Object Storage Service

Developer Guide

Developer Guide

The following table lists the documents that will help you fully utilize OSS:

Resource	Description
Alibaba Cloud OSS Developer Guide	Describes the core concepts, functions, and operation procedures of OSS, as well as examples about how to use APIs and SDKs.
Alibaba Cloud OSS Console User Guide	Describes all operations supported by the OSS Console.
Alibaba Cloud OSS Best Practice	Describes the application scenarios and configuration practices of OSS.
Alibaba Cloud OSS API Manual	Describes the RESTful API operations supported by OSS and provides related examples.
Alibaba Cloud OSS SDK Manual	Describes the SDK development and related parameters based on major languages.
Alibaba Cloud OSS Image Processing Guide	Describes various functions provided by Image Service.
Alibaba Cloud OSS Migration Tool	The migration tool can synchronize your files stored locally or in the third-party cloud to OSS.

Basic OSS concepts

This section introduces the basic concepts of OSS.

Object

An object (also known as a file) is a discrete unit of data.

An object is composed of:

- Metadata, known as Object Meta, which is a key-value pair that expresses the object’ s attributes, such as its last modification time and size, as well as user-defined information.
- User data, known as Data.
- A unique object name, known as a Key.

The size of an object varies with the upload method. Multipart Upload supports objects of up to 48.8 TB. Other upload methods only support objects of up to 5 GB.

An object’ s lifecycle starts from when it has been successfully uploaded, and ends when it has been deleted. During an object’ s lifecycle, its information cannot be changed. If you upload an object with a duplicate name in a bucket, it will overwrite the existing one. Therefore, unlike the file system, OSS does not allow users to modify only part of an object.

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

The name of an object must comply with the following rules:

- It must use UTF-8 encoding.
- It must be between 1-1023 bytes in length.
- It cannot start with a backslash “/ “ or forward slash “\” .

NOTE: Object names are case sensitive.

Bucket

A bucket is a virtual division of object storage that, unlike file systems, manages objects in a flat structure.

Bucket properties are as follows:

- All objects must belong to a bucket, and during an object’ s lifecycle it remains directly affiliated with the corresponding bucket.
- A user can have multiple buckets, with each bucket able to contain an unlimited number of objects.
- You can set and modify the attributes of a bucket for region and object access control and object lifecycle management. These attributes apply to all objects in the bucket.
- You can create different buckets to perform different management functions.

The name of a bucket must comply with the following rules:

- It can only contain lower-case letters, digits, and hyphens (-).
- It must start and end with a lower-case letter or number.
- It must be between 3-63 bytes in length.
- It must be globally unique within the OSS.

Once a bucket name is created, it cannot be changed.

Region

A region represents the physical location of an OSS data center.

Users can select regions based on fees, request sources, and other factors according to site requirements. Generally, the closer the user is in proximity to a region, the faster the access speed will be. For more details, refer to OSS Regions and Endpoints.

A region is specified when a bucket is created, and cannot be changed. All objects contained in this bucket are therefore stored in the corresponding data center. Setting different regions for objects in the same bucket is currently not supported.

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:

- The intranet endpoint is oss-cn-hangzhou-internal.aliyuncs.com
- The Internet endpoint is oss-cn-hangzhou.aliyuncs.com

For more details, refer to OSS Regions and Endpoints.

AccessKey

An AccessKey (AK) is composed of an AccessKeyId and AccessKeySecret. The AccessKeyId is a public key, whereas the AccessKeySecret is a private key and must be kept confidential. These are then paired together to perform access identity verification.

The OSS verifies the identity of a request sender by using the AccessKeyId/AccessKeySecret symmetric encryption method. The AccessKeyId identifies a user. With the AccessKeySecret, a user can encrypt the signature string. The OSS then uses the symmetric encryption method to verify the AccessKey of the signature string. In OSS, AccessKeys are generated as follows:

- Applied for by the bucket owner.
- Granted by the bucket owner to an authorized third-party requestor through RAM.
- Granted by the bucket owner to an authorized third-party requestor through STS.

For more information about AccessKeys, see RAM.

High consistency

In OSS, object operations are binary, that is, operations must either succeed or fail without an intermediate status. After a user uploads an object, OSS ensures that it is complete. OSS will not return a partial success response when uploading objects.

Object operations in OSS are likewise highly consistent. For example, once a user receives an upload

(PUT) success response, this object can be read immediately, and the data will have already been written in triplicate. The same concept applies to delete operations. Once a user deletes an object, this object no longer exists.

This high-consistency feature facilitates user architectural design. The logic of OSS usage is the same as that of a traditional storage device: modifications are immediately visible and users do not have to consider final consistency issues.

Comparison between OSS and file system

OSS is a distributed object storage service structure that uses a Key-Value pair format, whereas a file system uses a tree-type index structure of directories that contain files. In OSS, users retrieve object content based on unique object names (Keys). In file systems, users retrieve files based on their location in a directory.

The benefit of OSS is that it supports massive concurrent access volumes, which means large volumes of unstructured data (such as images, videos, and documents) can be stored and retrieved without excessive use of resources. The benefit of a file system is that folder operations such as renaming, moving, and deleting directories (which means renaming, moving, and deleting data) is considerably easier as data does not need to be copied and replaced.

The limitation of OSS is that saved objects cannot be modified. If an object needs modification, the entire object must be uploaded again to make the modification take effect. One exception is through using the append object operation, whereby users call a specific API, which allows a generated object be of a different type than normally uploaded objects. The limitations of a file system are that system performance is limited to a single device, and the more files and directories that are created in the system, the more resources are consumed, and the lengthier user processes become.

Comparisons between OSS and file system concepts are as follows:

OSS	File system
Object	File
Bucket	Main directory
Region	N/A
Endpoint	N/A
AccessKey	N/A
N/A	Multilevel directory
GetService	Retrieving the list of main directories
GetBucket	Retrieving the list of files
PutObject	Writing an object
AppendObject	Append writing an object
GetObject	Reading an object

DeleteObject	Deleting an object
N/A	Modifying file content
CopyObject (same target and source)	Modifying file attributes
CopyObject	Copying an object
N/A	Renaming an object

OSS Glossary

Term	Definition
Object	A discrete unit of data
Bucket	A virtual division of object storage
Endpoint	The domain name for OSS access
Region	A representation of the physical location of an OSS data center
AccessKey	An alias for the AccessKeyId and AccessKeySecret pair
Put Object	Simple upload
Post Object	Form upload
Multipart Upload	The uploading of an object as several chunks, then reassembling the chunks
Append Object	An upload that attaches to already uploaded data
Get Object	Simple download
Callback	Upload callback
Object Meta	Metadata of a file that includes the object’ s attributes and user-defined information
Data	Object information (typically user-defined information)
Key	Unique object name
ACL (Access Control List)	Permissions for buckets or files

Storage classes

Introduction to storage classes

OSS provides three storage classes: Standard, Infrequent Access (IA), and Archive, applicable to various hot and cold data storage scenarios.

- The Standard class provides common object storage services. It is suitable for the storage of audios and videos, pictures, and website static resource frequently accessed via WiFi hotspots. It supports high-throughput computing scenarios and is suitable for the storage of computing resources.
- The IA class is suitable for data that will be stored for long and infrequently accessed and applicable to backup of mobile applications, smart devices, and enterprise data. It supports real-time data access.
- The Archive class has the lowest unit price among the three storage classes. It is suitable for archive data, medical imaging, scientific data, and video materials to be stored for long and can effectively optimize the long-term storage costs. The restoration of data stored in the Archive class to the readable status takes one minute.

Standard

The Standard class provides object storage services featuring high reliability, availability, and performance, and supports frequent data access. The high-throughput and low-latency service response capability of OSS can effectively support access to hotspot data. The Standard class is the right choice for social, picture sharing, audio and video applications, large sites, and big data analysis scenarios.

Key features:

- Data reliability up to 99.99999999%
- Service availability up to 99.95%
- High-throughput and low-latency access performance
- HTTPS encryption transmission
- Picture processing

IA

The IA class is suitable for data that will be stored for long and infrequently accessed. Its unit price is lower than that of the Standard class. It is applicable to long-term backup of mobile applications, smart devices, and enterprise data. Objects of the IA class have the minimum storage duration. Earlier deletion or overwriting of files with the storage duration of less than 30 days will incur costs. Objects of the IA class have the minimum storage space. For an object with the size of less than 128 KB, the

minimum storage space is 128 KB. Data acquisition will incur costs.

Key features:

- Data reliability up to 99.99999999%
- Service availability up to 99.9%
- Real-time access
- HTTPS encryption transmission
- Picture processing
- Minimum storage duration and minimum storage space

Archive

The unit price of the Archive class is the lowest among the three storage class. It is suitable for archive data that will be stored for long (more that half a year is recommended) and seldom accessed during storage. The restoration of data to the readable status takes one minute. It is suitable for archive data, medical imaging, scientific data, and video materials to be stored for long. Objects of the Archive class have the minimum storage duration. Earlier deletion or overwriting of files with the storage duration of less than 30 days will incur costs. Objects of the Archive class have the minimum storage space. For an object with the size of less than 128 KB, the minimum storage space is 128 KB. Data acquisition will incur costs.

Key features:

- Data reliability up to 99.99999999%
- Service availability up to 99.9%
- It takes one minute to restore stored data from the archived status to the readable status.
- HTTPS encryption transmission
- Image processing is not supported.
- Minimum storage duration and minimum storage space

Comparison of storage classes

Comparison indicator	Standard	IA	Archive
Data reliability	99.99999999%	99.99999999%	99.99999999%
Designed for availability	99.95%	99.9%	99.9%
Minimum storage space of objects	Actual size of objects	128 KB	128 KB
Minimum storage duration	Not required	30 days	30 days
Data acquisition costs	Not collected	Collected based on the acquired data per GB	Collected based on the acquired data per GB

Data access latency	Milliseconds	Milliseconds	1 minute Data needs to be restored first, and then can be accessed.
Picture processing	Supported	Supported	Not supported

Note: Data in "data acquisition costs" refers to the volume of data read from the underlying distributed storage system. The volume of data transferred on the public network is included in billing items of the outbound traffic.

Access and control

Endpoints

Composition rules for domain names

With regards to all network requests for the OSS, except those for the GetService API, the domain names are third-level domain names for specific buckets. The domain name is composed of the bucket name and the endpoint: BucketName.Endpoint. Here, **Endpoint** can vary due to the region (data center) of the bucket and the intranet/Internet access method.

Endpoint naming rules for an external network

Here, external network refers to the Internet.

```
Region + .aliyuncs.com
```

Endpoint naming rules for an internal network

Here, internal network refers to Alibaba Cloud's intranet.

```
Region + -internal + .aliyuncs.com
```

OSS regions and endpoints

Refer to Regions and endpoints.

Endpoint settings in the OSS SDK

For each user operation, the OSS SDK has a spliced endpoint. However, users need to set different endpoints when configuring buckets of different regions.

For example, when using a Java SDK, users need to set the endpoint during class instantiation before configuring buckets in the following Hangzhou region:

```
String accessKeyId = "<key>";
String accessKeySecret = "<secret>";
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

Use intranet endpoints to access OSS in ECS

Intranet addresses can be used for access between an ECS instance and an OSS instance that are in the same region.

For example, a user has:

- An ECS instance located in Beijing
- An OSS bucket which is named beijingres and is located in Beijing
- An OSS bucket which is named qingdaores and is located in Qingdao
- Based on the preceding information:
- The user can access resources in beijingres through the intranet address ‘beijingres.oss-cn-beijing-internal.aliyuncs.com’ .
- The user cannot access qingdaores through the intranet address qingdaores.oss-cn-qingdao. They must instead access the OSS through the Internet address qingdaores.oss-cn-qingdao.aliyuncs.com.

In the preceding sample Java SDK, the Internet address of the bucket is used for OSS access. To access the OSS through the intranet, you only need to modify the endpoint:

```
String accessKeyId = "<key>";
String accessKeySecret = "<secret>";
String endpoint = "http://oss-cn-hangzhou-internal.aliyuncs.com";
OSSClient client = new OSSClient(endpoint, accessKeyId, accessKeySecret);
```

OSS access

OSS access URLs

OSS is an object storage service based on HTTP APIs. For all operations, users need to specify the OSS resource to access. This resource may be a bucket or an object. During OSS access, the OSS resource is expressed as a URL that indicates third-level domain name access, which is formatted as follows:

```
<Schema>://<Bucket>.<Endpoint>/<Object> Third-level domain name access method
```

A description of the URL parameters is as follows:

- Schema: value of HTTP or HTTPS
- Bucket: the user’ s OSS storage space
- Endpoint: the access domain name for a bucket’ s data center
- Object: an object uploaded by a user to the OSS

Notes

- When the resource is a bucket, the endpoint must be consistent with the region containing the bucket. For example, if a bucket is created in Hangzhou, the Hangzhou endpoint must be used. Endpoints for other regions cannot be used.
- To access the OSS, ECS instances can use the intranet endpoint for OSS resources in the same region.
- For a list of regions and their endpoints, refer to Regions and endpoints.

If a user uses HTTPS to send a request to the Hangzhou OSS for an object named mytest/oss-test-object in a bucket named oss-sample, the third-level domain name is as follows:

```
https://oss-sample.oss-cn-hangzhou.aliyuncs.com/mytest/oss-test-object
```

Users can directly use object URLs in HTML, as shown below:

```
<img src="https://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png" />
```

OSS access security

HTTP requests sent to the OSS are divided into two types depending on whether they include identity authentication information: Requests with identity verification information, and anonymous requests without identity verification information. The identity verification information in requests can be

structured in two ways:

- Authorization is contained in the request header, in the format: OSS + AccessKeyId + signature string.
- OSS AccessKeyId and signature fields are contained in the request URL.

OSS access verification process

Anonymous request access process

1. The user’ s request is sent to the OSS’ s HTTP server.
2. OSS resolves the URL to get the bucket and object.
3. OSS checks whether ACL is set for the object.
    - If no, go to Step **4**.
    - If yes, OSS then checks whether the object’ s ACL permits anonymous access.
        - If yes, go to step **5**.
        - If no, the request is rejected and the process ends.
4. OSS checks whether the bucket’ s ACL permits anonymous access.
    - If no, an error message is returned and the process ends.
    - If yes, go to step **5**.
5. The request passes permission verification and the object content is returned to the user.

Access process for requests with ID verification information

1. The user’ s request is sent to the OSS’ s HTTP server.
2. The OSS resolves the URL to get the bucket and object.
3. Based on the request’ s OSS AccessKeyId, the OSS retrieves the ID information of the requestor for authentication.
    - If the ID information cannot be obtained, an error message is returned and the process ends.
    - If the ID information is obtained, but the client is not permitted to access this resource, an error message is returned and the process ends.
    - If the ID information is obtained, but the signature calculated based on the request’ s HTTP parameters does not match the sent signature, an error message is returned and the process ends.
    - If the authentication succeeds, go to step **4**.
4. OSS checks whether ACL is set for the object.
    - If no, go to Step **5**.
    - If yes, OSS then checks whether the object’ s ACL permits anonymous access.
        - If yes, go to step **6**.
        - If no, the request is rejected and the process ends.
5. OSS checks whether the bucket’ s ACL permits anonymous access.
    - If yes, go to step **6**.

- If no, an error message is returned and the process ends.

6. The request passes permission verification and the object content is returned to the user.

Methods for OSS access using ID verification information

- Console
  On the console, the identity verification process is concealed from users. When users access the OSS through the console, they do not have to concern about the details of this process.
- SDKs
  The OSS provides SDKs for multiple development languages. A signature algorithm is implemented in an SDK, where users only need to input the AK information as a parameter.
- APIs
  If you want to write your own code to package a call to the RESTful API, you need to implement a signature algorithm to calculate the signature. For details about the signature algorithm, refer to Adding a signature to the header and Adding a signature to the URL.

Note: For an explanation of AccessKeys, as well as more information on identity authentication operations, refer to RAM.

Bind custom domain names (CNAME)

Users can bind custom domain names (CNAMEs) to their buckets. This operation must be performed through the OSS Console, and is available only when a user applies for an ICP license for the bound domain name and obtains permission from Alibaba Cloud. After the CNAME function is activated, the OSS will automatically process access requests on that domain name.

CNAME application example

1. User A has the website abc.com which contains a page with the link http://img.abc.com/logo.png. The image img.abc.com needs to be migrated to the OSS.
2. Through the OSS Console, user A submits an application to bind the user-defined domain name img.abc.com to abc-img, and provides the associated materials for CNAME function approval.
3. After Alibaba Cloud approves the application, the OSS background will map img.abc.com onto abc-img (permission verification will be performed at this time).
4. User A, using their own domain name server, then adds a CNAME rule, mapping img.abc.com onto abc-img.oss-cn-hangzhou.aliyuncs.com. This means all access traffic to the user’ s img.abc.com domain name will be forwarded to abc-img.oss-cn-hangzhou.aliyuncs.com on the OSS.
5. Once a request for http://img.abc.com/logo.png reaches the OSS, the OSS will locate the

img.abc.com and abc-img mapping and convert the request to an access request for the abc-img bucket. When a user attempts to access http://img.abc.com/logo.png, after passing through the OSS, the website accessed is http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo.png.

CNAME process comparisons

Without bound CNAME:

1. A request to access http://img.abc.com/logo.png is received.
2. DNS resolves to the user’ s server IP.
3. Access to logo.png on the user’ s server is achieved.

With bound CNAME:

1. A request to access http://img.abc.com/logo.png is received.
2. DNS resolves to abc-img.oss-cn-hangzhou.aliyuncs.com.
3. Access to logo.png in the OSS bucket abc-img is achieved.

Reference

- Console: Domain Name Management

Access control

Send an OSS access request

You can access the OSS directly by calling a RESTful API provided by the OSS or using an API-encapsulated SDK. Each request for access to the OSS requires identity verification or direct anonymous access based on the current bucket permission and operation.

According to the roles of visitors, the access to OSS resources is divided into owner access and third-party access. Here, the owner refers to the bucket owner, also known as “developer”. Third-party users are users who access resources in a bucket.

According to the identity of visitors, the access to OSS resources is divided into anonymous access and signature-based access. In the OSS, a request that does not contain any identification information is considered anonymous access. Signature-based access refers to requests that, according to the rules in the OSS API documentation, contain signature information in the request header or URL.

Types of AccessKeys

Currently, there are three types of AccessKey for OSS access. They are described below:

Alibaba Cloud account AccessKeys

These are the AccessKeys of bucket owners. The AccessKey provided by each Alibaba Cloud account has full access to its own resources. Each Alibaba Cloud account can simultaneously have 0 to 5 active or inactive AccessKey pairs (AccessKeyID and AccessKeySecret). You can add or delete AccessKey pairs on AccessKey console. Each AccessKey pair may be in two states: active and inactive.

- Active indicates that the user’ s AccessKey is in the active state and can be used for identity authentication.
- Inactive indicates that the user’ s AccessKey is in the inactive state and cannot be used for identity authentication.

The AccessKey of the Alibaba Cloud account should not be directly used unless necessary.

RAM account AccessKeys

Resource Access Management (RAM) is a resource access control service provided by Alibaba Cloud. RAM account AKs are the access keys granted by RAM. These AKs only allow access to resources in a bucket according to the rules defined by RAM. RAM helps you to collectively manage your users (such as employees, systems or applications) and controls which resources your users can access. For example, you can allow your users to have only the read permission on a bucket.
Subaccounts are subordinate to normal accounts and cannot own any actual resources. All resources belong to primary accounts.

STS account AccessKeys

The Alibaba Cloud STS (Security Token Service) is a service that provides temporary access credentials. STS account AKs are the AKs issued by the STS. These AKs only allow access to buckets in accordance with the rules defined by the STS.

Implementation of identity authentication

Currently, there are three methods of authentication:

- AK authentication
- RAM authentication
- STS authentication

Before sending a request to the OSS as an individual identity, a user needs to generate a signature string for the request according to the format specified by the OSS and then encrypt the signature

string using the AccessKeySecret to generate a verification code.
After receiving the request, the OSS finds the corresponding AccessKeySecret based on the AccessKeyID, and obtains the signature string and verification code in the same way. If the obtained verification code is the same as the provided verification code, the request is assumed valid. If not, the OSS rejects the request and returns an HTTP 403 error.
Users can directly use the SDKs provided by the OSS with different types of AccessKeys for different types of identity authentication.

Permission control

OSS provides various permission control mechanisms for access to its stored objects:

- Bucket-level permissions
- Object-level permissions
- Account-level permissions (RAM)
- Temporary account permissions (STS)

Bucket-level permissions

Bucket permission types

The OSS provides an Access Control List (ACL) for permission control. The OSS ACL provides bucket-level access control. Currently, three access permissions are provided for a bucket: public-read-write, public-read, and private. They are described as follows:

Permission	Access Restriction
Public-read-write	Anyone (including anonymous users) can read, write, and delete the objects in the bucket. The fees incurred by such operations shall be borne by the owner of the bucket. **Use this permission with caution**.
Public-read	Only the owner of a bucket can write or delete the objects in the bucket. Anyone (including anonymous users) can read the objects in the bucket.
Private	Only the owner of a bucket can read, write, and delete the objects in the bucket. Others cannot access the objects in the bucket without authorization.

Bucket permission settings and read methods

Function usage reference:

- API: Put BucketACL
- SDK: Java SDK-Set Bucket ACL
- Console: Create Bucket Permission Setting
- API: Get BucketACL
- SDK: Java SDK-Obtain Bucket ACL

Object-level permissions

Object permission types

The OSS ACL also provides object-level permission access control. Currently, four access permissions are available for an object, including private, public-read, public-read-write and default. You can use the "x-oss-object-acl" header in the Put Object ACL request to set the access permission. Only the bucket owner has the permission to perform this operation.

Permission	Access restriction
public-read-write	Indicates that the object can be read and written by the public. That is, all users have the permission to read and write the object.
public-read	Indicates that the object can be read by the public. Only the owner of the object has the permission to read and write the object. Other users only have the permission to read the object.
private	Indicates that the object is a private resource. Only the owner of the object has the permission to read and write the object. Other users have no permission to operate the object.
default	Indicates that the object inherits the permission of the bucket.

Considerations

- 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 has higher-level permission than the bucket ACL. For example, an object with the public-read permission can be accessed by authenticated users and anonymous users, regardless of the bucket permission.

Object permission settings and read methods

Function usage reference:

- API: Put Object ACL
- SDK: Java SDK-Set the object ACL in ObjectACL
- API: Get Object ACL
- SDK: Java SDK-Read the object ACL from ObjectACL

Account-level permissions (RAM)

Application scenarios

If you have purchased cloud resources and multiple users in your organization need to use them, these users have to share the AccessKey of your Alibaba Cloud account. There are two problems:

- If your key is shared by many people, it has a high risk of leakage.
- You cannot determine which resources (e.g. buckets) can be accessed by the users.

Solution: Under your Alibaba Cloud account, you can use RAM to create subusers with their own AccessKeys. In this case, your Alibaba Cloud account will be the primary account and the created accounts will be subaccounts. Subaccounts can only use their AccessKeys for the operations and resources authorized by the primary account.

Specific implementation

For details about the RAM, refer to RAM User Guide. The RAM User Guide describes how to grant permissions, create RAM accounts, and manage group permissions in details.

For details about how to configure the policies required in authorization, refer to the final section Configuration Rules of this chapter.

Temporary account permissions (STS)

Application scenarios

Users managed by your local identity system, such as your app users, your local corporate account, or third-party apps, may also directly access OSS resources. They are called federated users. In addition, users can also be the applications you create that have access to your Alibaba Cloud resources.

With respect to these federated users, short-term access permission management is provided for the Alibaba Cloud account (or RAM users) through the Security Token Service (STS) of Alibaba Cloud. You do not need to reveal the long-term key (such as the login password and AccessKey) of your Alibaba Cloud account (or RAM users), but only need to create a short-term access credential for a federated user. The access permission and validity of this credential are both up to you. You do not need to care about permission revocation. The access credential automatically becomes invalid when it expires.

STS-based access credentials include the security token (SecurityToken) and the temporary access key (AccessKeyId and AccessKeySecret). The AccessKey method is the same as the method of using the AccessKey of the Alibaba Cloud account or RAM user. In addition, each OSS access request must carry a security token.

Specific implementation

For details about the STS, refer to Role management in the RAM User Guide. The key is to call AssumeRole of the STS interface to obtain valid access credential. You can also directly use STS SDK to call the access credential.

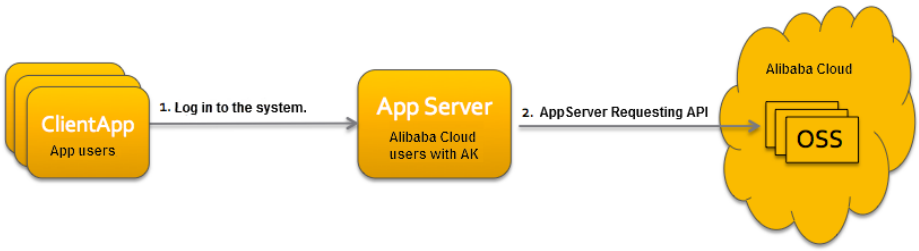
For details about how to configure the policies required in authorization, refer to the final section of this chapter.

RAM and STS application scenario practices

In different application scenarios, how the access identity is verified may vary. The following describes two methods for access identity verification in typical application scenarios.

A mobile app is used as an example. Assume that you are a mobile app developer. You attempt to use the Alibaba Cloud OSS to store end user data of the app. You also have to ensure data is isolated between app users to prevent an app user from obtaining data of other app users.

Mode 1: Using AppServer for data transit and data isolation

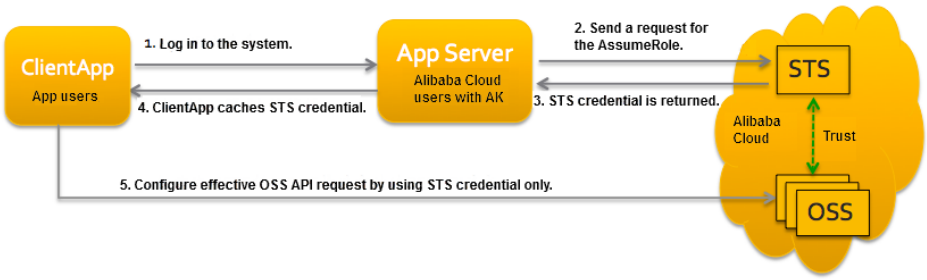


As shown in the figure above, you need to develop an AppServer. Only the AppServer can access the ECS. The ClientApp can read or write data only through the AppServer. The AppServer ensures isolated access to different user data.

In this method, you can use the key provided by your Alibaba Cloud account or RAM account for signature verification. In case of any security problem, you are recommended not to directly use the key of your Alibaba Cloud account (root account) to access the OSS.

Mode 2: Using STS for direct access to OSS

The STS solution is shown below:



The solution is described in detail as follows:

1. Log in as the app user. The app user is irrelative to the Alibaba Cloud account but is an end user of the app. The AppServer allows the app user to log in. For each valid app user, the AppServer needs to define the minimum access permission for them.
2. The AppServer requests a security token (SecurityToken) from the STS. Before calling STS, the AppServer needs to determine the minimum access permission (described in policy syntax) of app users and the expiration time of the authorization. Then, the AppServer uses AssumeRole to obtain a security token indicating a role. For details about role management and usage, refer to Role Management in the RAM User Guide.
3. The STS returns a valid access credential to the AppServer, where the access credential includes a security token, a temporary access key (AccessKeyID and AccessKeySecret), and the expiry time.
4. The AppServer returns the access credential to the ClientApp. The ClientApp caches this credential. When the credential becomes invalid, the ClientApp needs to request a new valid access credential from the AppServer. For example, if the access credential is valid for one hour, the ClientApp can request the AppServer to update the access credential every 30 minutes.
5. The ClientApp uses the access credential cached locally to request Alibaba Cloud Service APIs. The ECS perceives the STS access credential, relies on STS to verify the credential, and correctly responds to the user request.

RAM and STS authorization policy configuration

The detailed rules of the use of policies during RAM or STS authorization are as follows.

Example

First, let’ s look at the following policy example:

```
{
"Version": "1",
"Statement": [
{
```

```
"Action": [
"oss:GetBucketAcl",
"oss:ListObjects"
],
"Resource": [
"acs:oss:*:1775305056529849:mybucket"
],
"Effect": "Allow",
"Condition": {
"StringEquals": {
"acs:UserAgent": "java-sdk",
"oss:Prefix": "foo"
},
"IpAddress": {
"acs:SourceIp": "192.168.0.1"
}
}
},
{
"Action": [
"oss:PutObject",
"oss:GetObject",
"oss:DeleteObject"
],
"Resource": [
"acs:oss:*:1775305056529849:mybucket/file*"
],
"Effect": "Allow",
"Condition": {
"IpAddress": {
"acs:SourceIp": "192.168.0.1"
}
}
}
]
}
```

This is an authorization policy. You can use this policy to grant permissions for users through RAM or STS. The policy has a Statement (one policy can have multiple Statements). In the Statement, Action, Resource, Effect, and Condition are specified.

This policy authorizes your ‘mybucket’ and ‘mybucket/file*’ resources to corresponding users and supports GetBucketAcl, GetBucket, PutObject, GetObject, and DeleteObject actions. The Condition indicates 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 and Delimiter conditions apply during the GetBucket (ListObjects) action. For details about the two fields, see OSS API Documentation.

Configuration rules

Version

Policy version is defined. For configuration method in this document, it is set to “1”.

Statement

The Statement describes the authorization meaning. It can contain multiple meanings based on the business scenario. Each meaning includes a description of the Action, Effect, Resource, and Condition. The request system will check each statement for a match one by one. All successfully matched statements will be divided into Allow and Deny based on the difference of Effect settings, and Deny is given priority. If the matches are all Allow, the request passes authentication. If one of the matches is Deny or there are no matches, this request is denied to access.

Action

Actions fall into two categories: bucket-level actions and object-level actions. Bucket-level actions include oss:PutBucketAcl and oss:GetBucketLocation. The action objects are buckets and the action names correspond to the involved interfaces in a one-to-one manner. Object-level actions include oss:GetObject, oss:PutObject, oss:DeleteObject, oss:DeleteObject, and oss:AbortMultipartUpload. If you want to authorize actions for a type of object, you can select one or more of the above actions. In addition, all action names must be prefixed with “oss:”, as shown in the example above. Action is a list. There can be multiple Actions. The mapping between Actions and APIs is as follows:

Server-level

API	Action
GetService (ListBuckets)	oss:ListBuckets

Bucket-level

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
PutBucketReplication	oss:PutBucketReplication
GetBucketReplication	oss:GetBucketReplication
DeleteBucketReplication	oss:DeleteBucketReplication
GetBucketReplicationLocation	oss:GetBucketReplicationLocation
GetBucketReplicationProgress	oss:GetBucketReplicationProgress

Object level

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

Resource

Resource stands for a specific resource or resources on the OSS (the wildcard is supported). Resources are named in the format of "acs:oss:region:bucket_owner:bucket_name/object_name". For all bucket-level actions, the final part "/object_name" is not required. You can just render it as "acs:oss:region:bucket_owner:bucket_name". Resource is also a list and there can be multiple Resources. Here, the region field is currently not supported and set as "*".

Effect

Effect indicates the authorization result of the Statement. Two value options are available: Allow and Deny. When there are multiple Statement matches, the Deny is given higher priority.

For example, deny the deletion of a certain directory, but allow all operations for other files:

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": [
"oss:*"
],
"Resource": [
"acs:oss:*:*:bucketname"
]
},
{
"Effect": "Deny",
"Action": [
"oss:DeleteObject"
],
"Resource": [
"acs:oss:*:*:bucketname/index/*",
]
}
]
}
```

Condition

Condition indicates the conditions for the authorization policy. In the above example, you can set check conditions for acs:UserAgent and acs:SourceIp. The oss:Delimiter and oss:Prefix fields are used to restrict resources during the GetBucket action.

The OSS supports the following conditions:

Condition	Function	Valid value
acs:SourceIp	Specifying the IP address segment	Common IP address, wildcard (*) supported

acs:UserAgent	Specifying the http useragent header	String
acs:CurrentTime	Specifying valid access time	ISO8601 format
acs:SecureTransport	Whether HTTPS is used	“true” or “false”
oss:Prefix	Used as the prefix for ListObjects	Valid object name

Best practices

RAM and STS User Guide

Regions and endpoints

Regions and endpoints in a classic network

The Internet and intranet endpoints in each region for a classic network are as follows:

Region Name	Region Expression	Internet Endpoint	Internet endpoint supports HTTPS or not	Intranet Endpoint for ECS Access	Intranet endpoint supports HTTPS or not
China East 1 (Hangzhou)	oss-cn-hangzhou	oss-cn-hangzhou.aliyuncs.com	Yes	oss-cn-hangzhou-internal.aliyuncs.com	No
China East 2 (Shanghai)	oss-cn-shanghai	oss-cn-shanghai.aliyuncs.com	Yes	oss-cn-shanghai-internal.aliyuncs.com	No
China North 1 (Qingdao)	oss-cn-qingdao	oss-cn-qingdao.aliyuncs.com	Yes	oss-cn-qingdao-internal.aliyuncs.com	No
China North 2 (Beijing)	oss-cn-beijing	oss-cn-beijing.aliyuncs.com	Yes	oss-cn-beijing-internal.aliyuncs.com	No
China North 3 (zhangjiakou)	oss-cn-zhangjiakou	oss-cn-zhangjiakou.aliyuncs.com	Yes	oss-cn-zhangjiakou-internal.aliyuncs.com	No

China South 1 (Shenzhen)	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	Yes	oss-cn-shenzhen-internal.aliyuncs.com	No
Hong Kong	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	Yes	oss-cn-hongkong-internal.aliyuncs.com	No
US West 1 (Silicon Valley)	oss-us-west-1	oss-us-west-1.aliyuncs.com	Yes	oss-us-west-1-internal.aliyuncs.com	No
US East 1 (Virginia)	oss-us-east-1	oss-us-east-1.aliyuncs.com	Yes	oss-us-east-1-internal.aliyuncs.com	No
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	Yes	oss-ap-southeast-1-internal.aliyuncs.com	No
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	Yes	oss-ap-southeast-2-internal.aliyuncs.com	No
Asia Pacific NE 1 (Tokyo)	oss-ap-northeast-1	oss-ap-northeast-1.aliyuncs.com	Yes	oss-ap-northeast-1-internal.aliyuncs.com	No
EU Central 1 (Frankfurt)	oss-eu-central-1	oss-eu-central-1.aliyuncs.com	Yes	oss-eu-central-1-internal.aliyuncs.com	No
Middle East 1 (Dubai)	oss-me-east-1	oss-me-east-1.aliyuncs.com	Yes	oss-me-east-1-internal.aliyuncs.com	No

Note:

- Alibaba Cloud recommends that you use third-level domain names, that is, Bucket + Endpoint format, to share links or bind CNAME domain names. For example, the third-level domain name for the Shanghai bucket oss-sample would be oss-sample.oss-cn-shanghai.aliyuncs.com.

For new SDK versions (with the exception of C SDK), use http:// and https:// + Endpoint as the initialization parameters. Do not use a third-level domain name as the initialization parameter. For example, http://bucket.oss-cn-shanghai.aliyuncs.com cannot be used as an initialization parameter for a Shanghai endpoint. Note that currently the VPC endpoint does not support https.

Earlier SDK versions (for example, C, PHP, and Python SDKs) may directly use endpoints. For more details, refer to the documentation or code instructions for the SDK version you are using.

The original address oss.aliyuncs.com is directed to the Internet address of the Hangzhou node by default.

The original intranet address oss-internal.aliyuncs.com is directed to the intranet address of the Hangzhou node by default.

Regions and endpoints in a VPC network

To access OSS, ECS of a VPC network can only use the following endpoints:

Region Name	Region Expression	Endpoint of the VPC Network	Support HTTPS or not
China East 1 (Hangzhou)	oss-cn-hangzhou	vpc100-oss-cn-hangzhou.aliyuncs.com	No
China East 2 (Shanghai)	oss-cn-shanghai	vpc100-oss-cn-shanghai.aliyuncs.com	No
China North 1 (Qingdao)	oss-cn-qingdao	vpc100-oss-cn-qingdao.aliyuncs.com	No
China North 2 (Beijing)	oss-cn-beijing	vpc100-oss-cn-beijing.aliyuncs.com	No
China North 3 (zhangjiakou)	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	No
China South 1 (Shenzhen)	oss-cn-shenzhen	vpc100-oss-cn-shenzhen.aliyuncs.com	No
Hong Kong	oss-cn-hongkong	vpc100-oss-cn-hongkong.aliyuncs.com	No
US West 1 (Silicon Valley)	oss-us-west-1	vpc100-oss-us-west-1.aliyuncs.com	No

US East 1 (Virginia)	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	No
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	vpc100-oss-ap-southeast-1.aliyuncs.com	No
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	No
Asia Pacific NE 1 (Tokyo)	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	No
EU Central 1 (Frankfurt)	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	No
Middle East 1 (Dubai)	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	No

Access OSS

OSS-based app development

Development architecture

There are four components in typical OSS-based app development:

- OSS: Provides functions such as upload, download, and upload callback.
- Developer’ s mobile client (app or webpage application), called the client for short: Indirectly accesses the OSS though the service provided by the developer.
- Application server: The server that interacts with the client. This is also the server for the developer’ s service.
- Alibaba Cloud STS: Issues temporary credentials.

Service development process

Temporary credential upload authorization

1. The client sends a request to the application server asking to upload an object to OSS.
2. The application server must send a request to the STS server to obtain temporary credentials.

3. The application server replies to the client, returning the temporary credentials.
4. The client obtains authorization to upload to OSS (the STS AccessKey and token) and calls the mobile client SDK provided by OSS to upload data.
5. The client successfully uploads data to the OSS. If callback is not set, the process is complete. If the callback function is set, the OSS will call the relevant interface.

Here are several key points:

- The client does not have to request authorization from the application server for each upload. After the first authorization, the client will cache the temporary credentials returned by the STS until they expire.
- STS provides powerful access control functions that can restrict client access permission at the object level. This completely isolates the objects uploaded to the OSS by different clients, greatly enhancing the security of applications.

For more information, refer to Authorized Third-Party Uploads

Signed URL authorization for uploads and form uploads

1. The client sends a request to the application server asking to upload an object to OSS.
2. The application server replies to the client, returning credentials (signed URL or form).
3. The client obtains authorization to upload to OSS (the signed URL or form) and calls the mobile client SDK provided by OSS to upload data or directly uploads a form.
4. The client successfully uploads data to the OSS. If callback is not set, the process is complete. If the callback function is set, the OSS will call the relevant interface.

For more information, refer to Authorized Third-Party Uploads

Temporary credential download authorization

The process is similar to temporary credential upload authorization:

1. The client sends a request to the application server for downloading an object from OSS.
2. The application server must send a request to the STS server to obtain temporary credentials.
3. The application server replies to the client, returning the temporary credentials.
4. The client obtains authorization to download from OSS (the STS AccessKey and token) and calls the mobile client SDK provided by OSS to download data.
5. The client successfully downloads an object from OSS.

Here are several key points:

- Just as for uploads, the client will cache the temporary credentials to increase access speed.
- The STS likewise provides precise object download permission control, which, together with upload permission control, serves to completely isolate the OSS storage space of each mobile client.

Signed URL authorization for downloads

This is similar to signed URL authorization for uploads:

1. The client sends a request to the application server for downloading an object from OSS.
2. The application server replies to the client, returning the signed URL.
3. The client obtains authorization to download from OSS (the signed URL) and calls the mobile client SDK provided by OSS to download data.
4. The client successfully downloads an object from OSS.

Special note

The client cannot store the developer’ s AccessKey, but may only obtain a signed URL or the temporary credentials issued by the STS (the STS AccessKey and token) from the application server.

Reference for using the function

- SDK: Android SDK File Operations
- SDK: iOS SDK File Operations

Quick start

Quick start with the console

1. Log onto the OSS Console and activate OSS.
2. Create a bucket.
3. Upload and download files.

For details, refer to Get started with Alibaba Cloud OSS.

Quick introduction to OSS upload and download

Before getting started with SDKs, refer to the sections about the upload and download functions in the Developer Guide.

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

- OSS provides different file upload methods, such as using a single PUT request to complete a Simple Upload, using webpage forms for direct uploads, called Form Upload, and uploading large files with Multipart Upload. For video monitoring and other applications,

OSS also provides Append Object.
- Likewise, OSS provides multiple download methods: Simple Download and, for larger files, Resumable Download.

Quick start with SDKs

1. After activating OSS, retrieve the AccessKeyId and AccessKeySecret from the console.
2. Download the SDKs for various programming languages.
3. Based on the descriptions in the SDK documentation, perform uploads, downloads, and other operations.

For details, see the OSS SDK Reference.

Bucket management

Create a bucket

Users can create a bucket in an existing region. Note that the following conditions apply to bucket creation:

- Each user can create up to 30 buckets.
- The name of each bucket must be globally unique; otherwise, the bucket cannot be created.
- Each bucket name must comply with the naming rules.
- Once a bucket is created, its name and region cannot be modified.

OSS provides an Access Control List (ACL) for permission control. You can configure an ACL when creating a bucket, and modify the ACL after creating the bucket. If no ACL is configured, the default value is Private. For more details, refer to Set Bucket ACLs.

Reference

- Console: Create a bucket
- SDK: Java SDK-Create a bucket-Bucket
- API: Put Bucket

Set bucket read and write permissions (ACL)

Only a bucket owner can, when creating a bucket, set bucket read and write permissions using Access Control List (ACL). The owner can also modify the ACL for that bucket according to service 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 files in the bucket. The fees incurred by these operations will be borne by the owner of the bucket. Use this permission with caution.
public-read	Only the owner of the bucket can perform write operations on the files in the bucket, while others (including anonymous users) can perform read operations on the files.
private	Only authorized users are allowed to read, write, and delete files in the bucket. Others cannot access the files in the bucket without authorization.

For details, refer to Access Permissions.

Reference

Setting ACLs for a bucket

- Console: Access Permissions Configuration
- SDK：Java SDK-**Set Bucket ACL** in Bucket
- API: Put BucketACL

Obtaining ACLs for a bucket

- Console: After logging in to the console, users can view the ACL in the bucket attributes.
- SDK：Java SDK-**Obtain Bucket ACL** in Bucket
- API: Get BucketACL

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, users can directly view a list of all created buckets.
- API: GetService
- SDK：Java SDK-List buckets in Bucket

Additional links

- Create a bucket

Obtain bucket region information

The region represents the physical location of the data center. The returned Location field indicates the region where the 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, refer to Regions and endpoints.

Reference

- Console: After logging in to the console, users can directly view bucket attributes on the console.
- API: Get Bucket Location
- SDK: Java SDK-**Obtain the Bucket Address** in Bucket

Delete a bucket

You can delete buckets you have created. However, buckets must first be emptied of files and file fragments before deletion can occur. To delete all files in a bucket, it is recommended that you use Lifecycle Management.

Reference

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

Upload files

Simple upload

Simple upload is when a user uploads a single object by using the Put Object method described in the OSS API. This is applicable to any scenario where a single HTTP request interaction completes an upload, for example, upload of a small file.

Set object Meta when uploading files

A simple upload can carry object metadata that describes the object, for example, Content-Type and other standard HTTP headers, as well as user-defined information. For more details, refer to Object Meta.

Upload restrictions

- The maximum object size permitted for each simple upload is 5 GB.
- Naming restrictions:
    - The name must be UTF-8 encoded.
    - The length must be between 1-1,023 bytes
    - The name cannot start with a backslash “/ “ or forward slash “\” .

Upload large files

Only objects up to 5 GB can be uploaded at any one time using the simple upload process. For objects larger than 5 GB, refer to Multipart Upload.

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 details, refer to Access Control.

In addition to bucket-level and object-level access permissions, OSS also provides account-level authorization to authorize third-party uploads. For details, see Authorized Third-party Upload for Upload Security.

Post-upload operations

- To initiate a callback request to a specified application server in order to perform subsequent operations, use Upload Callback.
- To process uploaded images, use Image processing.

Reference

- API: PutObject
- SDK: Java SDK-**PutObject** in Object
- Console: Uploading Files

Best practices

- RAM and STS User Guide
- Web Client Direct Data Transfer and Upload Callback

Additional links

- Upload callback
- Introduction to mobile development upload scenarios
- Download uploaded files
- Cloud processing for uploaded Images
- Access control for upload security
- Authorized third-party upload for upload security
- Copy, delete, and manage uploaded files

Form upload

Form upload is when an object is uploaded using the Post Object request in the OSS API. Form upload bypasses objects being forwarded to the server, and instead uploads objects directly from the client to OSS, reducing bottlenecks at the server end due to object resizing.

Applicable scenarios

The following example scenarios demonstrate the application of the form upload process in a career search website environment:

Without using form upload

1. A website user uploads their object (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.

Using form upload

1. A website user uploads their resume.
2. The website server responds to the upload page.
3. The resume is uploaded to OSS.

Upload restrictions

- The maximum object size permitted for each form upload is 5 GB.
- Naming restrictions:
  - The name must be UTF-8 encoded.
  - The length must be between 1-1,023 bytes
  - The name cannot start with a backslash “/ “ or forward slash “\” .

Upload security and authorization

To prevent unauthorized third parties from uploading objects to the developer’ s bucket, OSS provides bucket- and object-level access permission control. For details, refer to Access Control.

In addition to bucket-level and object-level access permissions, OSS also provides account-level authorization to authorize third-party uploads. For details, refer to Post Object.

Basic process

1. Construct a post policy. For more details, refer to Post Policy.
   In this policy example:
   - The expiration time for site user uploads is 2115-01-27T10:56:19Z (in actual usage this setting is determined based on site requirements).
   - The maximum upload file size is 104,857,600 bytes.
   - Python code is used.
   - 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 encoding.
3. Use the OSS AccessKeySecret to sign the base64 encoded policy.
4. Construct an HTML page for uploads.

   Open the HTML page and select the file to upload. The following is an example output:

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

```
import base64
import hmac
from optparse import OptionParser

def convert_base64(input):
return base64.b64encode(input)

def get_sign_policy(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 encoding
base64policy = convert_base64(policy)
print("base64_encode_policy: %s" % base64policy)

#3 Use the OSS AccessKeySecret to sign the base64 encoded policy
signature = get_sign_policy(access_key_secret, base64policy)

#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="OSSAccessKeyId" 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_action_redirect" value="http://oss.aliyun.com">
<input type="text" name="success_action_status" value="201">
<input name="file" type="file" id="file">
<input name="submit" value="Upload" type="submit">
</form>
</body>
</html>
''' % (bucket, endpoint, access_key_id, base64policy, signature)
f = open(out, "wb")
f.write(form)
f.close()
print("form is saved into %s" % out)

if __name__ == '__main__':
parser = OptionParser()
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 segment as post_object.py and use Python to run it.

```
 Usage:
python post_object.py --bucket=Your bucket --endpoint=The bucket's OSS domain name --id=Your AccessKeyId --key=Your AccessKeySecret --out=Output file name

Example:
python post_object.py --bucket=oss-sample --endpoint=oss-cn-hangzhou.aliyuncs.com --id=tphpxp --key=ZQNJzf4QJRkrH4 --out=post.html
```

Note:

- "success_action_redirect" value="http://oss.aliyun.com" indicates the page to navigate to after a successful upload. This can be replaced as needed.
- "success_action_status" value="201" indicates that Status Code 201 is returned after a successful upload. This can be replaced according to site requirements.
- If the generated HTML file is post.html, open post.html and select the file to upload. In this example, the client navigates to the OSS homepage after a successful upload.

Reference

- API: PostObject

Best practices

- Web client direct data transfer
- Cross-origin resource sharing (CORS)

Reference links

- Upload Callback
- Introduction to Mobile Development Upload Scenarios
- Download uploaded files
- Cloud processing for uploaded images
- Access control for upload security
- Authorized third-party upload for upload security
- Copy, delete, and manage uploaded files

Multipart upload

Multipart upload allows objects larger than 5 GB to be split into multiple data blocks (or parts in OSS) with each data block then uploaded separately. When all data block uploads are complete, an OSS API is called to combine the parts into the original object.

Applicable scenarios

Multipart upload is recommended for the following scenarios:

Poor network connectivity
If at any point one data block upload fails, a user can re-upload only the failed data block without requiring all uploads involved to restart.

Resume upload required
An upload in progress can be paused and resumed at any time.

Accelerate an upload
If the file to be uploaded to OSS is very large, multiple parts can be uploaded in parallel to accelerate the upload.

Stream an upload
Objects of unknown sizes can be uploaded at any time.

The process is as follows:

1. The object is split into data blocks of equal sizes. Split the file to be uploaded according to a specified part size.
2. The multipart upload task is initialized, as indicated by InitiateMultipartUpload.
3. The data blocks are uploaded either in sequence or parallel, as indicated by UploadPart.
4. All data blocks are successfully uploaded and combined into the original object, as indicated by CompleteMultipartUpload.

Note:

- Each data block (except the last block) cannot be smaller than 100 KB; otherwise, the call to the CompleteMultipartUpload will fail.
- After splitting the file into parts, the parts are ordered by the partNumbers specified during the upload. The upload speed does not correlate to the number of data blocks uploaded (either in sequence or in parallel), as both the user’ s network conditions and

the device load must be considered.
- By default, when the upload is complete, but CompleteMultipartUpload has not been called, the parts will not be automatically recycled. Therefore, to terminate the upload and delete the data-occupied storage space, call AbortMultipartUpload. To automatically recycle uploaded parts, refer to Lifecycle Management.

Concepts

Multipart upload is recommended for scenarios such as mobile device data transfers, large object uploads, and video streaming, as the lifecycle of uploaded data blocks and objects is permanent.

If the system crashes during a multipart upload, you can resume the upload by using the ListMultipartUploads and ListParts APIs to retrieve all multipart upload tasks for an object and list the completed uploads in each task. This allows uploads to be resumed from the last uploaded part. The same concept apply to pausing and resuming uploads.

Restrictions

- Size limit: the object size is determined by data block size. The function supports a maximum of 10,000 parts, with a minimum data block size of 100 KB (with an exception to the final data block, which may be smaller) and a maximum data block size of 5 GB.
- Naming restrictions:
    - The name must be UTF-8 encoded.
    - The length must be between 1-1,023 bytes
    - The name cannot start with a backslash “/ “ or forward slash “\” .

Upload security and authorization

To prevent unauthorized third parties from uploading objects to a developer’ s bucket, OSS provides bucket- and object-level access permission control. For details, refer to Access Control.

In addition to bucket and object-level access permissions, OSS also provides account-level authorization to authorize third-party uploads. For details, refer to Authorized Third-party Upload for Upload Security.

Post-upload operations

- To initiate a callback request to a specified application server in order to perform subsequent operations, use Upload Callback.
- To process uploaded images, use Cloud Processing for Uploaded Images.

Reference

- APIs: MultipartUpload, InitiateMultipartUpload, UploadPart, UploadPartCopy, CompleteMultipartUpload, AbortMultipartUpload, ListMultipartUploads, ListParts
- SDK: Java SDK-Multipart upload in MultipartUpload

Best practices

- RAM and STS User Guide
- Web Client Direct Data Transfer

Additional links

- Upload callback
- Introduction to mobile development upload scenarios
- Download uploaded files
- Cloud processing for uploaded images
- Access control for upload security
- Authorized third-party upload for upload security
- Copy, delete, and manage uploaded files

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 will complicate the management of most objects. If the object parts are large, the data will suffer a substantial delay.

To simply 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 very simple. When video data are produced, they can be immediately added to the same object through the append object method. The client simply needs to regularly retrieve the object length and compare it with the previous value. If new readable data are 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
    - It uses UTF-8 encoding.
    - The length must be 1-1,023 bytes.
    - It cannot start with “/ “ or “\” .
- 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.

Upload security and authorization

To prevent unauthorized third parties from uploading objects to the developer’ s bucket, OSS provides bucket- and object-level access permission control. For details, refer to Access Control.
In addition to bucket-level and object-level access permissions, OSS also provides account-level authorization to authorize third-party uploads. For details, refer to Authorized Third-party Upload for Upload Security.

Post-upload Operations

To process uploaded images, users can use Cloud Processing for Uploaded Images.
For audio/video file format conversion, users can use Media Transcoding.

Reference for using the function

- API: Append Object
- SDK: Java SDK-Append Object Example

NOTE: Append object method does not support upload callback.

Best practices

- RAM and STS User Guide

Reference links

- Downloading Uploaded Files
- Cloud Processing for Uploaded Images
- Access Control for Upload Security
- Authorized Third-party Upload for Upload Security

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, there are two methods in which 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 ensure security. For details, refer to 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, refer to STS Java SDK.

Best practices

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

Additional links

- Upload callback
- Introduction to mobile development upload scenarios
- Download uploaded files
- Cloud processing for uploaded images
- Access control for upload security
- Form upload

Upload callback

When an upload is complete, OSS can perform a callback to the application server. To perform callback, users simply need to attach the relevant Callback parameter to the request sent to OSS. APIs that currently support callback include PutObject, PostObject, and CompleteMultipartUpload.

> **Note:** Currently, upload callback is supported only for Mainland China data centers, and only simple uploads (PutObject), form uploads (PostObject), and multipart uploads completion (CompleteMultipartUpload) support upload callback.

Applicable scenarios

A typical upload callback scenario is when authorized third-party users upload files to the OSS, the clients specify the servers for callback. Then after the upload is complete, the OSS automatically initiates a callback request to the application server over HTTP. This notifies the application server

that the upload is complete, so it can perform operations such as database modification. Upon receiving 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 parameters that provide certain information. The parameters are divided into two types:

- System-defined parameters
  This parameter specifies simple information such as bucket name and object name.
- User-defined parameters
  This parameter specifies information based on the application logic determined when sending a request, including callback to the OSS, and carries details such as the user ID of the request initiator. For information on user-defined parameters, refer to Callback.

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

Reference

- API: Callback

Best practices

Direct upload to OSS from web

Set up data callback for mobile apps

Reference links

- Permission management for a mobile app
- Introduction to mobile development upload scenarios
- Download uploaded files
- Cloud processing for uploaded images
- Access control for upload security
- Authorized third-party upload for upload security
- Copy, delete, and manage uploaded files

Download files

Simple download

A simple download occurs when a user downloads an uploaded file (object). The object download is accomplished through an HTTP GET request.

When a user accesses a certain object, there are two possibilities:

- 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 the rules of generating object URLs, refer to Accessing OSS.

For the access to an object by a user-defined domain name, refer to Accessing OSS with User-defined Domain Names.

For details about object and bucket access permission control, refer to Access Control.

To authorize a third-party user to download an object from a private bucket, refer to Authorized Third-party Download.

To use multipart download, refer to Multipart Download.

Reference

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

Best practices

- RAM and STS User Guide

Additional links

- File Upload Methods
- Upload Callback
- Mobile Client Development and Download Scenario Introduction

- Cloud Processing for Uploaded Images
- Secure Download Access Control
- Authorized Third-party Download for Download Security
- Copying, Deleting, and Managing Uploaded Files

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 will resume 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: Get Object

Additional links

- File Upload Methods
- Upload Callback
- Mobile Client Development and Download Scenario Introduction
- Cloud Processing for Uploaded Images
- Secure Download Access Control
- Authorized Third-party Download for Download Security
- Copying, Deleting, and Managing Uploaded Files

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>?OSSAccessKeyId=<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 desired URL expiration time.
- Signature, which is the developer’ s signature string. For details, refer to API Documentation - signature section.

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 will contain 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, refer to STS Java SDK.

Implementation method

1. Third-party users send a request to the application server to obtain an AccessKeyID,

AccessKeySecret, and STS Token issued by STS.
2. 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
- Console: Get object URL

Best practices

- RAM and STS User Guide

Additional links

- File Upload Methods
- Upload Callback
- Mobile Client Development and Download Scenario Introduction
- Cloud Processing for Uploaded Images
- Secure Download Access Control
- Authorized Third-party Download for Download Security
- Copying, Deleting, and Managing Uploaded Files

File management

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 will be 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: Put Object
- SDK: Set Object HTTP Headers and User-define Metadata in the Java SDK documentation

You can set object Meta when using multipart uploads.

Reference:

- API: InitiateMultipartUpload
- SDK: Java SDK-Initializing 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 for using the function:

- API: Copying Objects
- SDK: Java SDK-Using CopyObjectRequest to Copy Objects

Retrieve object Meta

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

Reference for using the function:

- API: Head Object
- SDK: Java SDK-Only Retrieve Object Metadata

View the object list

You 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.
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 will still contain 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 will display any object ending with “/“ as a folder.

Users can use a combination of Delimiters and Prefixes to simulate folder functions 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 the returned values will enumerate the files in the folder and the subfolders (directories) will be returned in the CommonPrefixes section. Recursive

files and folders in subfolders will not be 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 directories and files
Based on the API request conventions, you must set the Prefix to "", and the Delimiter to "/":
The returned results are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
<Name>oss-sample</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>File D</Key>
<LastModified>2015-11-06T10:07:11.000Z</LastModified>
<ETag>"8110930DA5E04B1ED5D84D6CC4DC9080"</ETag>
<Type>Normal</Type>
<Size>3340</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<CommonPrefixes>
<Prefix>Directory A/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory B/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

We can see that:
Contents returns the first-level file: "File D".
CommonPrefixes returns the first-level directories: "Directory A/" and "Directory B/", but the files in these directories are not shown.

2. List second-level directories and files under Directory A
Based on the API request conventions, 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"?>
<ListBucketResult>
```

```
<Name>oss-sample</Name>
<Prefix>Directory A/</Prefix>
<Marker> </Marker>
<MaxKeys>1000</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>false</IsTruncated>
<Contents>
<Key>Directory A/File C</Key>
<LastModified>2015-11-06T09:36:00.000Z</LastModified>
<ETag>"B026324C6904B2A9CB4B88D6D61C81D1"</ETag>
<Type>Normal</Type>
<Size>2</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<Contents>
<Key>Directory A/File D</Key>
<LastModified>2015-11-06T09:36:00.000Z</LastModified>
<ETag>"B026324C6904B2A9CB4B88D6D61C81D1"</ETag>
<Type>Normal</Type>
<Size>2</Size>
<StorageClass>Standard</StorageClass>
<Owner>
<ID>oss</ID>
<DisplayName>oss</DisplayName>
</Owner>
</Contents>
<CommonPrefixes>
<Prefix>Directory A/Directory B/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory A/Directory C/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
<Prefix>Directory A/Directory D/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

We can see that:
Contents returns the second-level files: "Directory A/File C" and "Directory A/File D".
CommonPrefixes returns the second-level directories: "Directory A/Directory B/", "Directory A/Directory C/", and "Directory A/Directory D/". The file names under these directories are not shown.

Reference

- API: Get Bucket
- SDK：Java SDK-Listing Files in a Bucket

Copy an object

In certain situations, you simply need 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, it is recommended 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 will only update the Object Meta. For more details about Object Meta, refer to Object Meta.

Before copying an object, note the following precautions:

- You must have permissions to operate the source object. Otherwise the operation will fail.
- 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: Copy Object
- SDK: Java SDK-Object

Copy large objects

You need to take different steps to copy a large object. 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

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, it is recommend that Lifecycle Management is implemented. Once the rules are specified, OSS will use 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

Manage object lifecycle

The lifecycle of a bucket can be configured to define various rules for the bucket’ s objects. Currently, you can use rules to delete matching objects. Each rule is composed of the following:

- Object name prefix
  This rule will only apply to objects with a matching prefix.
- Operation
  The operation you want to perform on the matching objects.
- Date or number of days
  The operation is executed on objects on the specified date, or across a specified number of days, after the object’ s last modification time.

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

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

- If the prefix of a rule is logs/, the rule applies to the first three objects that are prefixed with logs/.
- If the prefix of a rule is doc/readme.txt, the rule only applies to the object 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.

When an object matches an overdue rule, the OSS will include the x-oss-expiration header in the response to the Get Object or Head Object requests. The header contains two key-value pairs: expiry-date indicates the expiration date of the object; rule-id indicates the matched rule ID.

Example

You can set the lifecycle configurations of a bucket through the open interface of the OSS. Lifecycle configurations are given in XML format. Below is a specific example.

```
<LifecycleConfiguration>
<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>
<CreatedBeforeDate>2014-12-31T00:00:00.000Z</CreatedBeforeDate>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

In the above example, all elements are described as follows:

- **ID**: a unique identifier of each rule.
- **Status**: Enabled or Disabled. OSS only supports the Enabled rules.
- **Prefix**: the prefix.
- **Expiration**: the operation expiration date. The sub-elements **CreatedBeforeDate** and **Days** specify the absolute and relative expiry time, respectively.
    - **CreatedBeforeDate** indicates that files with a last modification time before 2014-12-31T00:00:00.000Z will be deleted. Objects modified after this time will not be deleted.
    - **Days** indicates that files that were last modified more than 10 days ago will be deleted.

In the first rule, the OSS will delete objects that are prefixed with logs/ and were last updated 10 days ago. The second rule indicates that objects prefixed with doc/ that were last modified before December 31, 2014 will be deleted, but the rule will not take effect because it is in disabled status.

Detailed analysis

- The naming rules of the prefix are the same as those of the object.
- When the prefix is empty, the rule applies to all objects in the bucket.
- Each prefix of a rule must be unique. For example, if a bucket has two rules whose prefixes are logs/ and logs/program, OSS will return an error.
- If a rule is set to delete objects on a specific date, the date must be midnight UTC and comply with the ISO8601 format, for example, 2014-01-01T00:00:00.000Z. In this example, OSS will delete matched objects after midnight on January 1, 2014.
- If, in a rule to delete objects, the number of days is specified, OSS will sum up the last update time (Last-Modified) as well as 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 midnight on April 16, 2014.
- OSS deletes the objects matched with 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 multiple times, the last update time will then be 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.

Reference

- API: Put Bucket Lifecycle

Cross-region replication

Bucket Cross-Region Replication automatically and asynchronously copies objects in buckets across different OSS data centers. It also synchronizes changes to objects in the source bucket and copies them across to the target bucket, enabling support for cross-region disaster recovery of buckets.

Application scenarios

Scenarios in which enabling Bucket Cross-Region Replication is recommended are as follows:

Compliance requirements
Through cross-region synchronization, data can be copied between OSS data centers at different locations to satisfy compliance requirements such as specified distances between objects.

Latency
In order to minimize object access latency between objects at different geographical points, a copy of the desired object can be maintained at an OSS data center that is physically closer to users.

Data backup and disaster recovery
In the event of natural disasters, backup data can be saved to another OSS data center to maintain data security and availability.

Atypical operations
If you have computing clusters in different OSS data centers, and wish to use them to analyze the same group of objects, you can maintain copies of the desired objects copies at the different data centers.

Supported synchronization functions

Currently, cross-region synchronization supports buckets with different names. For two buckets in different regions, a user can sync the data in the source bucket to the target bucket using the following features:

Real-time data synchronization
Monitor data addition, deletion, and modification in real time and sync changes to the target region bucket. For files of 2M or larger, synchronization may take several minutes. This ensures the ultimate consistency of the data on both sides.

Historical data migration
Synchronize historical data in the source bucket, creating two identical data copies.

Real-time synchronization progress retrieval
Display the latest synchronization time node for real-time data synchronization. For historical data migration, this feature shows the percentage of data migrated.

Easy configuration
Use the management interfaces on the OSS Console to easily monitor and perform actions based on site requirements.

Restrictions

Currently, cross-region synchronization is available in Mainland China. Support for other

regions will be announced when available.

The two buckets to be used for data synchronization must belong to different regions. Data synchronization cannot be performed between buckets in the same region.

Users can simultaneously operate on two buckets in the synchronization status. However, objects copied from the source bucket may overwrite any objects of the same name in the target bucket. Use this function with caution.

Because Bucket Replication uses an asynchronous copying method, it may take time for data to be fully copied to the target bucket. Prepare accordingly to ensure the process can be completed properly.

Cross-region synchronization applies only when the two buckets to be synced do not enable data to be synced to or from a third bucket. For example, if synchronization is activated from Bucket A to Bucket B, the user cannot activate synchronization from Bucket A to Bucket C before deleting the synchronization configuration between Bucket A and Bucket B.

Reference

- Console: Cross-Region Replication

Manage back-to-source settings

Back-to-source settings allow for multiple back-to-source reading methods to be applied, meeting hot data migration and specific request redirection requirements.

This setting enables the URL of each OSS Get request to be matched, which will then specify a back-to-source 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

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.

The process is as follows:

1. A client requests data of an object.
2. OSS determines the object does not exist, and forwards the request to the source URL.
3. The source URL returns the object through the OSS, which goes to the client. OSS simultaneously writes the object data in order to process future requests.

Example scenario

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 ossimport2 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 will be 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 will be scanned, and all non-migrated data will be 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.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.com/dir1/, the process is the same as the preceding example, but the OSS back-to-source URL will be http://www.example.com/dir1/object.jpg although the object written to the OSS will remain 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 will not be sent to the origin site.

If the origin site returns data in chunks, the OSS will likewise return data to the user in chunks.

The OSS will return and save the following header information from the origin site:

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

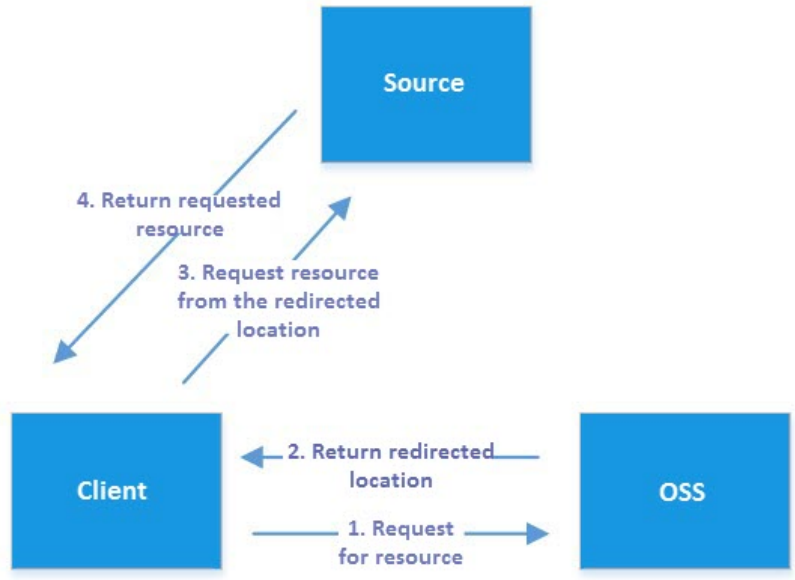
An x-oss-tag response header will be added to mirroring write-back files, with the value “MIRROR” + space + urldecode(back-to-source URL). In the example given above, this would be x-oss-tag:MIRROR http%3a%2f%2fwww.example-domain.com%2fdir1%2fimage%2fexampleobject.jpg. After the file is written back to the OSS, so long as it is not overwritten again, this header will be 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 will 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 will be the HTTP status 404, which will be 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 will return 424 to the user, the error code for ‘MirrorFailed’.

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:

1. A client requests data of an object.
2. OSS determines the object does not exist, and returns a redirect location source to the client.
3. The client sends the request direct to the redirection location provided by OSS.
4. The source returns the object directly to the client.

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 will then return 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 ensures that OSS errors are undetected by a user.

Reference

- Console: Back-To-Source Rule Management

Security management

Set access logging

The OSS provides automatic saving of server access logs. A bucket owner can log on to OSS Console to enable the server access logging feature for all the owner’ s buckets. When access logging is activated for a bucket (Source Bucket), the OSS will generate an object containing all access request logs of that bucket (by hour) and write the object into the user-designated bucket (Target Bucket) according to fixed naming rules.

Object naming rules for access logging

```
<TargetPrefix><SourceBucket>-YYYY-mm-DD-HH-MM-SS-UniqueString
```

In the naming rules, the TargetPrefix is specified by the user; YYYY, mm, DD, HH, MM and SS give the year, month, day, hour, minutes and seconds of the creation time in Arabic numerals (note the digits); and UniqueString is the string generated by the OSS system. An example for the name of an object actually used to store OSS access logs is given below:

```
MyLog-oss-example-2012-09-10-04-00-00-0000
```

In the above example, “MyLog- “ is the Object prefix specified by the user; “oss-example” is the name of the origin bucket; “2012-09-10-04-00-00” is the Object creation time (Beijing time); and “0000” is the string generated by the OSS system.

Log file format

(Separated by spaces from left to right):

Name	Example	Description

Remote IP	119.140.142.11	IP address from which the request is initiated (the proxy or user firewall may block this field)
Time	[02/May/2012:00:00:04 +0800]	Time when the OSS receives the request
Request-URI	“GET /aliyun-logo.png HTTP/1.1”	User-Requested URI (including query-string)
HTTP Status	200	HTTP status code returned by the OSS
SentBytes	5576	Traffic that the user downloads from the OSS
RequestTime (ms)	71	Time spent in completing this request (in ms)
Referer	http://www.aliyun.com/product/oss	Requested TTP Referer
User-Agent	curl/7.15.5	HTTP User-Agent header
HostName	oss-example.oss-cn-hangzhou.aliyuncs.com	Domain name for access request
Request ID	505B01695037C2AF032593A4	UUID used to uniquely identify this request
LoggingFlag	true	Whether the access logging function is enabled
Requester AliCloud ID	1657136103983691	AliCloud ID of the requester, “-” for anonymous access
Operation	GetObject	Request type
Bucket	oss-example	Name of the bucket requested for access
Key	/aliyun-logo.png	User-Requested Key
ObjectSize	5576	Object size
Server Cost Time (ms)	17	Time taken by the OSS server to process this request (in ms)
Error Code	NoSuchBucket	Error code returned by the OSS
Request Length	302	Length of user request (byte)
UserID	1657136103983691	ID of the bucket owner
Delta DataSize	280	Bucket size variation, “-” for no change
Sync Request	-	Whether this is a back-to-source request from CND, “-” for no

Detail analysis

- The source bucket and target bucket must belong to the same user.
- TargetPrefix indicates the name prefix of the object used for storing access logs. The field can be left blank.
- The source bucket and target bucket can be the same or different buckets. You can save logs from multiple source buckets to the same target bucket (in this case, it is recommended that you assign different values to TargetPrefix).
- The OSS generates a bucket access log file every hour. However, all requests in the hour may not be recorded in the log file, but may be recorded in the previous or next log file.
- In the naming rules for log files generated by the OSS, “UniqueString” is just a UUID that the OSS generates for an object to uniquely identify the file.
- Each time the OSS generates a bucket access log file, this is considered a PUT operation and the occupied space is recorded, but the generated traffic is not recorded. After log files are generated, you can operate these log files as common objects.
- The OSS ignores all query-string parameters prefixed by “x-” but such query-string parameters are recorded in access logs. If you want to mark a special request from massive access logs, you can add a query-string parameter prefixed by “x-” to the URL. For example:http://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.pnghttp://oss-example.oss-cn-hangzhou.aliyuncs.com/aliyun-logo.png?x-user=admin
  When the OSS processes the above two requests, the results are the same. However, you can search access logs with “x-user=admin” to quickly locate the marked request.
- You may see “-” in any field of OSS logs. It indicates that data is unknown or the field is invalid for the current request.
- Certain fields will be added to the end of OSS log files in the future based on the requirements. It is recommended that developers take compatibility issues into consideration when developing log processing tools.

Reference for using the function

- Console: Server Access Logging

Anti-leech settings

The 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. Users can log in to OSS Console or use APIs to configure a referer white list 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 white list to http://www.aliyun.com. Then, only requests with a referer of http://www.aliyun.com can access the objects in the bucket.

Detail analysis

- Anti-leech verification will be 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 comma “,” .
- The referer field supports the wildcard “* “ and “?” 。
- Users can set whether to allow access requests with empty referer fields.
- When the white list is empty, the system will not check if the referer field is null (otherwise, all requests will be rejected).
- When the white list is not empty and the rules do not allow null referer fields, only requests with referers in the white list will be allowed. Other requests (including null referer requests) will be rejected.
- If the white list is not empty and the rules allow empty referer fields, requests with empty referer and with the referers in the white list will be allowed. Other requests will be rejected.
- The three bucket permissions (private, public-read, and public-read-write) will all 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 will be 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 for using the function

- API: Put Bucket Referer
- Console: Set Anti-leech

Cross-origin resource sharing

Cross-origin access, or the cross-origin of JavaScript, is a browser restriction set for the sake of security, namely, the same-origin policy. When Website A tries to use the JavaScript code in its webpage to access Website B, the attempt will be rejected by the browser because A and B are two

websites of different origins.

Cross-origin access needs arise frequently in actual usage, such as when OSS is used at the back end 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 via www.a.com.If cross-origin access is set, users could upload their data directly to OSS instead of relaying it via 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 details about the specific CORS rules, refer to W3C CORS Norms. In simple terms, CORS indicates the origin of where the request is originated by using a header containing the origin of the HTTP request. As in the previous example, the origin header contains www.a.com.After receiving the request, the server will judge based on certain rules whether the request should be accepted or not. If yes, the server will attach the Access-Control-Allow-Origin header in the response. The header contains www.a.com, indicating that cross-origin access is allowed. In case that the server accepts all the cross-origin requests, just set the Access-Control-Allow-Origin header to *. The browser will determine whether the cross-origin request is successful or not based on whether the corresponding header has been returned or not. In case that no corresponding header is attached, the browser will block the request. In case that the request is not a simple one, the browser will firstly send an OPTIONS request to obtain the CORS configuration of the server. In case that the server does not support the following operations, the browser will also block the following requests.

OSS provides the configuration of the CORS rule, accepting or rejecting corresponding cross-origin requests as needed. 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, i.e. the OSS CORS rule is only used to determine whether to attach the relevant CORS headers. Whether the request should be blocked should 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 in the first place, the server will return to the user the resource with the Access-Control-Allow-Origin header “www.a.com” . When www.b.com initiates its request, the browser will return its previous cached request to the user. As the header content does not match the CORS request, the

subsequent request fails.

Reference for using the function

- API: Cross-origin Resource Sharing
- SDK: Java SDK-Cross-origin Resource Sharing
- Console: Cross-origin Resource Sharing

Server-side encryption

OSS supports server-side encryption of data uploaded by users: When a user uploads data, the OSS encrypts the received user data and permanently stores the encrypted data. When a user downloads data, the OSS automatically decrypts the encrypted 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 side. In other words, there is major big difference between downloading an object encrypted on the server side and downloading a common object, because the OSS manages the entire codec process for users.

Currently, the OSS’ s server-side encryption is an attribute of objects. When creating an object, a user only needs to add the HTTP Header “x-oss-server-side-encryption” to the Put Object request and specify its value as “AES256”. Then, the object can be encrypted on the server side before it is stored. Currently, server-side encryption is supported by the following operations:

- Put Object
- Copy Object
- Initiate Multipart Upload

Detail analysis

- Except the Put Object, Copy Object, and Initiate Multipart Upload requests, if any other request received by the OSS contains the ‘x-oss-server-side-encryption’ header, the OSS will directly return HTTP Status Code 400, with the error code in the message body being InvalidArgument.
- Currently, the OSS only supports the AES256 encryption algorithm. If the user specifies another value for the ‘x-oss-server-side-encryption’ header, the OSS will directly return HTTP Status Code 400, with the error code in the message body being ‘InvalidEncryptionAlgorithmError’.
- For objects stored after server-side encryption, the OSS returns the x-oss-server-side-encryption header in the API requests below, with its value being the entropy encryption algorithm:
    - Put Object
    - Copy Object

- Initiate Multipart Upload
- Upload Part
- Complete Multipart Upload
- Get Object
- Head Object

Reference for using the function

- API: Append Object
- API: Put Object
- API: Copy Object
- API: Post Object

Static website hosting

On 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/
```

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 mandatory.

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 will return 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 APS.NET content.

For access to a bucket-based static website through a user-defined domain name, refer to Bind custom domain names.

OSS restricts access by bucket domain names, which means user files cannot be directly viewed in a browser. Users are recommended to use CNAMEs.

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 static website hosting mode is set for a bucket, and the user accesses the root domain name of a static website or a nonexistent object, the OSS will return a specified object to the user and bills the return traffic and requests to the bucket owner.

Reference

- API: Put Bucket Website
- Console: Static Website Hosting

Monitoring service

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 details, refer to 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 Record**.

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, refer to Alarm service overview.
- For instructions about how to use the OSS alarm service, refer to Alarm service user guide.
- For details about OSS metric indicators, refer to 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 in order to read the data storage of Cloud Monitor. For details, refer to Metric data access through the OpenAPI.

The console displays metric data up until the past seven days. To view historical metric data earlier than seven days, use the Cloud Monitor SDK. For details, refer to Metric data access through the OpenAPI.

Metric data access through the OpenAPI

The OpenAPI of Cloud Monitor allows you to access OSS metric data. For usage information, refer to:

- Cloud Monitor OpenAPI User Manual
- 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, as well as 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 will be applied to the new bucket.

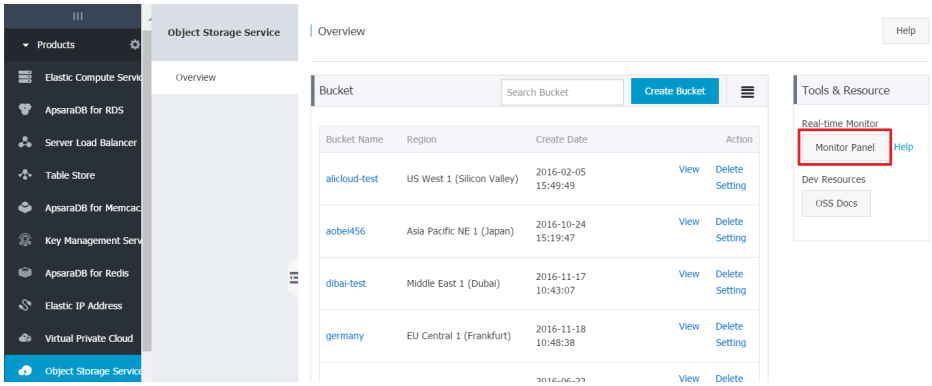
Monitoring service user guide

Cloud Monitor Console

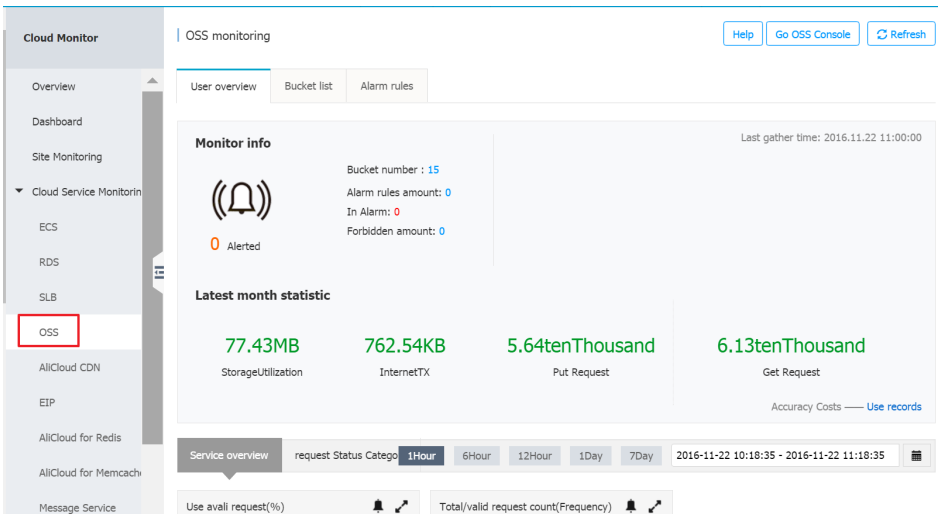
OSS monitoring entry

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 your Alibaba Cloud account, and then on the homepage navigate to **Object Storage Service** in the left-side navigation bar. Open the OSS overview page, and then click **Monitor Panel** on the right side, as shown below:



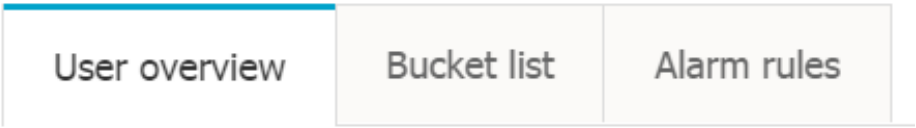
Log on to your Alibaba Cloud account, and then on the homepage navigate to Cloud Monitor in the left-side navigation bar. Open the Cloud Monitor overview page, and then click **Cloud Service Monitoring** > **OSS** as shown below:



OSS monitoring page

The OSS monitoring page consists of the following three tabs:

- User overview
- 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 OSS Console** to log on to the OSS console.

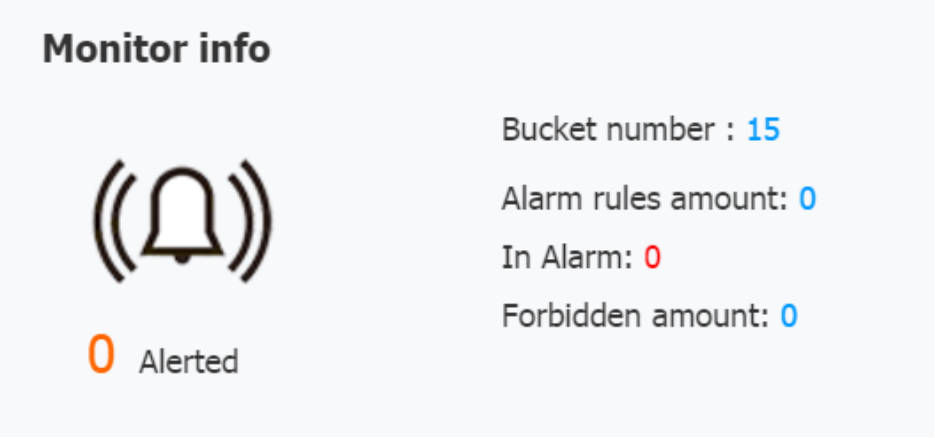
User overview

The User overview page displays the following information:

- User monitoring information
- Latest month statistics
- User-level metric indicators

User monitoring information

This module shows the total number of your buckets and related alarm rules.



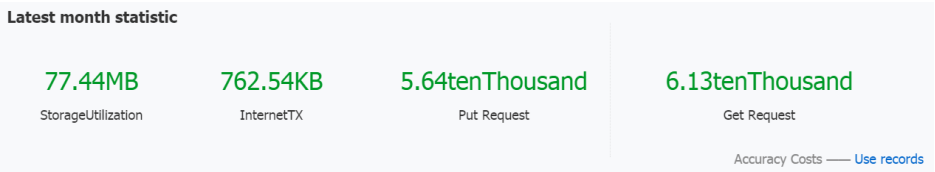
The parameters are as follows:

- 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

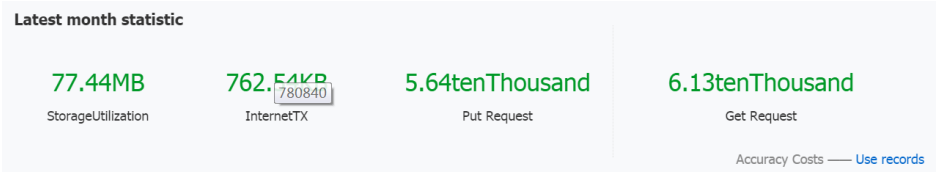
Latest month statistics

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 utilization
- Internet traffic
- Put request
- Get request

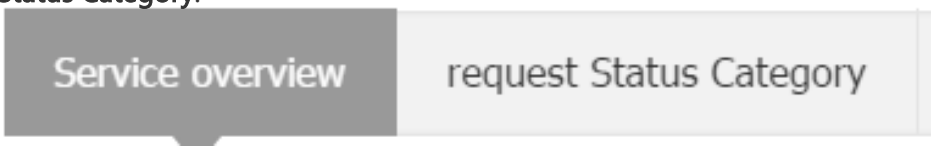


The unit of each value is automatically adjusted by the order of magnitude. The exact value is displayed when you hover the cursor over the selected value.



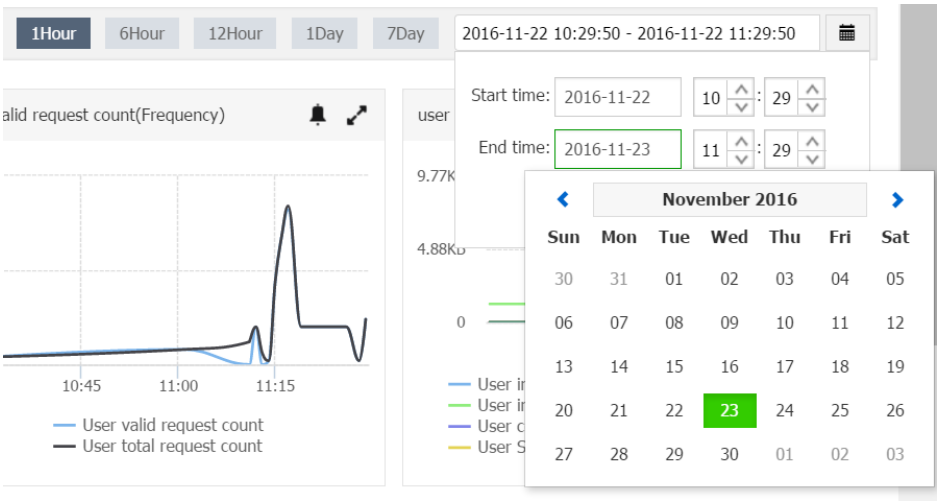
User-level metric indicators

This module displays user-level metric charts and tables and consists of **Service overview** and **request Status Category**:



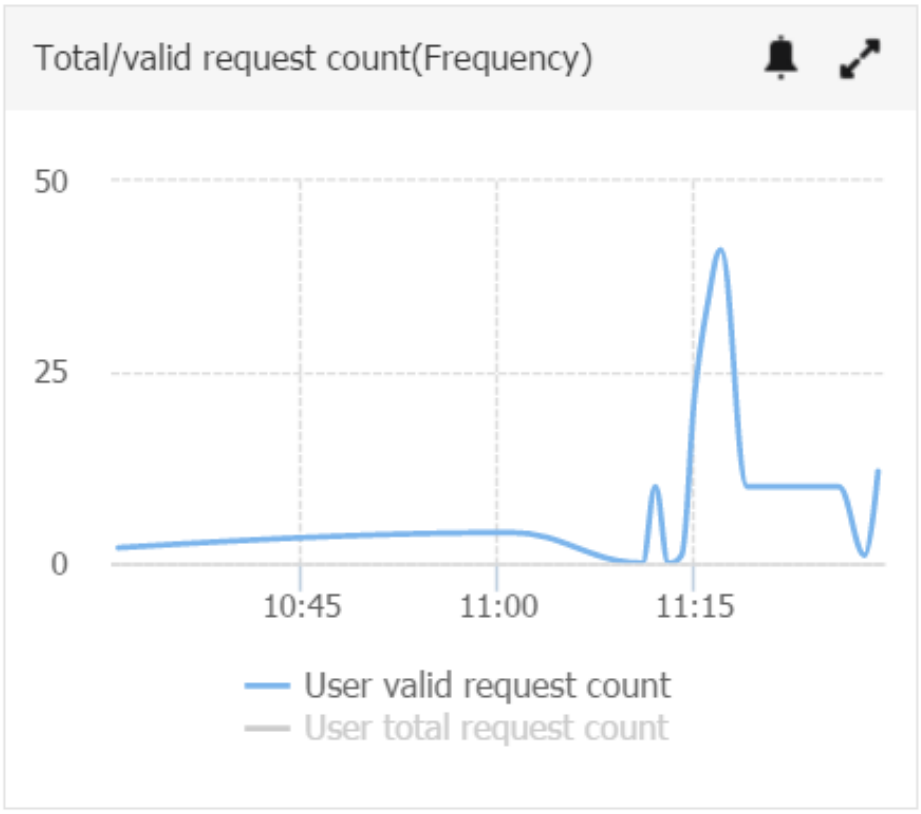
You can select a pre-determined time range, or define a time range in the custom time boxes, to display the corresponding metric chart or table.

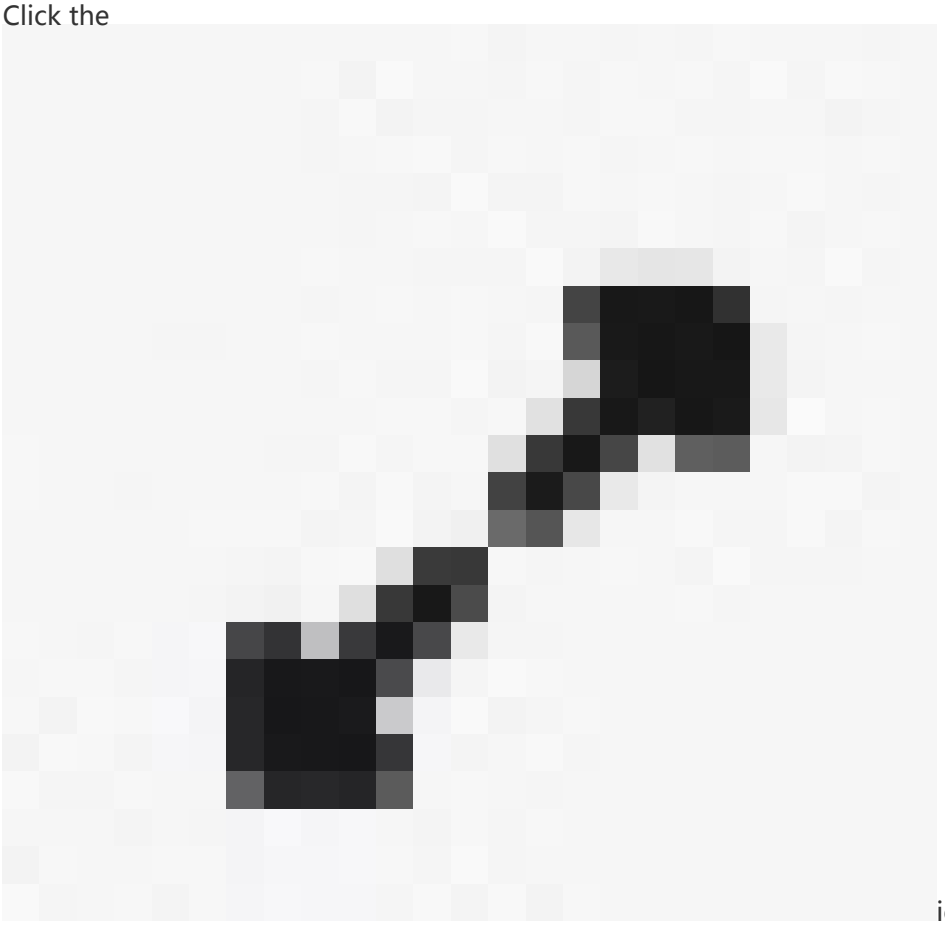
- The following time range options are available: 1 hour, 6 hours, 12 hours, 1 day, and 7 days. The default option is 1 hour.
- The custom time boxes allow the start time and the end time to be defined at the minute-level.



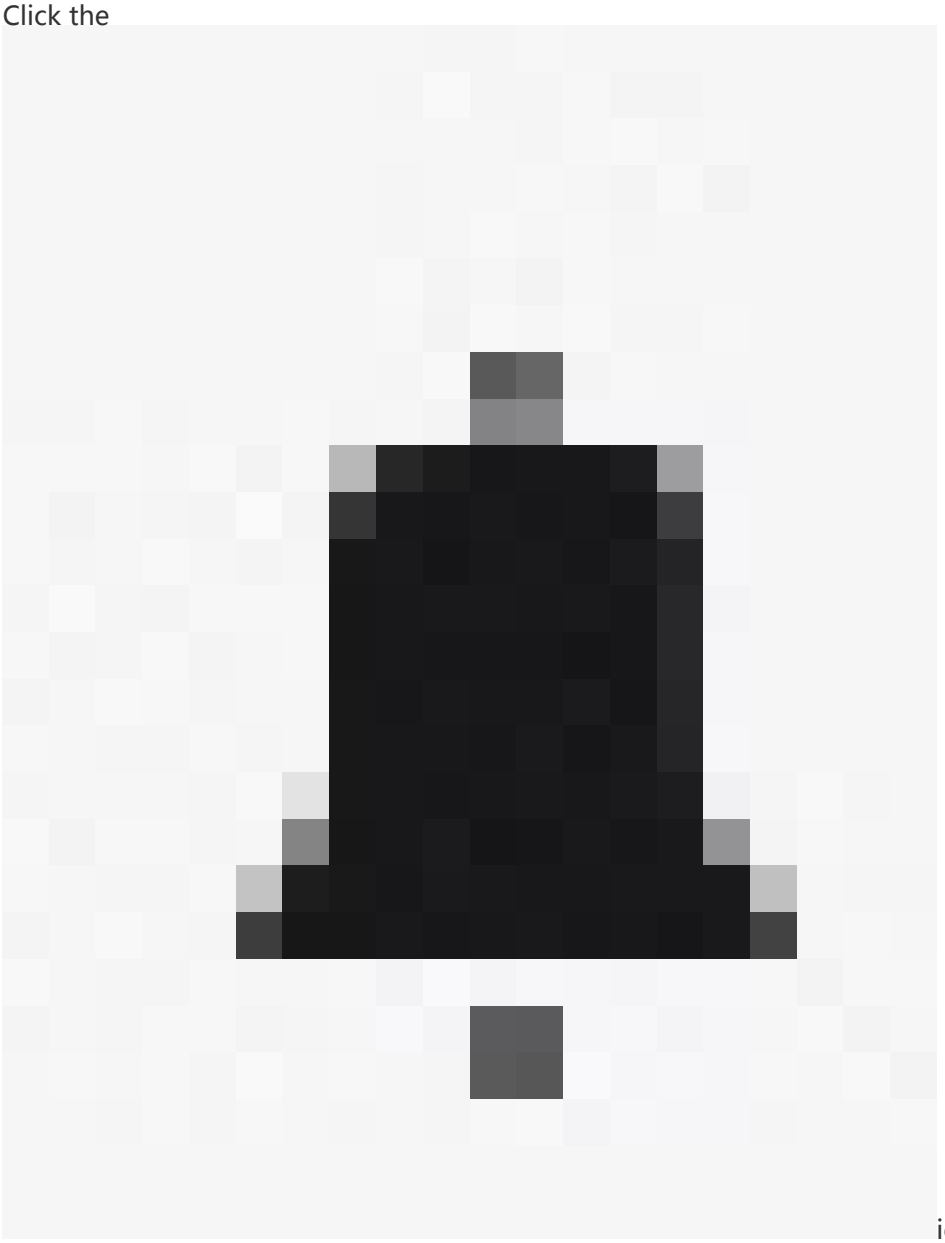
Metric charts/tables support the following display modes:

- Legend hiding: You can click a legend to hide the corresponding indicator curve, as shown in the figure below:



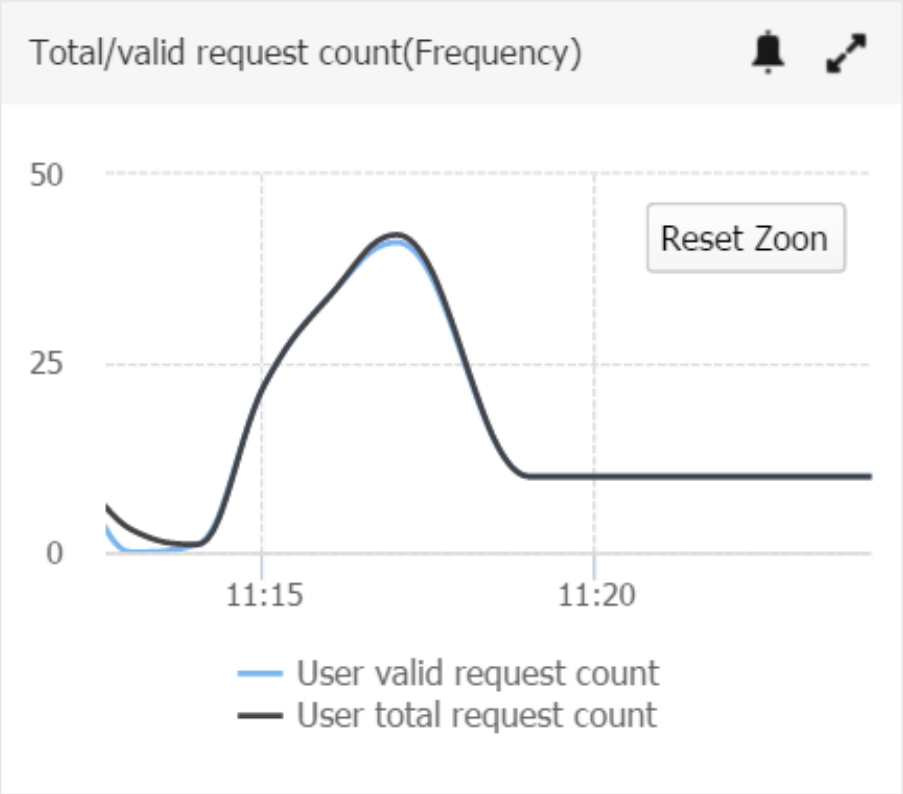
- Click the  icon in the upper-

right corner of a metric chart to zoom in on the chart. NOTE: 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 details, refer to the Alarm Service User Guide.

  Note: You cannot set alarm rules for tables and metering reference indicators.

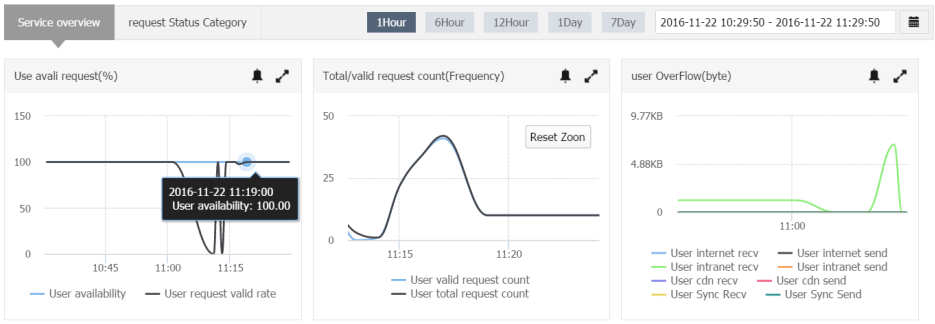
- Place the cursor inside the curve area of a chart, and press and hold the left button on the mouse while dragging the mouse to extend the time range. Click **Reset Zoom** to restore the original time range.



Service overview

The **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.



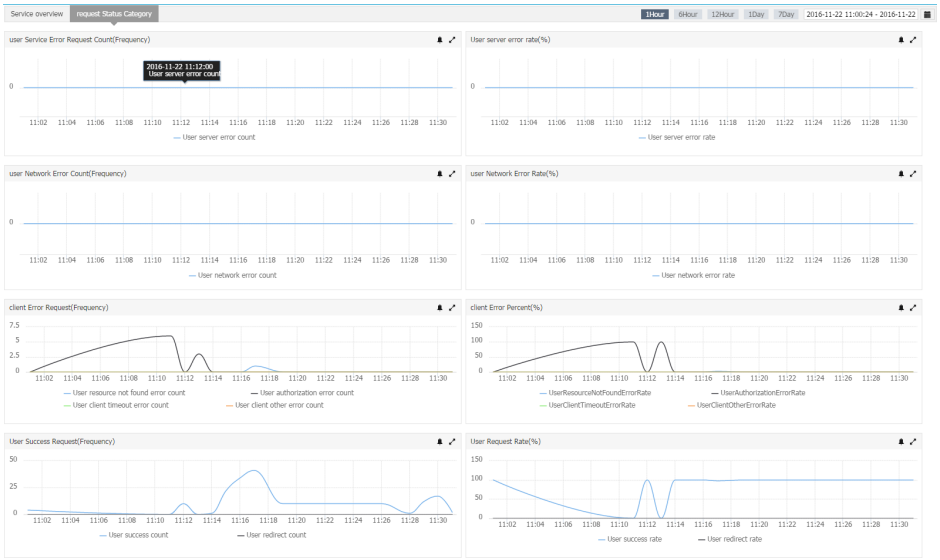
User level request

Metric	Sum value	Percent
User authorization error count	9yunjiankong.metric.unitName.frequency	5.73%
User resource not found error count	1yunjiankong.metric.unitName.frequency	0.64%
User success count	147yunjiankong.metric.unitName.frequency	93.63%
Sum	157yunjiankong.metric.unitName.frequency	100%

Request status category

The **Request status category** page displays the metric data of request state distribution through the following metric charts:

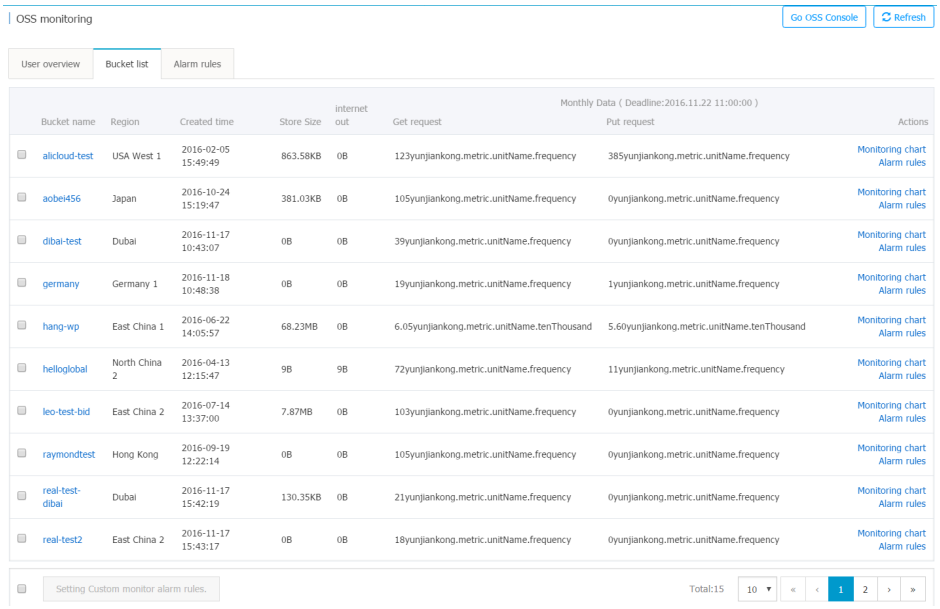
- User service error request count
- User server error rate
- User network error count
- User network error Rate
- Client error request, which includes four metric indicators: number of error requests indicating resource not found, number of authorization error requests, number of client-site timeout error requests, and number of other client-site error requests
- Client error percent, which includes four metric indicators: percentage of error requests indicating resource not found, percentage of authorization error requests, percentage of client-site timeout error requests, and percentage of other client-site error requests
- User success request, which includes two metric indicators: number of successful requests and number of redirect requests
- User request rate, 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.



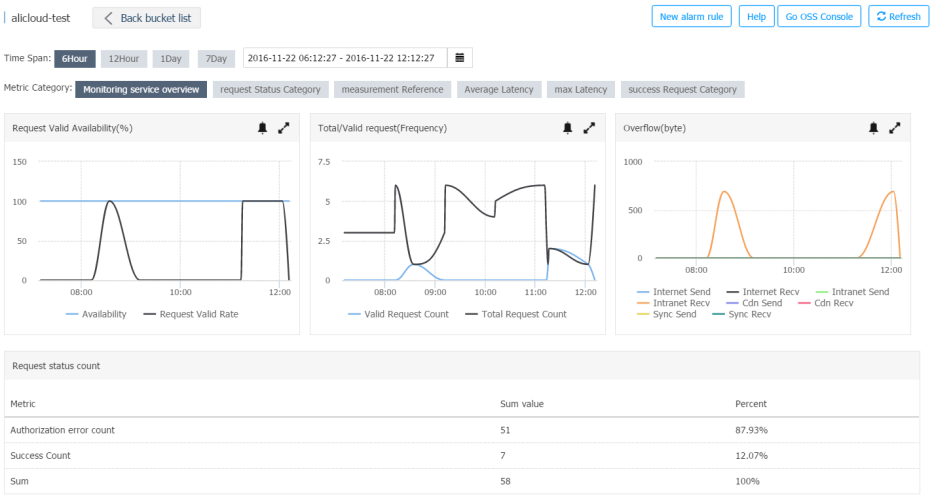
Display parameters are as follows:

- 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 desired bucket, or go to the Alarm rules tab to display all

alarm rules of the bucket.
- Enter the desired 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 desired bucket names and click Setting custom monitor alarm rules to batch set alarm rules. For details, refer to the Alarm Service User Guide.

Bucket-level monitoring view

Click **Monitoring chart** next to the desired 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 category
- Measurement reference
- Average latency
- Maximum latency
- Success request category

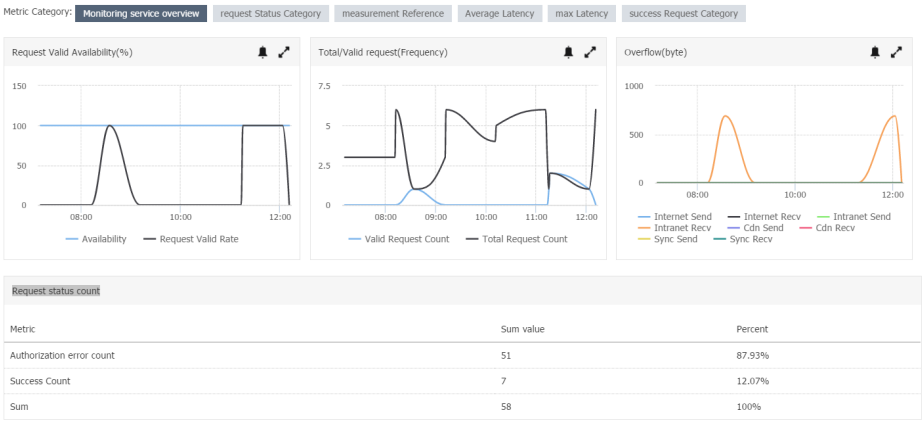
Except measurement reference, other indicators are displayed with an aggregation granularity of 60s. The default time range for bucket-level metric charts is of the previous six hours, whereas that for user-level metric charts is of the previous hour. Click **Back to bucket list** in the upper-left corner 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

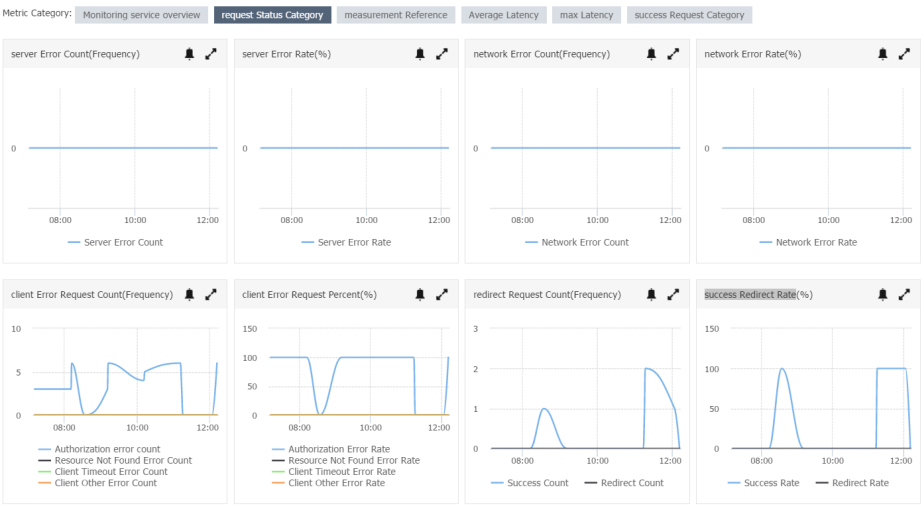
number of valid requests
- Overflow, which includes eight metric indicators: Internet outbound traffic, Internet inbound traffic, Intranet outbound traffic, Intranet inbound traffic, CDN outbound traffic, CDN inbound traffic, outbound traffic of cross-region replication, and inbound traffic of cross-region 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 category

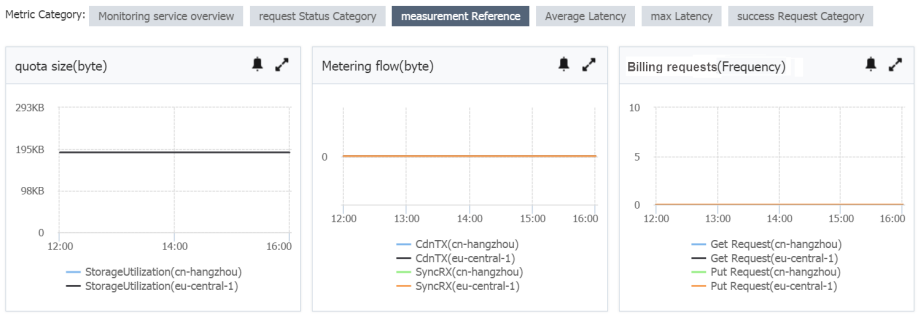
This indicator group is similar to the request state details at the user level, but the former displays metric data at the bucket level. The main metric charts include:

- Server error count
- Server error rate
- Network error count
- Network error 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 timeout 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 timeout 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 figure below:



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.

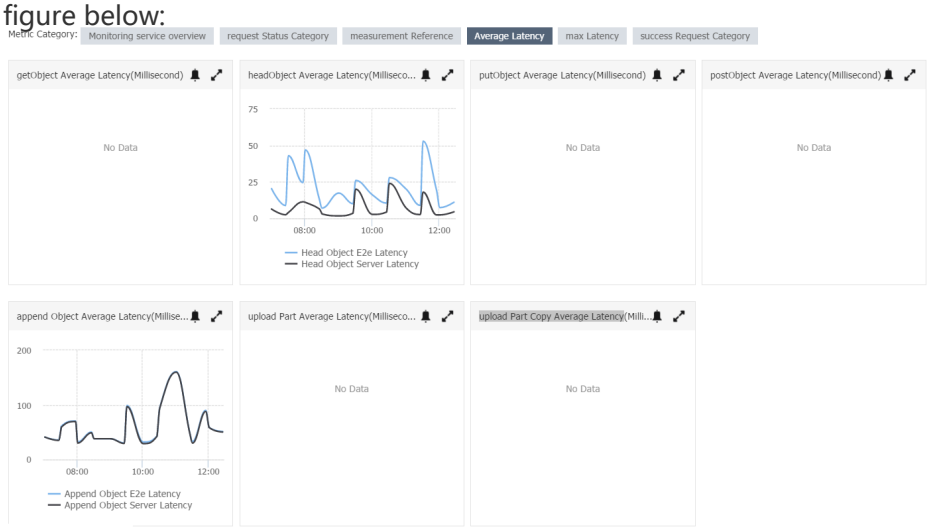
Average latency

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:

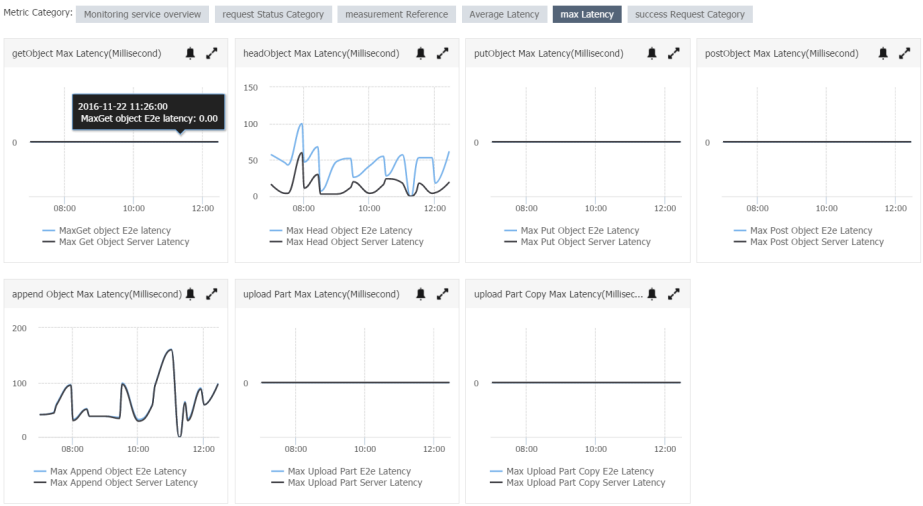


Maximum latency

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 figure below:

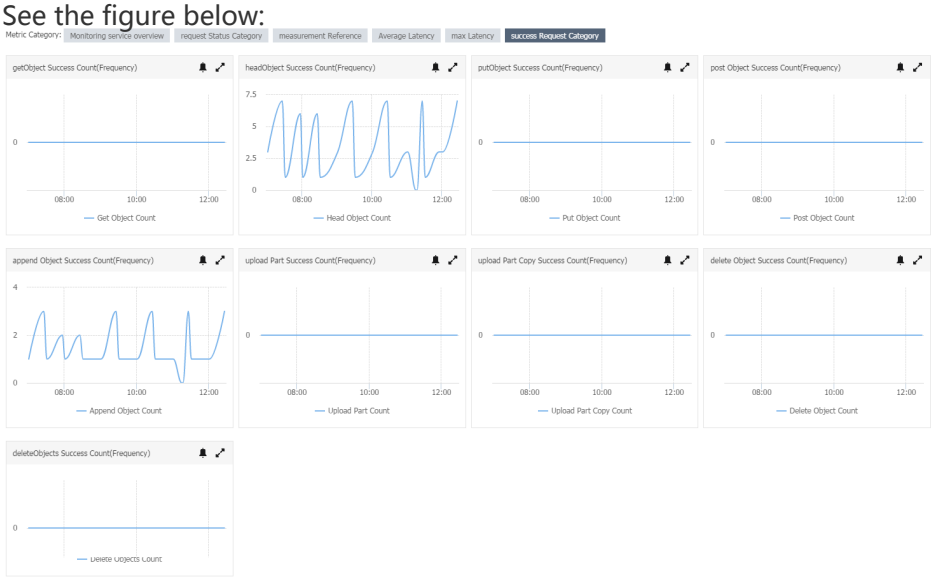


Success request category

This indicator group contains the successful request count indicators of API monitoring. The metric charts include:

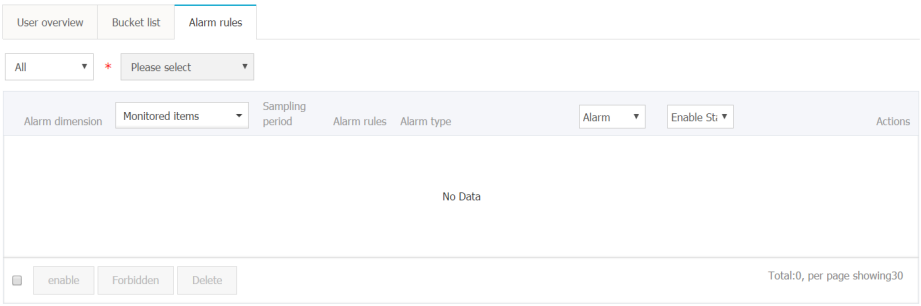
- getObject Success Count
- headObject Success Count
- putObject Success Count
- post Object Success Count
- append Object Success Count
- upload Part Success Count
- upload Part Copy Success Count
- delete Object Success Count
- deleteObjects Success Count

See the figure below:



Alarm rules

The “Alarm rules” tab page allows you to view and manage all your alarm rules, as shown in the figure below:



For the description and usage of the “Alarm Rules” tab page, refer to the Alarm Service User Guide.

Additional links

For more details regarding the important points and user guide of the monitoring service, refer to the related chapter in Monitoring, diagnosis, and troubleshooting.

Alarm service user guide

Prerequisites

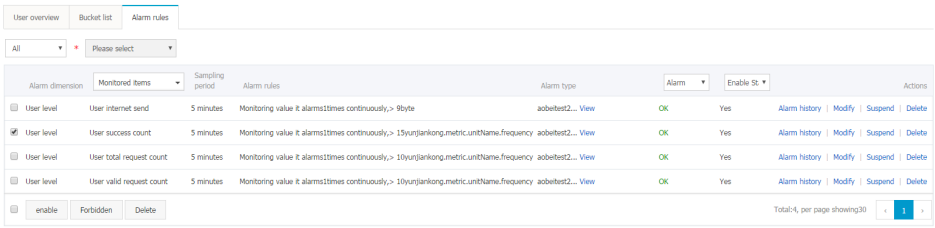
In order to help familiarize yourself with the basic concepts and configurations of alarm contacts and alarm contact groups, it is recommended that the following documents are read prior to this user guide:

- 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. There are two alarm dimensions: user-level and bucket-level.

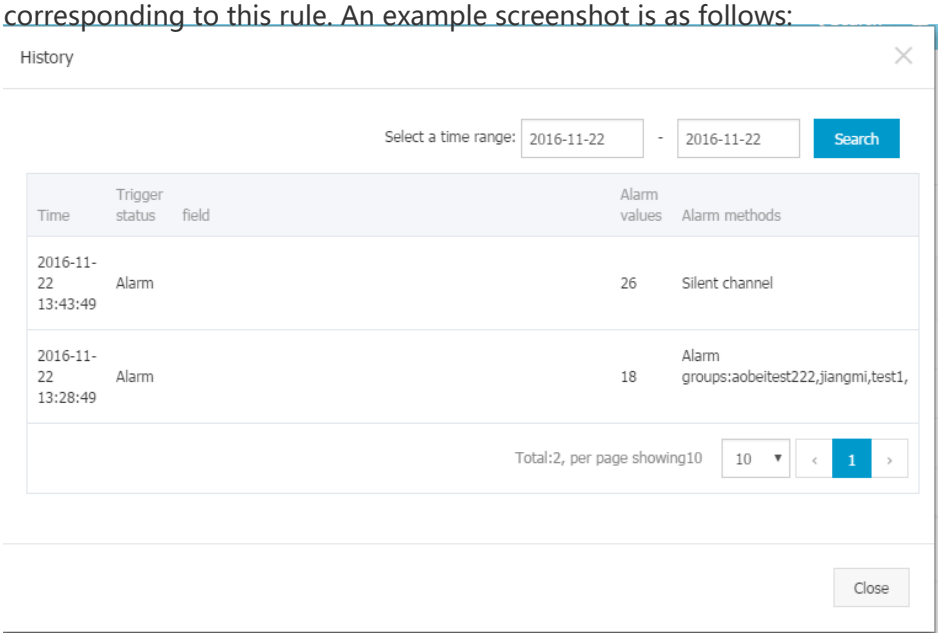
Alarm rule page

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:



The parameters are as follows:

- Click **Modify** next to the desired alarm rule to modify it.
- Click **Delete** next to the desired 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 desired alarm rule to deactivate it. Once the alarm rule is suspended, you will 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 desired alarm rule to activate it. The rule will 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 desired alarm rule to view information on past alarms corresponding to this rule. An example screenshot is as follows:



Alarm history concepts

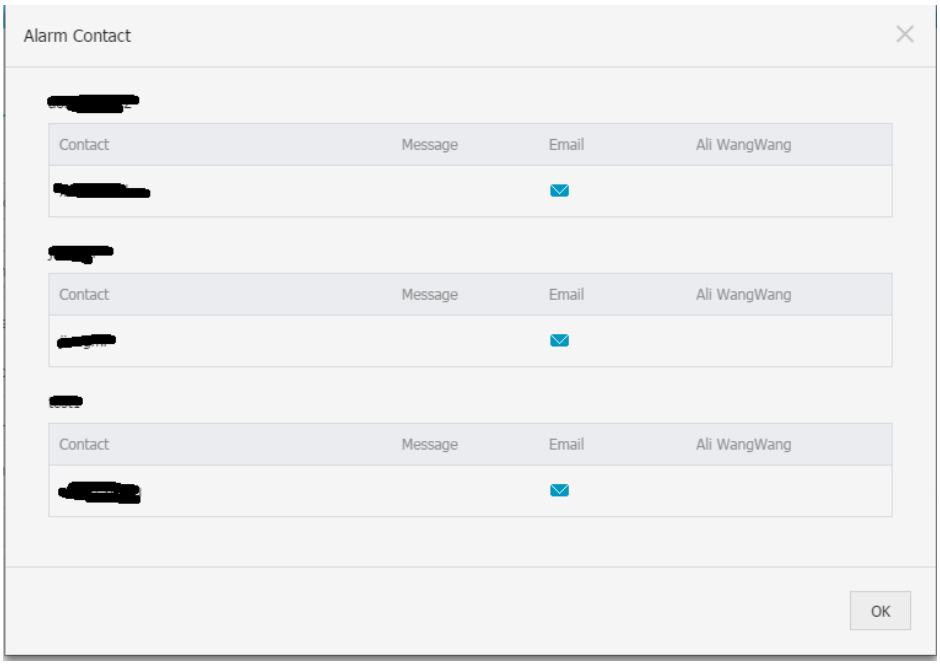
Alarm history refers to past status changes of a selected alarm rule. Operations such as switching from normal status to alarm status, or switching from alarm status to normal status, are considered status changes. Additionally, a status change called **channel silence** is also available.

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 will be 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 will be 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 desired 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 will be displayed.

- Bucket drop-down box: If you select **Bucket Level** in the alarm dimension drop-down box, this box will list 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.

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 desired bucket in order to see alarm rules for that bucket. You can then filter returned information using selections in the drop-down box such as **All**, **Metric Item**, **Alarm status**, and **Activation status**.

View alarm rules for a specific bucket

If you need 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.

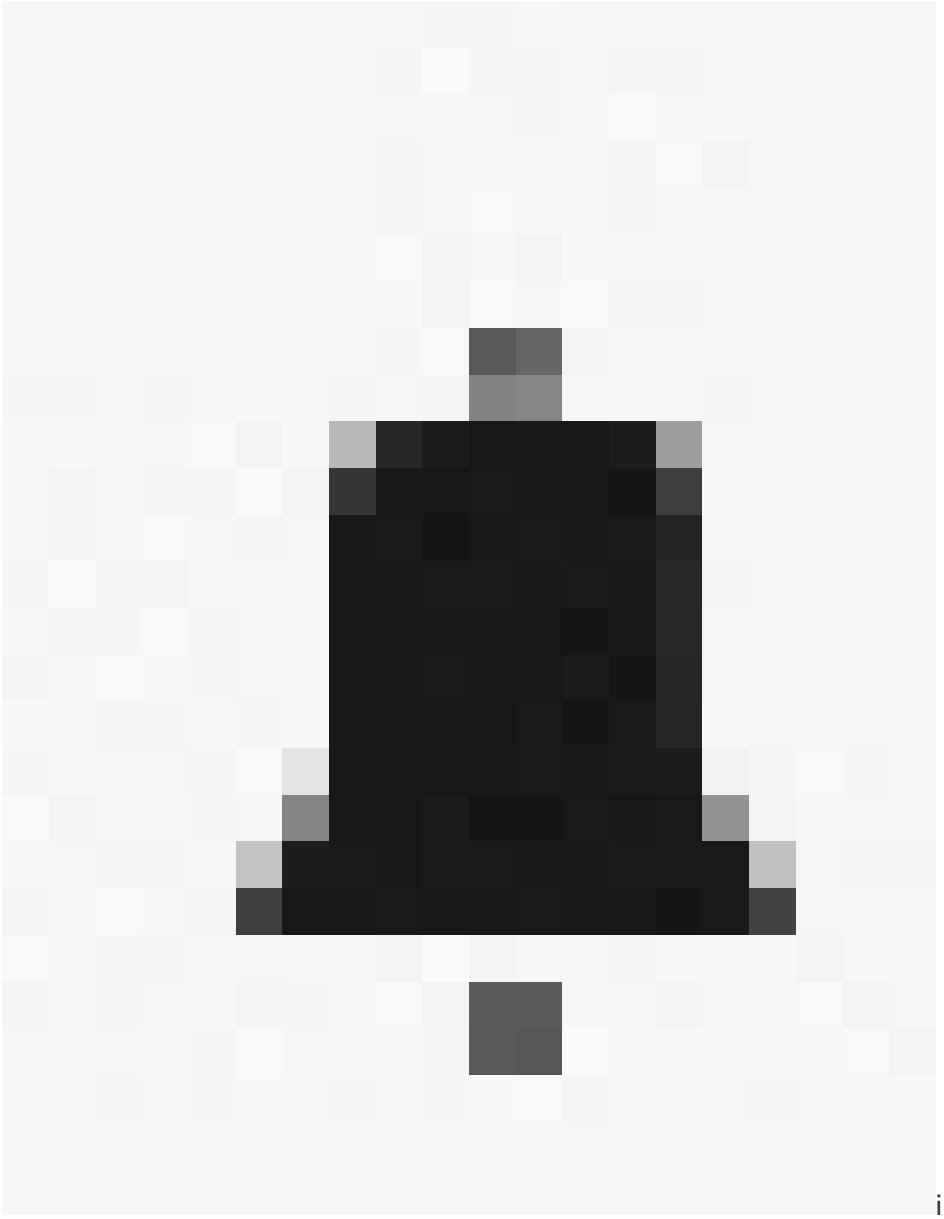
View alarm rules in a certain activation status

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

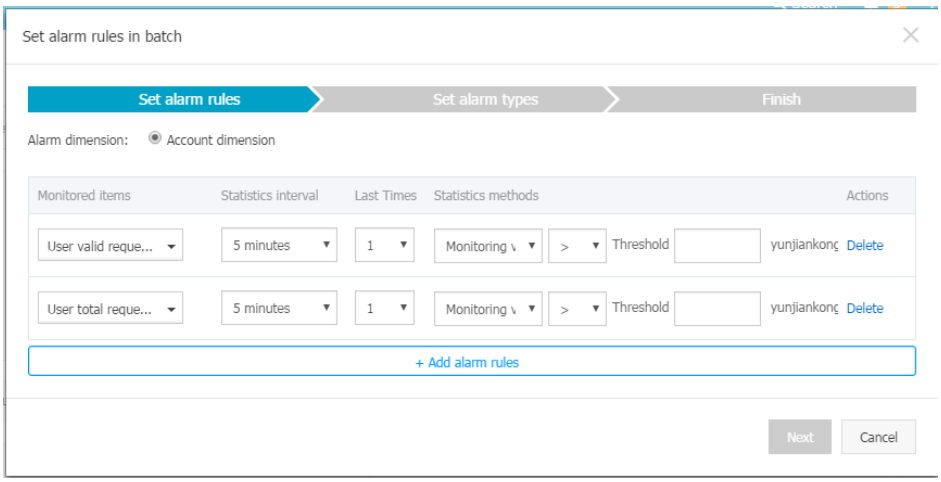
After specifying a bucket in the **Bucket List** Tab, click **Set Alarm Rule** to set an alarm rule.

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.

Note: To learn more about the terms and concepts used below, see the CloudMonitor's **Alarm service overview**.

1. Set the following parameters:

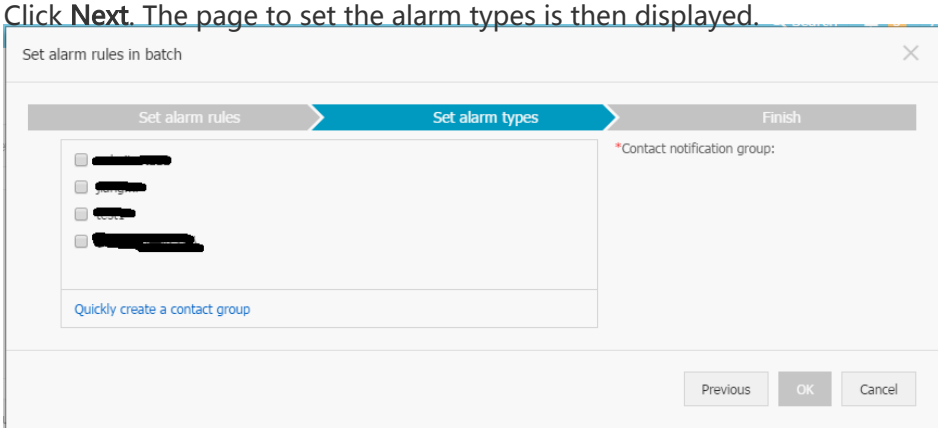


- **Alarm dimension** specifies the monitoring dimension of the alarm rule to set. If the dimension is set to bucket-level, the desired bucket with which to set the alarm rule for must be specified.
- **Monitored items** specifies all the metric items for the selected alarm dimension. You can use the 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**.

Note:

- Click + Add alarm rules to set additional metric item alarm rules.
- Click Delete next to the desired alarm rule to delete it.

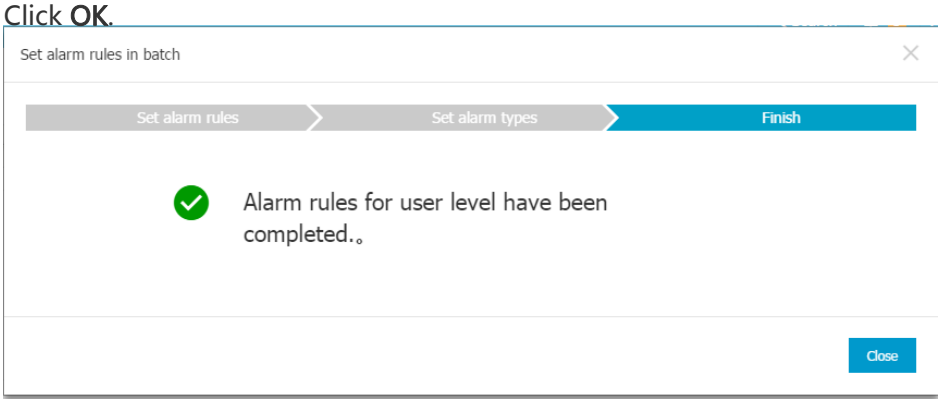
Click **Next**. The page to set the alarm types is then displayed.



If you have set alarm contract groups, they will be displayed on the interface. If you have not set alarm contact groups, click **Quickly create a contact group** and follow the prompts

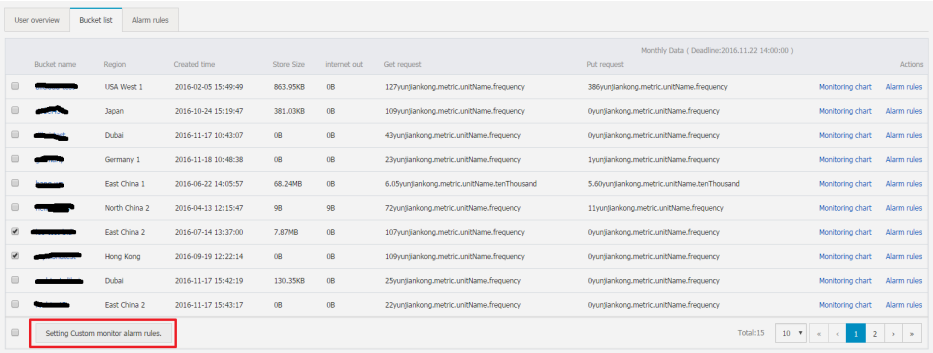
to create a group. For more details, refer to Manage alarm contact.

Click **OK**.



Add alarm rules in the Bucket list

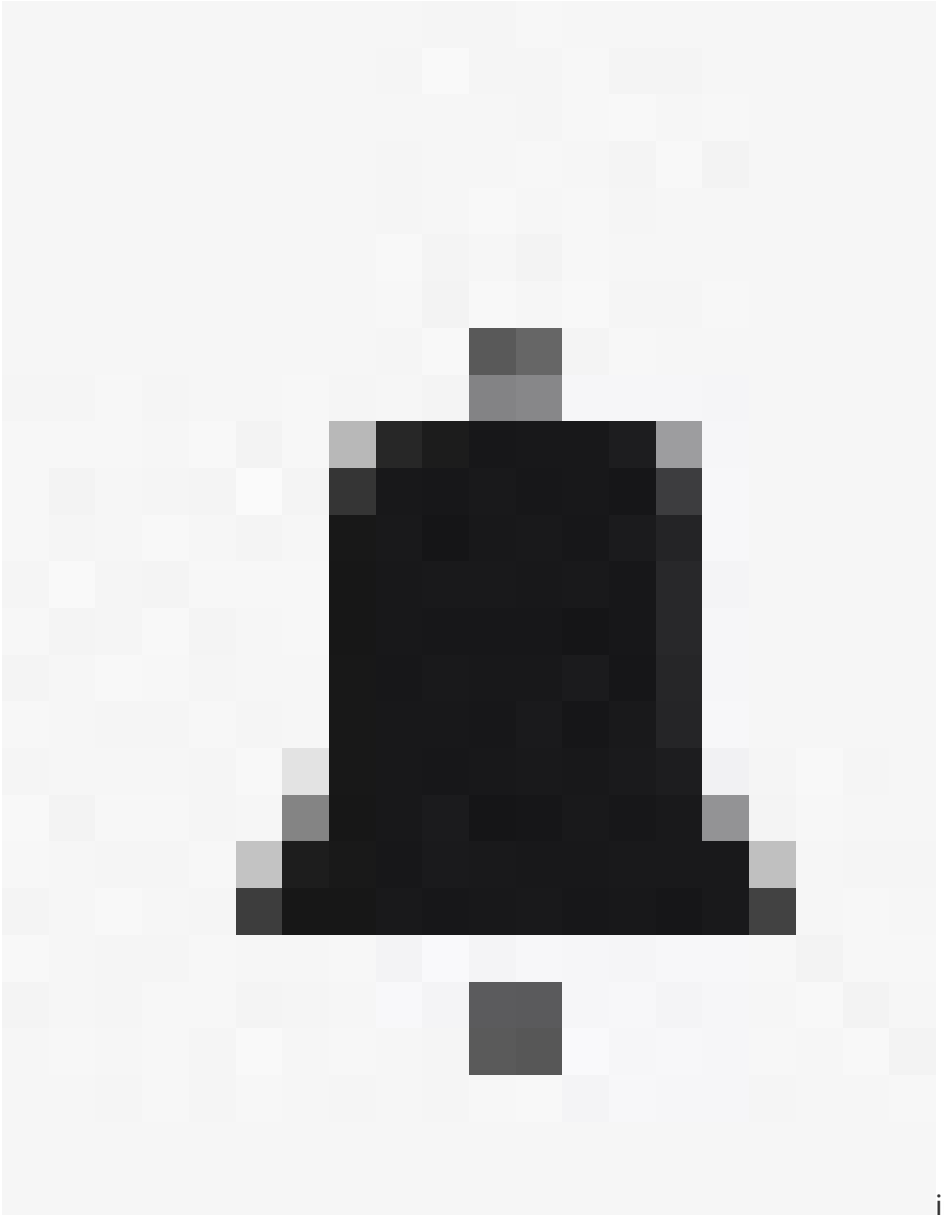
Under the **Bucket list** tab, you can add identical alarm rules for multiple buckets at the same time. Select the desired buckets for which to configure alarm rules and click Set Custom monitor alarm rules to go to the alarm rule settings page previously described in **Add alarm rules**.



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 desired bucket, click


in the top-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 will not be deleted. Before deleting a bucket, it is recommended that you delete any corresponding alarm rules first.

Metric item reference

This chapter provides parameter references to use with the OpenAPI, or the Cloud Monitor 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:

```
QueryMetricRequest request = new QueryMetricRequest();
request.setProject("acs_oss");
```

StartTime and EndTime

The value range of the time parameters for Cloud Monitor 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 Cloud Monitor 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 details about other time parameters, refer to Cloud Monitor API description.

Sample code written by the Java SDK:

```
request.setStartTime("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_bucket_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.setDimensions("{\"BucketName\":\"your_bucket_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	Byte	User level

	inbound traffic		
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
UserServerErrorRate	User-level server-site error request rate	%	User level
UserNetworkErrorCount	User-level network-site error requests	Times	User level
UserNetworkErrorRate	User-level network-site error request rate	%	User level
UserAuthorizationErrorCount	User-level client-site authorization error requests	Times	User level
UserAuthorizationErrorRate	User-level client-site authorization error request rate	%	User level
UserResourceNotFoundErrorCount	User-level client-site error requests indicating resource not found	Times	User level
UserResourceNotFoundErrorRate	User-level client-site error request rate indicating resource not found	%	User level
UserClientTimeoutErrorCount	User-level client-site timeout error request	Times	User level
UserClientTimeoutErrorRate	User-level client-site timeout error request rate	%	User level
UserClientOtherErrorCount	Other user-level client-site error requests	Times	User level
UserClientOtherErrorRate	Other user-level client-site error request rate	%	User level
UserSuccessCount	Successful user-level requests	Times	User level
UserSuccessRate	Successful user-level	%	User level

	request rate		
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
AuthorizationErrorCount	Client-site authorization error requests	Times	Bucket level
AuthorizationErrorRate	Client-site authorization error request rate	%	Bucket level
ResourceNotFoundErrorCount	Client-site error requests indicating	Times	Bucket level

	resource not found		
ResourceNotFoundErrorRate	Client-site error request rate indicating resource not found	%	Bucket level
ClientTimeoutErrorCount	Client-site timeout error requests	Times	Bucket level
ClientTimeoutErrorRate	Client-site timeout 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
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
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
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
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.
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("UserAvailability");
```

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

User-level indicators consist of three monitoring indicator details: current-month metering statistics, service monitoring overview, and request state details.

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.

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: Availability = 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 details about valid requests, refer to the description below.
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
CDN outbound traffic	Byte	Downstream CDN traffic when CDN acceleration service is activated, that is, the back-to-source 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 replication 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

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
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 timeout error requests	Times	Total number of requests whose return status code is 408 or return OSS error code is RequestTimeout
Client-end timeout error requests rate	%	Percentage of requests with client-end timeout 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
Redirect requests	Times	Total number of requests whose return code is 3xx.
Redirect requests rate	%	Percentage of redirect requests in all requests

Bucket-level indicators

Bucket-level indicators are used to monitor OSS operations of specific buckets and are applicable for business scenarios. As well as 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.
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.
Gut 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
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
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 UploadPartCopy
Average server latency of UploadPartCopy requests	Millisecond	Average server latency of successful requests whose request API is UploadPartCopy
Maximum E2E latency of UploadPartCopy requests	Millisecond	Maximum E2E latency of successful requests whose request API is UploadPartCopy
Maximum server latency of	Millisecond	Maximum server latency of

UploadPartCopy requests		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 UploadPartCopy 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

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.

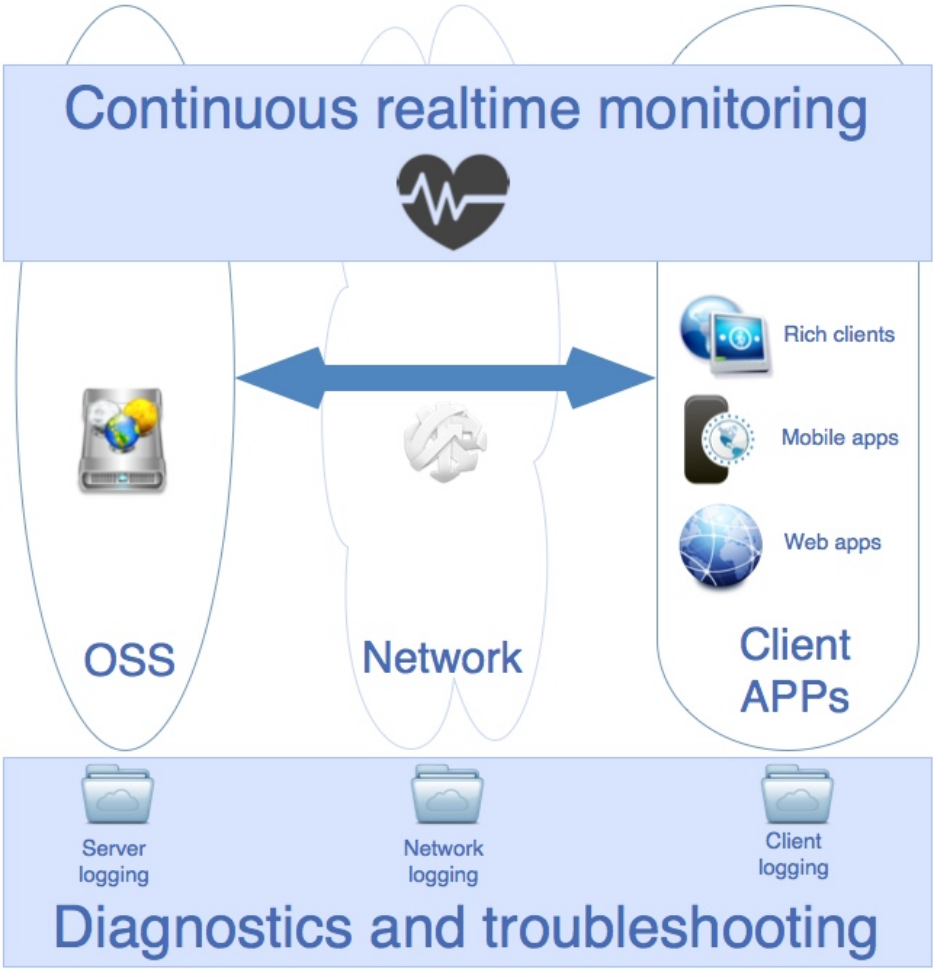
Overview

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, as well as how to associate the relevant information in log files for tracking and diagnosis.
- Troubleshooting: Describes typical problems and corresponding troubleshooting methods.

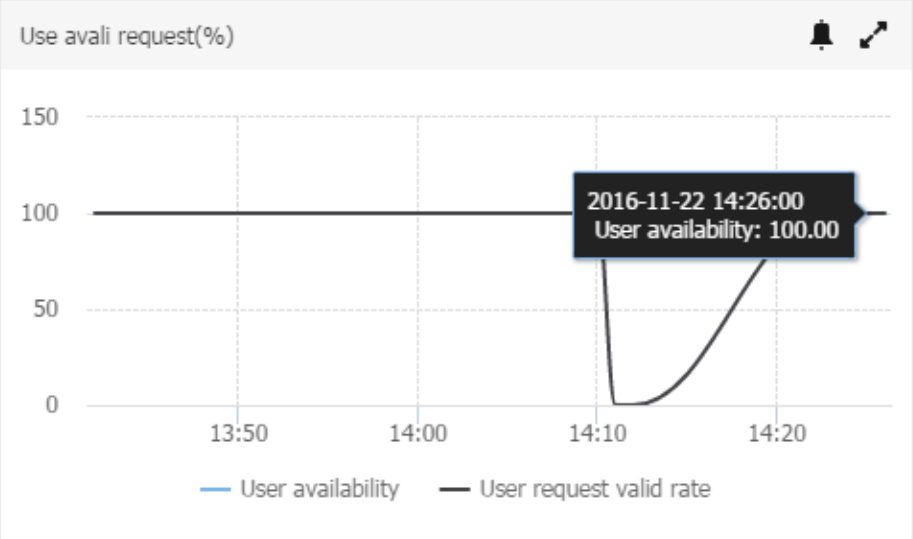


OSS monitoring

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.



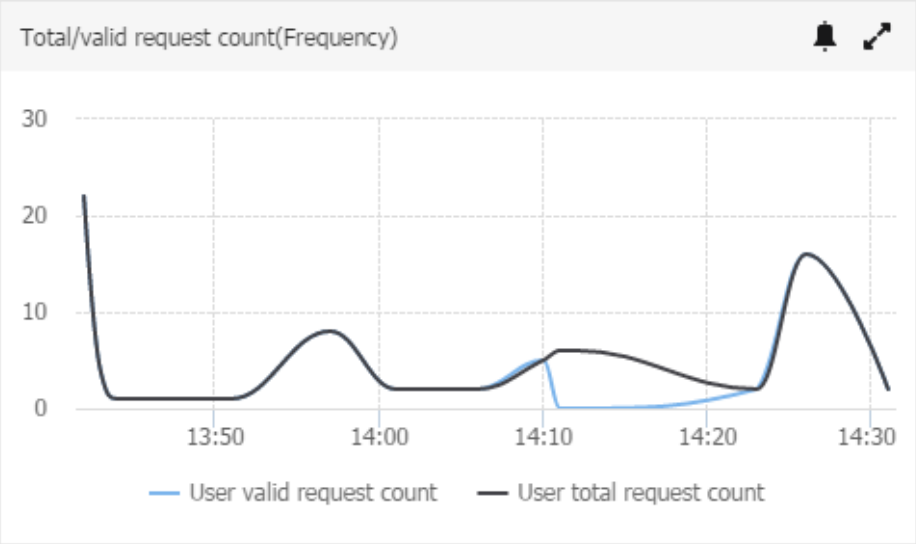
Of course, availability may also temporarily fall below 100% due to system optimization factors, such as partition migration for load balancing. In these cases, OSS SDKs can provide relevant retry mechanisms to handle this type of intermittent failure, keeping the service end unware.

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. Of course, 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 will return a 404 error code (resource does not exist error). This will inevitably produce 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 there are sharp increases or decreases. In such cases, follow-up action is required. You can set alarm rules to ensure you receive prompt notifications. For periodic businesses, you can set periodic alarm rules (periodic alarms will be available soon). For details, see the Alarm Service User Guide.

Request status distribution statistics

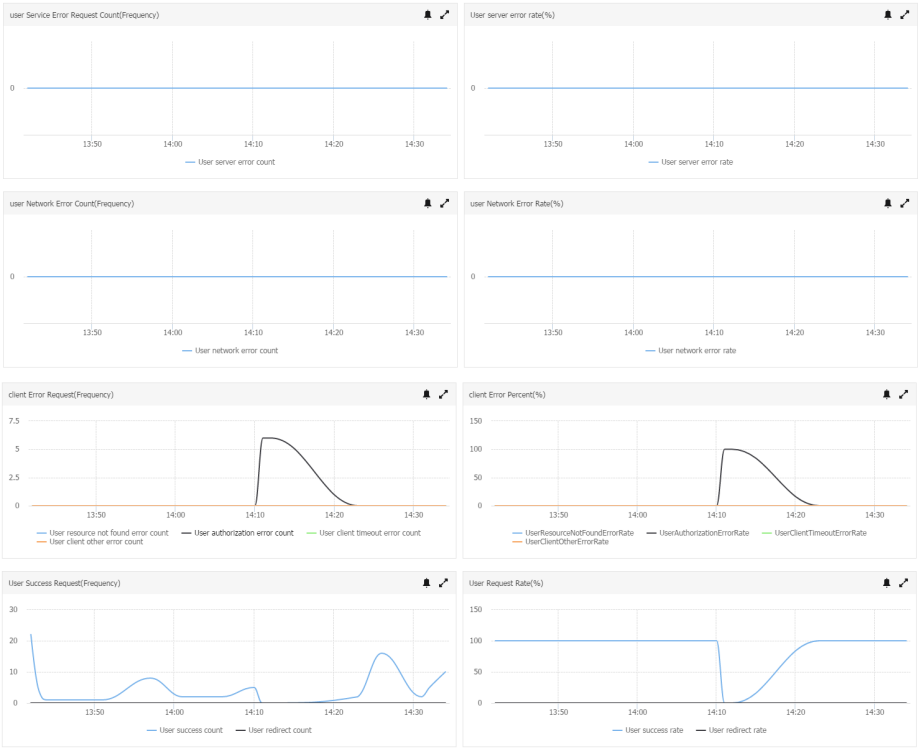
When availability or the valid request rate falls below 100% (or the No. of valid requests is not equal to the total No. of requests), you can look at the request status distribution statistics to quickly determine the request error types. For more information about this metric indicator, see the OSS 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

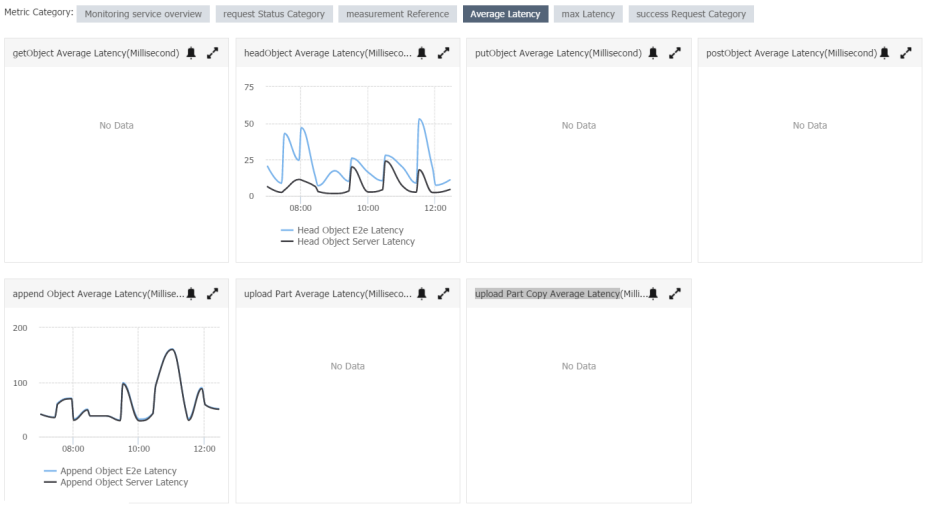
Request status details provides more details about the request monitoring status on the basis of request status distribution statistics. They let you monitor certain types of requests in more detail.



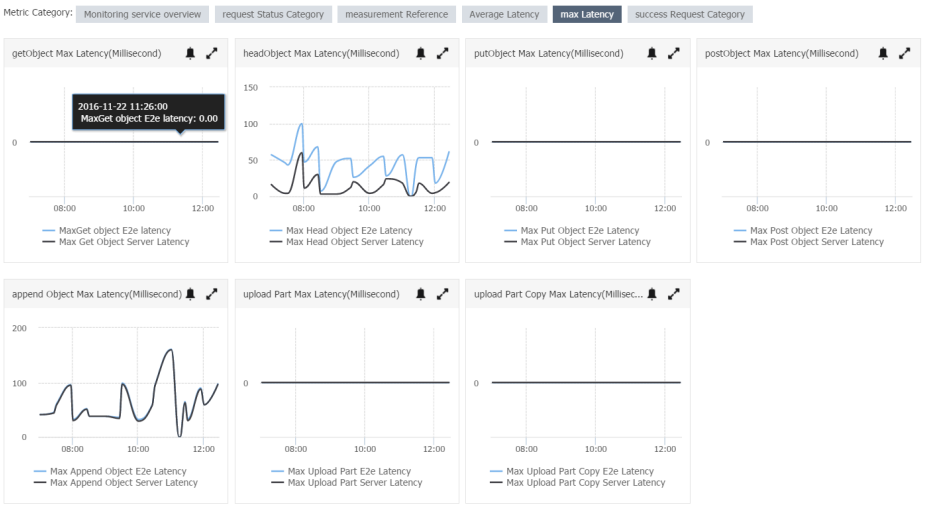
Performance monitoring

The monitoring service provides the following metric items that can be used as indicators for performance monitoring.

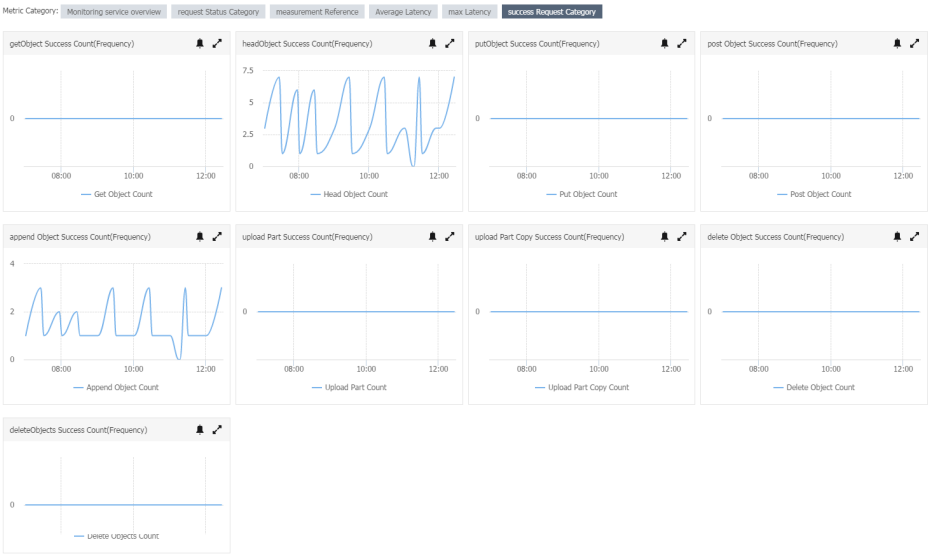
- Average latency
  - E2E average latency
  - Server average latency



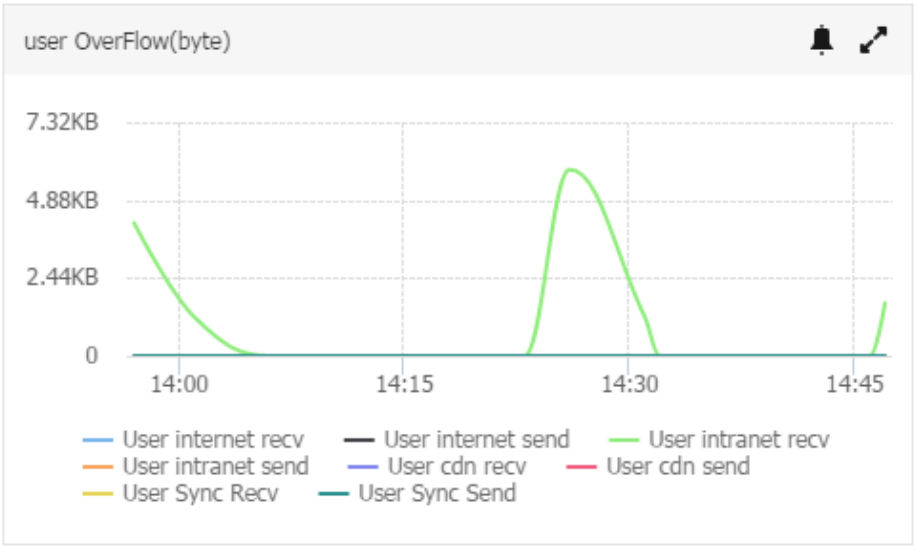
- Maximum latency
  - E2E maximum latency
  - Server maximum latency



- Successful request categories



- Traffic



The metric items above (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, as well as 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 there is a sudden increase in E2E latency but no significant change in server latency, 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 above, ‘successful request operation categories’ also monitors the quantity of requests for the two API operation types below:

- DeleteObject
- DeleteObjects

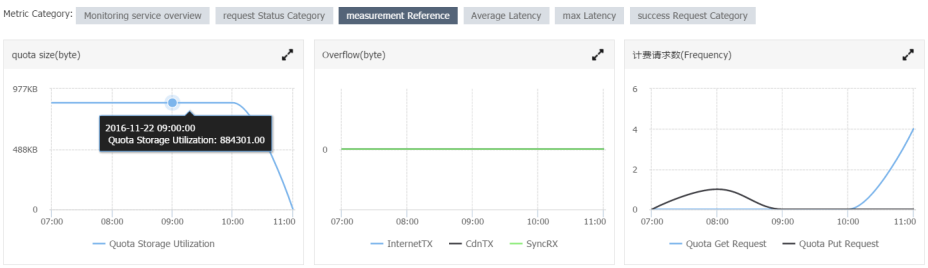
The traffic indicator is used to monitor the overall situation for a user or a specific bucket. It looks at the usage of network resources in Internet, intranet, CDN back-to-source, cross-domain replication, and other such scenarios.

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).

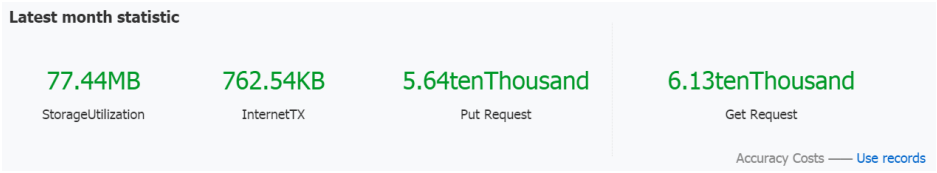
Billing monitoring

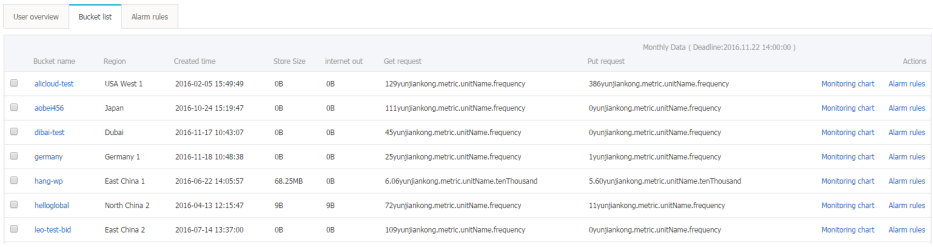
At press time, the OSS monitoring service can only monitor storage space, outbound Internet traffic, Put requests, and Get requests (not including cross-domain replication outbound traffic and CDN outbound traffic). It does not support alarm setting or OpenAPI read operations for billing data.

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 figure below:



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 will increase your understanding of your resource usage and computation fees for the current month in real time, as shown below:





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 there is a performance problem. 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 should find detailed information in the relevant logs to help you further diagnose and troubleshoot any faults.

In the "Troubleshooting" section below, we will 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 will 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 will 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, refer to Set logging.

For more details on Log Service naming rules and record formats, refer to Set access logging.

Using network logging tools

In many situations, you can diagnose problems simply 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.

For more detailed information on Wireshark operations, refer to the 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 should 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 there are two possible causes of high E2E latency and low server latency:

- Slow client application response speed
- Network factors

Investigate client performance problems

There are several possible causes of a slow client application response speed:

Limited number of available connections or threads

- The following method can be used to solve available connection quantity issues:
  a. Use the relevant command to check if the system has a large number of connections in the TIME_WAIT status.
  b. If yes, adjust the core parameters to solve this problem.
- When there is a limited number of available threads, first check for bottlenecks affecting the client CPU, memory, network, or other resources. If there are none, increase the number of concurrent threads properly.
- If the problem persists, you will 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: Refer to 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 use the method below to determine if the client has a retry problem:

  Check the client log. The detailed log entries will 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 there is no problem with the client, 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 there is a high server latency during downloads or uploads, 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 will also reduce your traffic fees. In the case of upload, you may consider revoking write permissions for this object (bucket), as long as this will 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 will help you solve the problem.

Server errors

When there is an increase in server-side errors, there are two scenarios to consider:

- 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 will avoid temporary service unavailability due to system optimization, upgrades, and other such operations (such as partition migration for system load balancing), but will also avoid high pressure during business peaks.

- Permanent increase

When there is a sustained increase in the number of server-side errors, please provide our back-end staff with the RequestIDs in your client logs or in the logs recorded by the logging function, and they will 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 need to 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.
    i. 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 need to confirm the bucket’ s region and then correct the domain name information.

ii. If you have activated the CDN acceleration service, this problem may occur when CDN binds an incorrect back-to-source domain name. In this case, check that the CDN back-to-source 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, refer to 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 need to 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 need to 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 need to 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 login 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 will receive the value 'false'. However, the server will actually produce 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 will initiate a retry. This second request will then produce 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 timeout 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 refer to 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 there is an abnormal increase in storage capacity 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:

i. Check if the valid request rate has decreased, because a failed delete request may cause storage objects to fail to be deleted as expected.
ii. 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, simply 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 preceding troubleshooting sections did not cover your problem, use one of the following methods to diagnose and troubleshoot the problem.

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.

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.

If the information in the server log is insufficient, use the client long to investigate the client application, or use a network tool such as Wireshark to check your network for problems.