to check for a network problem.

  *Wireshark*is one of the most common network log analysis tools. This free protocol analyzer runs on the packet level and provides a view of detailed packet

information for various network protocols. This can help you troubleshoot packet loss and connection problems.

see *Wireshark User Guide*.

E2E tracking and diagnosis

Requests are initiated by a client application process and pass through the network environment to the OSS server, where they are processed. Then, a response is sent by the server over the network environment and received by the client. This is an end-to-end tracking process. Associating client application logs, network tracking logs, and server logs provides detailed information for you to troubleshoot the root cause of a problem and discover potential problems.

In OSS, the provided RequestIDs serve as identifiers used to associate the information from various logs. In addition, the log timestamps not only allow you to quickly query specific log time ranges, but can also show you the time points when request events and other client application, network, and service system events occurred during this period. This helps you analyze and investigate problems.

- RequestID

  Whenever the OSS receives a request, it allocates it a unique server request ID, its RequestID. In different logs, the RequestID is located in different fields:

  - In server logs recorded by the OSS logging function, the RequestID is located in the “Request ID” column.
  - In the process of network tracking (for example, when using Wireshark to capture data streams), the RequestID is the x-oss-request-id header value in the response message.
  - In client applications, you must use the client code to manually print the RequestID in the client log. At the press time, the latest Java SDK version already supported printing RequestID information for normal requests. You can use the getRequestId operation to retrieve RequestIDs from the results returned by different APIs. All OSS SDK versions allow you to print RequestIDs for abnormal requests. You can call the OSSException’s getRequestId method to obtain this information.

- Timestamps

  You can use timestamps to find relevant log entries. You must note that there may be some deviations between the client time and server time. On a client, you can use timestamps to search for server log entries recorded by the logging function. For this, you must add or subtract 15 minutes.

Troubleshooting

Common performance-related problems

- High average E2E latency, with low average server latency

  We have already discussed the differences between average E2E latency and average server latency. Therefore, we can say that high E2E latency and low server latency are caused by two possible reasons:

  - Slow client application response speed
  - Network factors

  A slow client application response speed can be caused by several possible reasons:

  - Limited number of available connections or threads
    - Use the relevant command to check if the system has a large number of connections in the TIME_WAIT status. If yes, adjust the core parameters to solve this problem.
    - When the number of available threads is limited, first check for bottlenecks affecting the client CPU, memory, network, or other resources. If no bottleneck is found, increase the number of concurrent threads properly.
    - If the problem persists, you have to optimize the client code. For example, you can use an asynchronous access method. You can also use the performance analysis function to analyze client application hotspots, and then perform the necessary optimization.
  - Insufficient resources, such as CPU, memory, or bandwidth
    - For this type of problem, you must first use the relevant system monitoring function to find client resource bottlenecks. Then, optimize the client code

to rationalize resource usage or increase the client resources (increase the number of cores or the memory).

Investigate network latency problems

Generally, high E2E latency due to network factors is temporary. You can use Wireshark to investigate temporary and persistent network problems, such as packet loss problems.

- Low average E2E latency, low average server latency, but high client request latency

When the client experiences high request latency, the most probable cause is that the requests are not reaching the server. Therefore, we must find out why the client requests are not arriving at the server.

Two client-side factors can cause high client request sending latency:

- A limited number of available connections or threads: see the solution described in the preceding section.
- Client requests are retried multiple times: In this situation, you must find and solve the cause of the request retries based on the retry information. You can follow these steps to determine if the client has a retry problem:
  - Check the client log. The detailed log entries indicate if retries have occurred . Using the OSS Java SDK as an example, you can search for the following warn or info-level log entries. If such entries are found in the log, this indicates that requests have been retried.

```
[ Server ] Unable to execute HTTP request :
    Or
  [ Client ] Unable to execute HTTP request :
```

  - If the client log level is debug, search for the following log entries (again we are using the OSS Java SDK as an example). If such entries exist, this indicates requests have been retried.

```
 Retrying on
```

If no problem with the client occurs, you must check for potential network problems, such as packet loss. You can use a tool such as Wireshark to investigate network problems.

- High average server latency

  If the server latency during downloads or uploads is high, this may be caused by the following two factors:

  - A large number of clients are frequently accessing the same small object.

    In this situation, you can view the server log recorded by the logging function to determine if a small object or a group of small objects are being frequently accessed in a short period of time.

    For download scenarios, we suggest you activate the CDN service for this bucket , to improve performance. This also reduces your traffic fees. In the case of upload, you may consider revoking write permissions for this object (bucket), if this does not affect your business.
  - Internal system factors

    For internal system problems or problems that cannot be solved through optimization, please provide our system staff with the RequestIDs in your client logs or in the logs recorded by the logging function, and they can help you solve the problem.

Server errors

When the number of server-side errors increases, two scenarios must to be considered:

- Temporary increase

  For this type of problem, you must adjust the retry policy in the client program and adopt a reasonable concession mechanism, such as exponential backoff. This not only avoids temporary service unavailability due to system optimization, upgrades , and other such operations (such as partition migration for system load balancing), but also avoids high pressure during business peaks.
- Permanent increase

  When the number of server-side errors sustainably increases, please provide our back-end staff with the RequestIDs in your client logs or in the logs recorded by the logging function, and they can help you find the problem.

Network errors

Network errors occur when the server is processing a request and the connection is lost (not due to a server-side issue), so the HTTP request header cannot be returned. In such a situation, the system records an *HTTP Status Code of 499* for this request. In the following situations, the server may change the request status code to 499:

- Before processing a received read/write request, if the server detects that the connection is unavailable, the request is recorded as 499.
- When the server is processing a request and the client preemptively closes the connection, the request is recorded as 499.

In summary, a network error occurs during the request process when a client independently closes the request or the client is disconnected from the network. If the client independently closes requests, you can check the client code, to identify the cause and time of the client's disconnection from OSS. When the client loses its network connection, you can use a tool such as Wireshark to investigate network connection problems.

Client errors

- Increase in client authorization errors

  If you detect an increase in client authorization errors or the client receives a large number of 403 request errors, this is most commonly caused by the following problems:

  - The bucket domain name accessed by the user is incorrect.
    - If the user uses a third-level or second-level domain name to access a bucket, this may cause a 403 error if the bucket is not in the region indicated by the domain name. For example, if you have created a bucket in the Hangzhou region, but a user attempts to access it using the domain name Bucket.oss-cn-shanghai.aliyuncs.com. In this case, you must confirm the bucket's region and then correct the domain name information.
    - If you have activated the CDN acceleration service, this problem may occur when CDN binds an incorrect origin retrieval domain name. In this case, check that the CDN origin retrieval domain name is the bucket's third-level domain name.
  - If you encounter 403 errors when using JavaScript clients, this may be caused by a problem in the CORS (Cross-Origin Resource Sharing) settings, because web browsers implement "same source policy" security restrictions. In this

case, you must check the bucket’s CORS settings and correct any errors. For information about CORS settings, see *CORS*.

- Access control problems can be divided into four types:
  - When you use a primary AK for access, you must check the AK settings for errors if the AK is invalid.
  - When you use a RAM sub-account for access, you must check that the sub-account is using the correct sub-account AK and that the sub-account has the relevant permissions.
  - When you use temporary STS tokens for access, you must confirm that the temporary token has not expired. If the token has expired, apply for a new one.
  - If you use bucket or object settings for access control, you must check that the bucket or object to be accessed supports the relevant operations.
- When you authorize third-party downloads (using signed URLs to access OSS resources), if access was previously normal and then suddenly reports a 403 error, it is likely that the URL has expired.
- When RAM sub-accounts use OSS utilities, this may also produce 403 errors. These utilities include ossftp, ossbrowser, and the OSS console client. When you enter the relevant AK information during logon and the system throws an error , if you entered the correct AK, you must check that the AK is a sub-account AK and that this sub-account has permission for GetService and other operations.

· Increase in client-side ‘resource does not exist’ errors

When the client receives a 404 error, this means that you are attempting to access a resource or information that does not exist. When the monitoring service detects an increase in ‘resource does not exist’ errors, this is most likely caused by one of the following problems:

- Service usage: For example, when you first need to check that an object exists before performing another operation and you call the doesObjectExist method (using the Java SDK as an example), if the object does not exist, the client receives

the value "false". However, the server actually produces a 404 request error. Therefore, in this business scenario, 404 errors are normal.

- The client or another process previously deleted this object. You can confirm this problem by searching for the relevant object operation in the server log recorded by the logging function.
- Network faults case packet loss and retries. For example, the client may initiate a delete operation to delete a certain object. The request reaches the server and successfully executes the delete operation. However, if the response packet is lost during transmission on the network, the client initiates a retry. This second request then produces a 404 error. You can confirm that network problems are producing 404 errors using the client log and server log:
  - Check for retry requests in the client application log.
  - Check if the server log shows two delete operations for this object and that the first delete operation has an HTTP status of 2xx.

· Low valid request rate and high number of other client-side request errors

The valid request rate is the number of requests that return an HTTP status code of 2xx/3xx as a percentage of total requests. Status codes of 4XX or 5XX indicate a failed request and reduce the valid request rate. Other client-side request errors indicate requests errors other than the following: server errors (5xx), network errors (499), client authorization errors (403), resource does not exist errors (404), and client time-out errors (408 or OSS error code: RequestTimeout 400).

Check the server log recorded by the logging function to determine the specific errors encountered by these requests. You can see *OSS error responses* to find a list of common error codes returned by OSS. Then, check the client code to find and solve the specific cause of these errors.

Abnormal increase in storage capacity

If storage capacity increases abnormally without a corresponding increase in upload requests, this is generally caused by a delete problem. In such a case, check for the following two factors:

- When the client application uses a specific process to regularly delete storage objects to free up space: The investigation processes for this request are as follows:
  1. Check if the valid request rate has decreased, because a failed delete request may cause storage objects to fail to be deleted as expected.
  2. Find the specific cause for the decrease in the valid request rate by looking at the error types of the requests. Then, you can combine the specific client logs to see the detailed error information (for example, the STS temporary token used to free up storage space may have expired).
- When the client sets a LifeCycle to delete storage objects: Use the console or an API to check that the current bucket LifeCycle value is the same as before. If not, modify the configuration and use the server log recorded by the logging function to find information on the previous modification of this value. If the LifeCycle is normal but inactive, contact an OSS system administrator to help identify the problem.

Other OSS problems

If the Troubleshooting section did not cover your problem, use one of the following methods to diagnose and troubleshoot the problem.

1. View the OSS monitoring service, to see if there have been any changes compared to the expected baseline behavior. Using the monitoring view, you may be able to determine if this problem is temporary or permanent and which storage operations are affected.
2. The monitoring information can help you search the server log data recorded by the logging function, to find information on any errors that may have occurred when the problem started. This information may be able to help you find and solve the problem.
3. If the information in the server log is insufficient, use the client long to investigat e the client application, or use a network tool such as Wireshark to check your network for problems.

19 Cloud data processing

Image Processing

For introduction and more information about functions, see *Image Processing*.

Media Processing

Media Processing is a transcoding computing service for multimedia data. It provides an economic, easy-to-use, elastic, and highly scalable method for conversion of audio and video stored on OSS into formats suitable for playing on PCs, TVs, or mobile devices.

Media Processing was constructed based on Alibaba Cloud computing services. In the past, users had to make a high investment to purchase, build, and manage transcoding software and hardware, and perform complex configuration optimizati on, transcoding parameter adaptation, and other operations. Media Processing has transformed everything. It has enhanced the elasticity of cloud computing services . Media Processing offers transcoding capabilities to fulfill business transcoding demands to its extreme and also curbs the wastage of resources.

Media Processing functions include the Web management console, service APIs, and SDKs. Users can use and manage Media Processing and integrate transcoding functions into their own apps and services.

Media Processing function list

- Transcoding
- Pipelines
- Screenshot
- Media information
- Watermark
- Preset templates
- Custom templates
- Video clip output
- Resolution scaling
- M3U8 custom segment length output
- Audio/Video extraction

- Video image rotation
- Video-to-GIF conversion

For introduction and more information about functions, see *Media Processing documentation*.

20 Hide

20.1 Access control

20.1.1 Bucket permission separation

Another scenario is introduced in this section. If another user is using the developed app, you can use an individual bucket to store your app data. Assume that the bucket is the ram-test-app. In consideration of permission separation, the application server must not be allowed to access the ram-test-app; that is, the account ram_test_pub is permitted only to read ram-test-dev. This can also be realized through the RAM permission system. The procedure is as follows:

1. Because the system has no default bucket-level policy, we must create a custom policy.

   The bucket access policy is shown as follows. For more information, see *RAM Policy Description* and *OSS Authorization FAQ* .

   ```
   {
   " Version ": " 1 ",
   " Statement ": [
    {
      " Effect ": " Allow ",
      " Action ": [
        " oss : ListObject  s ",
        " oss : GetObject "
      ],
      " Resource ": [
        " acs : oss :*:*: ram - test - dev ",
        " acs : oss :*:*: ram - test - dev /*"
      ]
    }
   ]
   }
   ```

   After setting, we can see the policy in the custom authorization policy list.

2. In user authorization management, add this policy to the selected authorization policy list. Also in Users > Management > Authorization policy, all previously granted OSS read permissions can be revoked.

3. Test the validity of permission configured.

   · The object in ram-test-dev can be accessed:

```
$./ osscmd   get   oss :// ram - test - dev / test . txt   test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   oOhue
 ****** Frogv  - k   OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
 100 %  The   object   test . txt   is   downloaded   to   test .
 txt ,  please   check .
 0 . 047 ( s )  elapsed
```

   · The object in ram-test-app cannot be accessed:

```
$./ osscmd   get   oss :// ram - test - app / test . txt   test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   oOhue
 ****** Frogv  - k   OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
  Error   Headers :
 [(' content - length ', ' 229 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 5646ED53F9  EEA2F33241  91A2 '), (' date ', ' Sat ,  14   Nov
 2015   08 : 14 : 11   GMT '), (' content - type ', ' applicatio
 n / xml ')]
  Error   Body :
 <?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
 < Error >
   < Code > AccessDeni  ed </ Code >
   < Message > AccessDeni  ed </ Message >
   < RequestId > 5646ED53F9  EEA2F33241  91A2 </ RequestId >
   < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs .
 com </ HostId >
   </ Error >
  Error   Status :
  403
  get   Failed !
```

   · Files cannot be uploaded to oss-test-app:

```
$./ osscmd   put   test . txt   oss :// ram - test - app / test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   oOhue
 ****** Frogv  - k   OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA

  100 %  Error   Headers :
 [(' content - length ', ' 229 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 5646ED7BB8  DE437A912D  C7A8 '), (' date ', ' Sat ,  14   Nov
 2015   08 : 14 : 51   GMT '), (' content - type ', ' applicatio
 n / xml ')]
  Error   Body :
 <?  XML   version  = " 1 . 0 "  encoding  = " UTF - 8  "? >
 < Error >
   < Code > AccessDeni  ed </ Code >
   < Message > AccessDeni  ed </ Message >
   < RequestId > 5646ED7BB8  DE437A912D  C7A8 </ RequestId >
   < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs .
 com </ HostId >
 </ Error >
  Error   status :
  403
```

```
put   Failed !
```

Using the preceding configuration, we have successfully separated the permissions for ram-test-dev and ram-test-app.

The preceding section explains how to use the subaccount permission control function to separate permissions and minimize the potential risk of information leakage.

If you want to implement more complex access control, see *RAM User Guide*.

20.1.2 STS temporary access authorization

In the previous documents, we used only the RAM user functions. These user accounts are for long-term normal use. This poses as a serious risk if the RAM user permissions cannot be promptly revoked in case of information leakage.

In the previous example, assume that our developer’ s app allows users to upload data to the OSS bucket am-test-app and currently, the number of app users is large. In this case, how can the app securely grant data upload permissions to many users and how can it be certain of storage isolation among multiple users?

In such scenarios, we need to grant users temporary access using STS. STS can be used to specify a complex policy that restricts specified users by only granting them the minimum necessary permissions.

Create a role

Based on the example in the previous document, the app user has a bucket, ram-test-app, to store personal data. A role can be created as follows:

1. Create a RAM user account named ram_test_app using the process illustrated in the previous documents. Do not grant this account any permissions, because it inherits the permissions of a role which it assumes.
2. Create roles. Here you must create two roles for users to perform read operations and to upload files respectively.
   - Log on to the RAM console and select Roles > New Role.
   - Select a role type. Here you must select User role.
   - Enter the role type information. Because this role has been used by its own Alibaba Cloud account. Use the default setting.
   - Configure basic role information.

3. When the role was created, it did not have any permissions. Therefore, we must create a custom authorization policy using the process described earlier. The following is the authorization policy:

```
{
" Version ": " 1 ",
" Statement ": [
 {
   " Effect ": " allow  ",
   " Action ": [
     " oss : ListObject  s ",
     " Oss :  GetObject "
   ],
   " Resource ": [
     " acs : oss :*:*: ram - test - app ",
     " acs : oss :*:*: ram - test - app /*"
   ]
 }
]
}
```

This indicates read-only permission for ram-test-app.

4. After the policy is established, give the role RamTestAppReadOnly the ram-test-app read-only permission on the role management page.

5. Perform the same procedure to create the role RamTestAppWrite and use a custom authorization policy to grant ram-test-app write permission. The authorization policy is as follows:

```
{
" Version ": " 1 ",
" Statement ": [
 {
   " Effect ": " Allow ",
   " Action ": [
     " oss : DeleteObje  ct ",
     " oss : ListParts ",
     " oss : AbortMulti  partUpload ",
     " oss : PutObject "
   ],
   " Resource ": [
     " acs : oss :*:*: ram - test - app ",
     " acs : oss :*:*: ram - test - app /*"
   ]
 }
]
```

```
}
```

Now we have created two roles, RamTestAppReadOnly and RamTestAppWrite, with read-only and write permissions for ram-test-app, respectively.

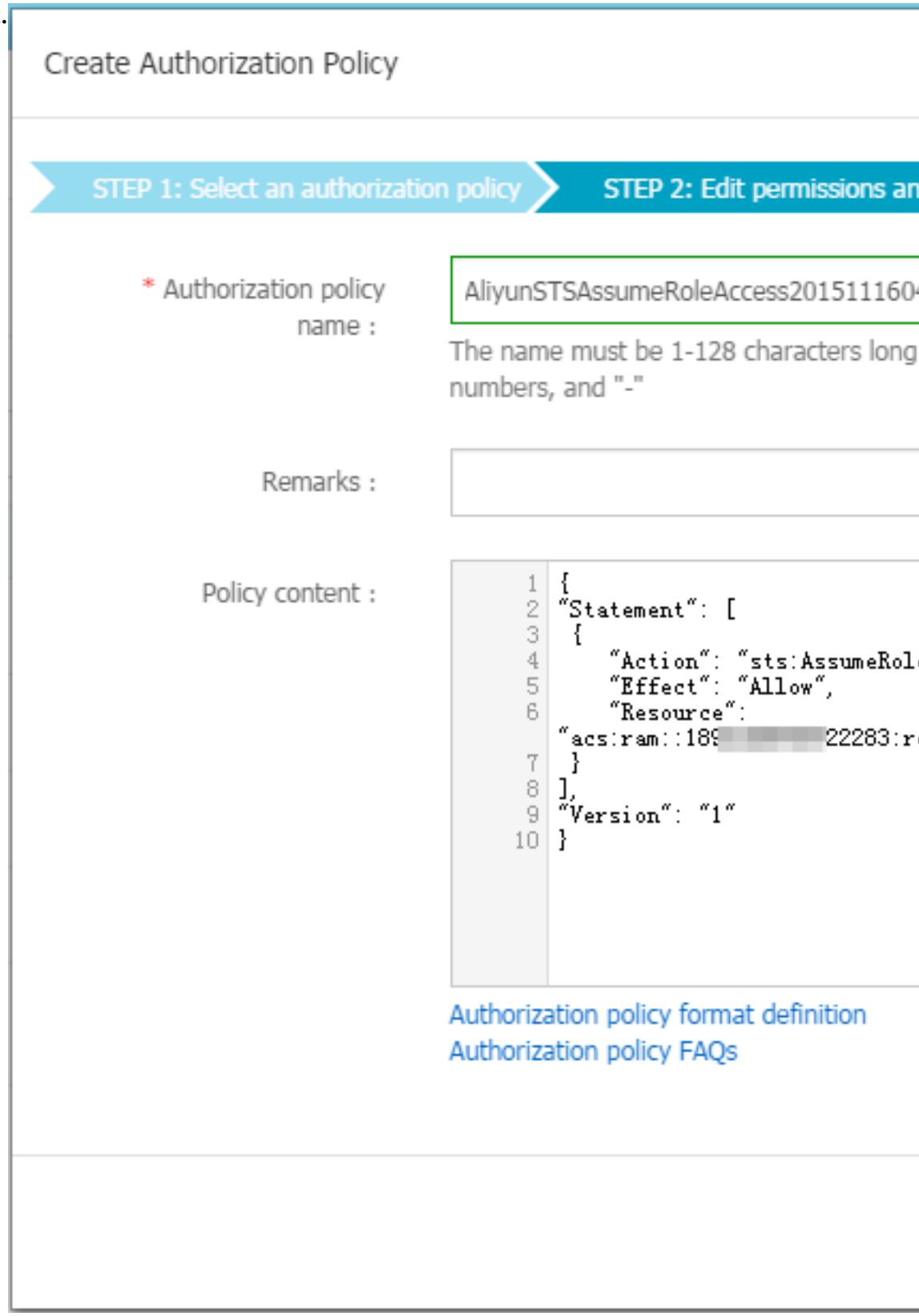
Temporary access authorization

After creating roles, we can use them to grant temporary access to OSS.

Preparation

Authorization is required for assuming roles. Otherwise, any RAM user could assume these roles, which can lead to unpredictable risks. Therefore, to assume correspond ing roles, a RAM user needs to have explicitly configured permissions.

1. Create two custom authorization policies in authorization policy management.




```
{
" Statement ": [
 {
   " Action ": " sts : AssumeRole ",
   " Effect ": " Allow ",
```

```
    " Resource ": " acs : ram :: 1894xxxxxx  722283 : role /
 ramtestapp  readonly "
 }
],
" Version ": " 1 "
}
```

Create another custom authorization policy using the same method:

```
{
" Statement ": [
 {
   " Action ": " sts : AssumeRole ",
   " Effect ": " Allow ",
   " Resource ": " acs : ram :: 1894xxxxxx  722283 : role /
 ramtestapp  write "
 }
],
" Version ": " 1 "
}
```

Here, the content entered after Resource is a role’s ID. Role IDs can be found in Roles > Role Details .

2. Grant the two authorization policies to the account ram_test_app.

Use STS to grant access permissions

Now, we are ready with the platform to officially use STS to grant access permissions.

Here we use a simple STS Python command line tool *sts.py*. The calling method is as follows:

```
$ python  ./ sts . py   AssumeRole   RoleArn = acs : ram ::
 1894xxxxxx  722283 : role / ramtestapp  readonly   RoleSessio  nName
 = usr001   Policy ='{" Version ":" 1 "," Statement ":[{" Effect ":"
 Allow "," Action ":[" oss : ListObject  s "," oss : GetObject "],"
 Resource ":[" acs : oss :*:*: ram - test - app "," acs : oss :*:*:
 ram - test - app /*"]}]}'  DurationSe  conds = 1000  -- id = id  --
 secret = secret
```

- RoleArn: indicates the ID of a role to be assumed. Role IDs can be found inRoles > Role details .
- RoleSessionName: indicates the name of the temporary credentials. Generally, we recommend that you separate this using different application users.
- Policy: indicates a permission restriction, which is added when the role is assumed .
- DurationSeconds: indicate the validity time of the temporary credentials in seconds. The minimum value is 900, and the maximum value is 3600.
- id and secret: indicate the AccessKey of the RAM user to assume a role.

Here, we need to explain what is meant by “Policy”. The policy mentioned here is used to restrict the temporary credential permissions after a role is assumed. Ultimately, the permissions obtained by means of temporary credentials are overlapping permissions of the role and the policy passed in.

When a role is assumed, a policy can be entered to increase the flexibility. For example, when uploading the files, we can add different upload path restrictions for different users. This is shown in the following example.

Now, let's test the STS function. To test the bucket, first use the console to put the file test.txt in ram-test-app, with the content ststest.

Firstly, use the RAM user account ram_test_app to directly access the file. Next, replace AccessKey with your own access key used in the test.

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
$./ osscmd   get   oss :// ram - test - app / test . txt   test . txt
  -- host = oss - cn - hangzhou . aliyuncs . com  - i   oOhue ******
 Frogv  - k   OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
 Error   Headers :
[(' content - length ', ' 229 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 564A94D444  F4D8B2225E  4AFE '), (' date ', ' Tue ,  17   Nov   2015
   02 : 45 : 40   GMT '), (' content - type ', ' applicatio  n / xml
 ')]
 Error   Body :
<?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
< Error >
  < Code > AccessDeni  ed </ Code >
  < Message > AccessDeni  ed </ Message >
  < RequestId > 564A94D444  F4D8B2225E  4AFE </ RequestId >
  < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs . com
 </ HostId >
</ Error >
 Error   Status :
 403
 get   Failed !
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
$./ osscmd   put   test . txt   oss :// ram - test - app / test . txt
  -- host = oss - cn - hangzhou . aliyuncs . com  - i   oOhue ******
 Frogv  - k   OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
 100 %  Error   Headers :
[(' content - length ', ' 229 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 564A94E5B1  119B445B9F  8C3A '), (' date ', ' Tue ,  17   Nov   2015
   02 : 45 : 57   GMT '), (' content - type ', ' applicatio  n / xml
 ')]
 Error   Body :
<?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
< Error >
  < Code > AccessDeni  ed </ Code >
  < Message > AccessDeni  ed </ Message >
  < RequestId > 564A94E5B1  119B445B9F  8C3A </ RequestId >
  < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs . com
 </ HostId >
</ Error >
```

```
Error   Status :
403
put   Failed !
```

Without access permission, access attempts using the RAM user account ram_test_app are failed.

Use temporary authorization for downloads

Now, we use STS to download files. To make it simple to understand, the entered policy and the role policy are the same. The expiration time is set to 3600s, and the app user here is usr001. The steps are as follows:

1. **Use STS to obtain a temporary credential.**

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
$ python  ./ sts . py   AssumeRole   RoleArn = acs : ram ::
1894xxxxxx  722283 : role / ramtestapp  readonly   RoleSessio
nName = usr001   Policy ='{" Version ":" 1 "," Statement ":[{"
Effect ":" Allow "," Action ":[" oss : ListObject  s "," oss :
GetObject "]," Resource ":[" acs : oss :*:*: ram - test - app ","
acs : oss :*:*: ram - test - app /*"]}]}' -- id = oOhue ******
Frogv  -- secret = OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
 https :// sts . aliyuncs . com /? SignatureV  ersion = 1 . 0
& Format = JSON & Timestamp = 2015 - 11 - 17T03 % 3A07 % 3A25Z
& RoleArn = acs % 3Aram % 3A % 3A1894xxxx  xx722283 % 3Arole %
2Framtesta  ppreadonly & RoleSessio  nName = usr001 & AccessKeyI
d = oOhu ****** 3Frogv & Policy =% 7B % 22Version % 22 % 3A % 221
% 22 % 2C % 22Statemen  t % 22 % 3A % 5B % 7B % 22Effect % 22 % 3A
% 22Allow % 22 % 2C % 22Action % 22 % 3A % 5B % 22oss % 3AListObje
cts % 22 % 2C % 22oss % 3AGetObjec  t % 22 % 5D % 2C % 22Resource
% 22 % 3A % 5B % 22acs % 3Aoss % 3A % 2A % 3A % 2A % 3Aram - test
- app % 22 % 2C % 22acs % 3Aoss % 3A % 2A % 3A % 2A % 3Aram - test
- app % 2F % 2A % 22 % 5D % 7D % 5D % 7D & SignatureM  ethod =
HMAC - SHA1 & Version = 2015 - 04 - 01 & Signature = bshxPZpwRJ
v5ch3SjaBi  XLodwq0 % 3D & Action = AssumeRole & SignatureN  once =
53e1be9c - 8cd8 - 11e5 - 9b86 - 008cfa5e49  38
{
  " AssumedRol  eUser ": {
    " Arn ": " acs : ram :: 1894xxxxxx  722283 : role / ramtestapp
readonly / usr001 ",
    " AssumedRol  eId ": " 3174463476  57426289 : usr001 "
  },
  " Credential  s ": {
    " AccessKeyI  d ": " STS .  3mQEbNf ****** wa180Le ",
    " AccessKeyS  ecret ": " B1w7rCbR4d  zGwNYJ ****** 3PiPqKZ3gj
QhAxb6mB ",
    " Expiration ": " 2015 - 11 - 17T04 : 07 : 25Z ",
    " SecurityTo  ken ": " CAESvAMIAR  KAASQQ ****** 7683CGlhdG
sv2 / di8uI + X ****** DxM5FTd0fp  5wpPK / 7UctYH2MJ ///
c4yMN1PUCc  EHI1zppCIN  mpDG2XeNA3  OS16JwS6ES  mI50sHyWBm
sYkCJW15gX  nfhz / OK + mSp1bYxlfB  33qfgCFe97  Ijeuj8RMgq
Fx0Hny2BzG  hhTVFMuM21  RRWJOZnR5Y  zl1T3dhMTg  wTGUiEjMxN
zQ0NjM0NzY  1NzQyNjI4O  SoGdXNyMDA  xMJTrgJ2RK  joGUnNhTUQ
1QpsBCgExG  pUBCgVBbG ****** CgxBY3Rpb2  5FcXVhbHMS  BkFjdGlvbh
ogCg9vc3M6  TGlzdE9iam  VjdHMKDW9z  czpHZXRPYm  plY3QSUgoO
UmVzb3VyY2  VFcXVhbHMS  CFJlc291cm  NlGjYKGGFj  czpvc3M6Kj
oqOnJhbS10  ZXN0LWFwcA  oaYWNzOm9z  czoq ****** FtLXRlc3Qt
YXBwLypKED  E4OTQxODk3  Njk3MjIyOD  NSBTI2ODQy  Wg9Bc3N1bW
```

```
 VkUm9sZVVz  ZXJgAGoSMz  E3NDQ2MzQ3  NjU3NDI2Mj  g5chJyYW10
 ZXN0YXBwcm  VhZG9ubHk ="
   },
   " RequestId ": " 8C009F64 - F19D - 4EC1 - A3AD - 7A718CD0B4  9B "
 }
```

2. **Use the temporary credential to download files. Here sts_token is the SecurityToken returned by the STS.**

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $./ osscmd   get   oss :// ram - test - app / test . txt   test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   STS .
 3mQEbNf ****** wa180Le  - k   B1w7rCbR4d  zGwNYJ ****** 3PiPqKZ3gj
 QhAxb6mB  -- sts_token = CAESvAMIAR  KAASQQ ****** 7683CGlhdG  sv2
 / di8uI + X ****** DxM5FTd0fp  5wpPK / 7UctYH2MJ /// c4yMN1PUCc
 EHI1zppCIN  mpDG2XeNA3  OS16JwS6ES  mI50sHyWBm  sYkCJW15gX  nfhz
 / OK + mSp1bYxlfB  33qfgCFe97  Ijeuj8RMgq  Fx0Hny2BzG  hhTVFMuM21
 RRWJOZnR5Y  zl1T3dhMTg  wTGUiEjMxN  zQ0NjM0NzY  1NzQyNjI4O
 SoGdXNyMDA  xMJTrgJ2RK  joGUnNhTUQ  1QpsBCgExG  pUBCgVBbG ******
 CgxBY3Rpb2  5FcXVhbHMS  BkFjdGlvbh  ogCg9vc3M6  TGlzdE9iam
 VjdHMKDW9z  czpHZXRPYm  plY3QSUgoO  UmVzb3VyY2  VFcXVhbHMS
 CFJlc291cm  NlGjYKGGFj  czpvc3M6Kj  oqOnJhbS10  ZXN0LWFwcA
 oaYWNzOm9z  czoq ****** FtLXRlc3Qt  YXBwLypKED  E4OTQxODk3
 Njk3MjIyOD  NSBTI2ODQy  Wg9Bc3N1bW  VkUm9sZVVz  ZXJgAGoSMz
 E3NDQ2MzQ3  NjU3NDI2Mj  g5chJyYW10  ZXN0YXBwcm  VhZG9ubHk =
  100 %  The   object   test . txt   is   downloaded   to   test .
 txt ,  please   check .
  0 . 061 ( s )  elapsed
```

3. **As you can see, we can use the temporary credentials to download the file. Next, we will test if we can use them to upload a file.**

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $./ osscmd   put   test . txt   oss :// ram - test - app / test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   STS .
 3mQEbNf ****** wa180Le  - k   B1w7rCbR4d  zGwNYJ ****** 3PiPqKZ3gj
 QhAxb6mB  -- sts_token = CAESvAMIAR  KAASQQ ****** 7683CGlhdG  sv2
 / di8uI + X ****** DxM5FTd0fp  5wpPK / 7UctYH2MJ /// c4yMN1PUCc
 EHI1zppCIN  mpDG2XeNA3  OS16JwS6ES  mI50sHyWBm  sYkCJW15gX  nfhz
 / OK + mSp1bYxlfB  33qfgCFe97  Ijeuj8RMgq  Fx0Hny2BzG  hhTVFMuM21
 RRWJOZnR5Y  zl1T3dhMTg  wTGUiEjMxN  zQ0NjM0NzY  1NzQyNjI4O
 SoGdXNyMDA  xMJTrgJ2RK  joGUnNhTUQ  1QpsBCgExG  pUBCgVBbG ******
 CgxBY3Rpb2  5FcXVhbHMS  BkFjdGlvbh  ogCg9vc3M6  TGlzdE9iam
 VjdHMKDW9z  czpHZXRPYm  plY3QSUgoO  UmVzb3VyY2  VFcXVhbHMS
 CFJlc291cm  NlGjYKGGFj  czpvc3M6Kj  oqOnJhbS10  ZXN0LWFwcA
 oaYWNzOm9z  czoq ****** FtLXRlc3Qt  YXBwLypKED  E4OTQxODk3
 Njk3MjIyOD  NSBTI2ODQy  Wg9Bc3N1bW  VkUm9sZVVz  ZXJgAGoSMz
 E3NDQ2MzQ3  NjU3NDI2Mj  g5chJyYW10  ZXN0YXBwcm  VhZG9ubHk =
  100 %  Error   Headers :
 [(' content - length ', ' 254 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 564A9A2A17  90CF0F53C1  5C82 '), (' date ', ' Tue ,  17   Nov
 2015   03 : 08 : 26   GMT '), (' content - type ', ' applicatio  n
 / xml ')]
  Error   Body :
 <?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
 < Error >
   < Code > AccessDeni  ed </ Code >
   < Message > Access   denied   by   authorizer ' s   policy .</
 Message >
   < RequestId > 564A9A2A17  90CF0F53C1  5C82 </ RequestId >
```

```
  < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs .
com </ HostId >
</ Error >
 Error   Status :
 403
 put   Failed !
```

The file upload is failed. This is because the assumed role only has download permission hence.

Use temporary authorization for uploads

Now, we will try to use STS to upload a file. The steps are as follows:

1. **Obtain an STS temporary credential. The app user is usr001.**

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $ python  ./ sts . py   AssumeRole   RoleArn = acs : ram ::
 1894xxxxxx  722283 : role / ramtestapp  write   RoleSessio  nName
 = usr001   Policy ='{" Version ":" 1 "," Statement ":[{" Effect ":"
 Allow "," Action ":[" oss : PutObject "]," Resource ":[" acs : oss
 :*:*: ram - test - app / usr001 /*"]}]}' -- id = oOhue ****** Frogv
  -- secret = OmVwFJO3qc  T0 ****** FhOYpg3p0K  nA
  https :// sts . aliyuncs . com /? SignatureV  ersion = 1 . 0
 & Format = JSON & Timestamp = 2015 - 11 - 17T03 % 3A16 % 3A10Z
 & RoleArn = acs % 3Aram % 3A % 3A1894xxxx  xx722283 % 3Arole %
 2Framtesta  ppwrite & RoleSessio  nName = usr001 & AccessKeyI  d
 = oOhu ****** 3Frogv & Policy =% 7B % 22Version % 22 % 3A % 221 %
 22 % 2C % 22Statemen  t % 22 % 3A % 5B % 7B % 22Effect % 22 % 3A %
 22Allow % 22 % 2C % 22Action % 22 % 3A % 5B % 22oss % 3APutObjec
 t % 22 % 5D % 2C % 22Resource % 22 % 3A % 5B % 22acs % 3Aoss % 3A
 % 2A % 3A % 2A % 3Aram - test - app % 2Fusr001 % 2F % 2A % 22 % 5D
 % 7D % 5D % 7D & SignatureM  ethod = HMAC - SHA1 & Version = 2015
 - 04 - 01 & Signature = Y0OPUoL1Pr  CqX4X6A3 % 2FJvgXuS6c % 3D &
 Action = AssumeRole & SignatureN  once = 8d0798a8 - 8cd9 - 11e5 -
 9f49 - 008cfa5e49  38
 {
   " AssumedRol  eUser ": {
     " Arn ": " acs : ram :: 1894xxxxxx  722283 : role / ramtestapp
 write / usr001 ",
     " AssumedRol  eId ": " 3554078476  60029428 : usr001 "
   },
   " Credential  s ": {
     " AccessKeyI  d ": " STS . rtfx13 ****** NlIJlS4U ",
     " AccessKeyS  ecret ": " 2fsaM8E2ma  B2dn ****** wpsKTyK4aj
 o7TxFr0zIM ",
     " Expiration ": " 2015 - 11 - 17T04 : 16 : 10Z ",
     " SecurityTo  ken ": " CAESkwMIAR  KAAUh3 / Uzcg13 ******
 y0IZjGewMp  g31ITxCleB  FU1eO / 3Sgpudid + GVs + Olvu1vXJn ******
 a8azKJKtzV  0oKSy + mwUrxSvUSR  VDntrs78Cs  NfWoOJUMJK  jLIxdWnGi1
 pgxJCBzNZ2  YV / 6ycTaZySSE  1V6kqQ7A + GPwY ****** LpdGhhTVFM
 ucnRmeDEzR  FlNVWJjTmx  JSmxTNFUiE  jM1NTQwNzg  0NzY2MDAyO
 TQyOCoGdXN  yMDAxMOPzo  J2RKjoGUnN  hTUQ1QnYKA  TEacQoFQWx
 sb3cSJwoMQ  WN0aW9uRXF  1YWxzEgZBY  3Rpb24aDwo  Nb3NzOlB1d
 E9iamVjdBI / Cg5SZXNvdX  JjZUVxdWFs  cxIIUmVzb3  VyY2UaIwoh
 YWNzOm9zcz  oqOio6cmFt  LXRlc3Qt ****** VzcjAwMS8q  ShAxODk0MT
 g5NzY5NzIy  MjgzUgUyNj  g0MloPQXNz  dW1lZFJvbG  VVc2VyYABq
 EjM1NTQwNz  g0NzY2MDAy  OTQyOHIPcm  FtdGVzdGFw  cHdyaXRl "
   },
   " RequestId ": " 19407707 - 54B2 - 41AD - AAF0 - FE87E8870B  0D "
```

```
 }
```

2. Let us test if we can use the credentials to upload and download.

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $./ osscmd   get   oss :// ram - test - app / test . txt   test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   STS .
 rtfx13 ****** NlIJlS4U  - k   2fsaM8E2ma  B2dn ****** wpsKTyK4aj
 o7TxFr0zIM  -- sts_token = CAESkwMIAR  KAAUh3 / Uzcg13 ******
 y0IZjGewMp  g31ITxCleB  FU1eO / 3Sgpudid + GVs + Olvu1vXJn ******
 a8azKJKtzV  0oKSy + mwUrxSvUSR  VDntrs78Cs  NfWoOJUMJK  jLIxdWnGi1
 pgxJCBzNZ2  YV / 6ycTaZySSE  1V6kqQ7A + GPwY ****** LpdGhhTVFM
 ucnRmeDEzR  FlNVWJjTmx  JSmxTNFUiE  jM1NTQwNzg  0NzY2MDAyO
 TQyOCoGdXN  yMDAxMOPzo  J2RKjoGUnN  hTUQ1QnYKA  TEacQoFQWx
 sb3cSJwoMQ  WN0aW9uRXF  1YWxzEgZBY  3Rpb24aDwo  Nb3NzOlB1d
 E9iamVjdBI / Cg5SZXNvdX  JjZUVxdWFs  cxIIUmVzb3  VyY2UaIwoh
 YWNzOm9zcz  oqOio6cmFt  LXRlc3Qt ****** VzcjAwMS8q  ShAxODk0MT
 g5NzY5NzIy  MjgzUgUyNj  g0MloPQXNz  dW1lZFJvbG  VVc2VyYABq
 EjM1NTQwNz  g0NzY2MDAy  OTQyOHIPcm  FtdGVzdGFw  cHdyaXRl
  Error   Headers :
 [(' content - length ', ' 254 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 564A9C31FF  FC811F24B6  E7E3 '), (' date ', ' Tue ,  17   Nov
 2015   03 : 17 : 05   GMT '), (' content - type ', ' applicatio  n
 / xml ')]
  Error   Body :
 <?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
 < Error >
   < Code > AccessDeni  ed </ Code >
   < Message > Access   denied   by   authorizer ' s   policy .</
 Message >
   < RequestId > 564A9C31FF  FC811F24B6  E7E3 </ RequestId >
   < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs .
 com </ HostId >
 </ Error >
  Error   Status :
  403
  get   Failed !
 [ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $./ osscmd   put   test . txt   oss :// ram - test - app / test .
 txt  -- host = oss - cn - hangzhou . aliyuncs . com  - i   STS .
 rtfx13 ****** NlIJlS4U  - k   2fsaM8E2ma  B2dn ****** wpsKTyK4aj
 o7TxFr0zIM  -- sts_token = CAESkwMIAR  KAAUh3 / Uzcg13 ******
 y0IZjGewMp  g31ITxCleB  FU1eO / 3Sgpudid + GVs + Olvu1vXJn ******
 a8azKJKtzV  0oKSy + mwUrxSvUSR  VDntrs78Cs  NfWoOJUMJK  jLIxdWnGi1
 pgxJCBzNZ2  YV / 6ycTaZySSE  1V6kqQ7A + GPwY ****** LpdGhhTVFM
 ucnRmeDEzR  FlNVWJjTmx  JSmxTNFUiE  jM1NTQwNzg  0NzY2MDAyO
 TQyOCoGdXN  yMDAxMOPzo  J2RKjoGUnN  hTUQ1QnYKA  TEacQoFQWx
 sb3cSJwoMQ  WN0aW9uRXF  1YWxzEgZBY  3Rpb24aDwo  Nb3NzOlB1d
 E9iamVjdBI / Cg5SZXNvdX  JjZUVxdWFs  cxIIUmVzb3  VyY2UaIwoh
 YWNzOm9zcz  oqOio6cmFt  LXRlc3Qt ****** VzcjAwMS8q  ShAxODk0MT
 g5NzY5NzIy  MjgzUgUyNj  g0MloPQXNz  dW1lZFJvbG  VVc2VyYABq
 EjM1NTQwNz  g0NzY2MDAy  OTQyOHIPcm  FtdGVzdGFw  cHdyaXRl
  100 %  Error   Headers :
 [(' content - length ', ' 254 '), (' server ', ' AliyunOSS '), ('
 connection ', ' keep - alive '), (' x - oss - request - id ', '
 564A9C3FB8  DE437A91B1  6772 '), (' date ', ' Tue ,  17   Nov
 2015   03 : 17 : 19   GMT '), (' content - type ', ' applicatio  n
 / xml ')]
  Error   Body :
 <?  xml   version =" 1 . 0 "  encoding =" UTF - 8 "? >
 < Error >
   < Code > AccessDeni  ed </ Code >
```

```
  < Message > Access  denied  by  authorizer ' s  policy .</
Message >
  < RequestId > 564A9C3FB8  DE437A91B1  6772 </ RequestId >
  < HostId > ram - test - app . oss - cn - hangzhou . aliyuncs .
com </ HostId >
</ Error >
 Error  Status :
 403
 put  Failed !
```

The test.txt upload fails. We have formatted the entered policy discussed at the beginning of this document, which is as follows:

```
{
     " Version ": " 1 ",
         " Statement ": [
         {
             " Effect ": " Allow ",
             " Action ": [
                 " oss : PutObject "
                 ],
             " Resource ": [
                 " acs : oss :*:*: ram - test - app / usr001 /*"
                 ]
         }
     ]
 }
```

This policy indicates that users are only allowed to upload files like usr001/ to the ram-test-app bucket. If the app user is usr002, the policy can be changed to only allow for the uploading of files like usr002/. By setting different policies for different app users, we can isolate the storage space of different app users.

3. Retry the test and specify the upload destination as ram-test-app/usr001/test.txt.

```
[ admin @ NGIS - CWWF344M01  C  / home / admin / oss_test ]
 $./ osscmd  put  test . txt  oss :// ram - test - app / usr001
 / test . txt  -- host = oss - cn - hangzhou . aliyuncs . com  -
 i  STS . rtfx13 ****** NlIJlS4U  - k  2fsaM8E2ma  B2dn ******
 wpsKTyK4aj  o7TxFr0zIM  -- sts_token = CAESkwMIAR  KAAUh3 / Uzcg13
 ****** y0IZjGewMp  g31ITxCleB  FU1eO / 3Sgpudid + GVs + Olvu1vXJn
 ****** a8azKJKtzV  0oKSy + mwUrxSvUSR  VDntrs78Cs  NfWoOJUMJK
 jLIxdWnGi1  pgxJCBzNZ2  YV / 6ycTaZySSE  1V6kqQ7A + GPwY ******
 LpdGhhTVFM  ucnRmeDEzR  FlNVWJjTmx  JSmxTNFUiE  jM1NTQwNzg
 0NzY2MDAyO  TQyOCoGdXN  yMDAxMOPzo  J2RKjoGUnN  hTUQ1QnYKA
 TEacQoFQWx  sb3cSJwoMQ  WN0aW9uRXF  1YWxzEgZBY  3Rpb24aDwo
 Nb3NzOlB1d  E9iamVjdBI / Cg5SZXNvdX  JjZUVxdWFs  cxIIUmVzb3
 VyY2UaIwoh  YWNzOm9zcz  oqOio6cmFt  LXRlc3Qt ****** VzcjAwMS8q
 ShAxODk0MT  g5NzY5NzIy  MjgzUgUyNj  g0MloPQXNz  dW1lZFJvbG
 VVc2VyYABq  EjM1NTQwNz  g0NzY2MDAy  OTQyOHIPcm  FtdGVzdGFw
 cHdyaXRl
  100 %
  Object  URL  is :  http :// ram - test - app . oss - cn -
 hangzhou . aliyuncs . com / usr001 % 2Ftest . txt
  Object  abstract  path  is :  oss :// ram - test - app / usr001
 / test . txt
  ETag  is  " 946A0A1AC8  245696B9C6  A6F3594269  0B "
```

```
0 . 071 ( s )  elapsed
```

The upload is successful.

Summary

This section describes how to grant users temporary access authorization for OSS using STS. In typical mobile development scenarios, STS can be used to grant temporary authorizations to access OSS when different app users need to access the app. The temporary authorization can be configured with expiration time to greatly reduce the hazards caused by leaks. When obtaining temporary authorization, we can enter different authorization policies for different app users to restrict their access permissions. For example, to restrict the object paths accessible to users. This isolates the storage space of different app users.

20.1.3 FAQ about subaccount settings

How to create an STS temporary account and how to use it to access resources?

See *STS temporary access authorization*.

Client or console logon error reported for an authorized sub-account

See *Why does a sub-account encounters an error of no operation permission for a bucket on the OSS console after it has been granted the bucket operation permission*.

How to authorize a sub-account with the operation permission for a single bucket

See *How to assign the full operation permission for a specified bucket to a sub-account*.

How to authorize a sub-account with the operation permission for a directory in a bucket

See *OSS directory authorization*

How to authorize a sub-account with the read-only permission for a bucket

See *Authorize a sub-user to list and read resources in a bucket*.

Error upon an OSS SDK call: InvalidAccessKeyId

See *STS errors and troubleshooting*.

Error upon an STS call: Access denied by authorizer’s policy

Detailed error information: ErrorCode: AccessDenied ErrorMessage: Access denied by authorizer’s policy.

Cause of the error:

- The temporary account has no access permission.
- The authorization policy specified for assuming the role of this temporary account does not assign the access permission to the account.

For more STS errors and the causes, see *OSS permission errors and troubleshooting*.