

Alibaba Cloud Object Storage Service

Developer Guide

Issue: 20181220

Legal disclaimer

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

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

secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).

6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

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

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Usage instructions.....	1
2 Endpoint.....	2
2.1 Endpoints.....	2
2.2 Regions and endpoints.....	4
3 Signature.....	9
3.1 ACL verification.....	9
3.2 Add a signature to the header.....	13
3.3 Add a signature to a URL.....	20
4 Access and control.....	23
4.1 Access control.....	23
4.2 Tutorial: Control access to buckets and folders.....	34
5 Access OSS.....	57
5.1 OSS-based app development.....	57
5.2 Quick start.....	61
6 Manage buckets.....	62
6.1 Set bucket read and write permissions.....	62
6.2 Obtain bucket region information.....	63
6.3 View the bucket list.....	63
6.4 Set a WORM strategy.....	63
6.5 Bind a custom domain name.....	65
6.6 Delete a bucket.....	66
7 Upload files.....	67
7.1 Simple upload.....	67
7.2 Form upload.....	68
7.3 Multipart upload.....	71
7.4 Append object.....	74
7.5 Authorized third-party upload.....	76
7.6 Upload callback.....	77
8 Download files.....	79
8.1 Simple download.....	79
8.2 Multipart download.....	79
8.3 Authorized third-party download.....	80
9 Manage files.....	82
9.1 Object Meta.....	82
9.2 View the object list.....	83
9.3 Copy an object.....	86

9.4 Delete an object.....	87
9.5 Manage object lifecycle.....	87
9.6 Cross-region replication.....	90
9.7 Manage back-to-origin settings.....	92
10 Security management.....	96
10.1 Anti-leech settings.....	96
10.2 Cross-origin resource sharing.....	97
10.3 Server-side encryption.....	98
10.4 Set security token.....	102
11 Static website hosting.....	107
11.1 Configure static website hosting.....	107
11.2 Tutorial: Host a static website using a custom domain name.....	109
12 Monitoring service.....	117
12.1 Monitoring service overview.....	117
12.2 Monitoring service user guide.....	119
12.3 Alarm service user guide.....	131
12.4 Metric item reference	136
12.5 Monitoring indicators reference.....	145
12.6 Service monitoring, diagnosis, and troubleshooting.....	154
13 Cloud data processing.....	171

1 Usage instructions

If it is your first time using Alibaba Cloud OSS, see the [OSS Quick Start Guide](#) to quickly get started with OSS.

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

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

2 Endpoint

2.1 Endpoints

Composition rules for domain names

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

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

Access OSS through the Internet

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

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

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

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

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

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

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

```

```

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

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

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

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

Access OSS through an intranet

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

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

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

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

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

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

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

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

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



Note:

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

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

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

2.2 Regions and endpoints

Regions and OSS endpoints in classic networks

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

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

Region name	OSS region	Internet endpoint	Internet endpoint protocol	Intranet endpoint for ECS access	Intranet endpoint protocol
China North 5 (Hohhot)	oss-cn-huhehaote	oss-cn-huhehaote.aliyuncs.com	HTTP and HTTPS	oss-cn-huhehaote-internal.aliyuncs.com	HTTP and HTTPS
China South 1 (Shenzhen)	oss-cn-shenzhen	oss-cn-shenzhen.aliyuncs.com	HTTP and HTTPS	oss-cn-shenzhen-internal.aliyuncs.com	HTTP and HTTPS
Hong Kong	oss-cn-hongkong	oss-cn-hongkong.aliyuncs.com	HTTP and HTTPS	oss-cn-hongkong-internal.aliyuncs.com	HTTP and HTTPS
US West 1 (Silicon Valley)	oss-us-west-1	oss-us-west-1.aliyuncs.com	HTTP and HTTPS	oss-us-west-1-internal.aliyuncs.com	HTTP and HTTPS
US East 1 (Virginia)	oss-us-east-1	oss-us-east-1.aliyuncs.com	HTTP and HTTPS	oss-us-east-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	oss-ap-southeast-1.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-2-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 3 (Kuala Lumpur)	oss-ap-southeast-3	oss-ap-southeast-3.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-3-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 5 (Jakarta)	oss-ap-southeast-5	oss-ap-southeast-5.aliyuncs.com	HTTP and HTTPS	oss-ap-southeast-5-internal.aliyuncs.com	HTTP and HTTPS

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

**Note:**

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

Regions and OSS endpoints in VPC networks

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

Region name	OSS region	Endpoint in VPC networks	Protocol
China East 1 (Hangzhou)	oss-cn-hangzhou	oss-cn-hangzhou-internal.aliyuncs.com	HTTP and HTTPS
China East 2 (Shanghai)	oss-cn-shanghai	oss-cn-shanghai-internal.aliyuncs.com	HTTP and HTTPS
China North 1 (Qingdao)	oss-cn-qingdao	oss-cn-qingdao-internal.aliyuncs.com	HTTP and HTTPS
China North 2 (Beijing)	oss-cn-beijing	oss-cn-beijing-internal.aliyuncs.com	HTTP and HTTPS
China North 3 (Zhangjiakou)	oss-cn-zhangjiakou	oss-cn-zhangjiakou-internal.aliyuncs.com	HTTP and HTTPS
China North 5 (Hohhot)	oss-cn-huhehaote	oss-cn-huhehaote-internal.aliyuncs.com	HTTP and HTTPS
China South 1 (Shenzhen)	oss-cn-shenzhen	oss-cn-shenzhen-internal.aliyuncs.com	HTTP and HTTPS
Hong Kong	oss-cn-hongkong	oss-cn-hongkong-internal.aliyuncs.com	HTTP and HTTPS
US West 1 (Silicon Valley)	oss-us-west-1	oss-us-west-1-internal.aliyuncs.com	HTTP and HTTPS
US East 1 (Virginia)	oss-us-east-1	oss-us-east-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 1 (Singapore)	oss-ap-southeast-1	oss-ap-southeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 2 (Sydney)	oss-ap-southeast-2	oss-ap-southeast-2-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 3 (Kuala Lumpur)	oss-ap-southeast-3	oss-ap-southeast-3-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SE 5 (Jakarta)	oss-ap-southeast-5	oss-ap-southeast-5-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific NE 1 (Tokyo)	oss-ap-northeast-1	oss-ap-northeast-1-internal.aliyuncs.com	HTTP and HTTPS
Asia Pacific SOU 1 (Mumbai)	oss-ap-south-1	oss-ap-south-1-internal.aliyuncs.com	HTTP and HTTPS

Region name	OSS region	Endpoint in VPC networks	Protocol
EU Central 1 (Frankfurt)	oss-eu-central-1	oss-eu-central-1-internal.aliyuncs.com	HTTP and HTTPS
UK (London)	oss-eu-west-1	oss-eu-west-1-internal.aliyuncs.com	HTTP and HTTPS
Middle East 1 (Dubai)	oss-me-east-1	oss-me-east-1-internal.aliyuncs.com	HTTP and HTTPS

Usage of endpoints

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

3 Signature

3.1 ACL verification

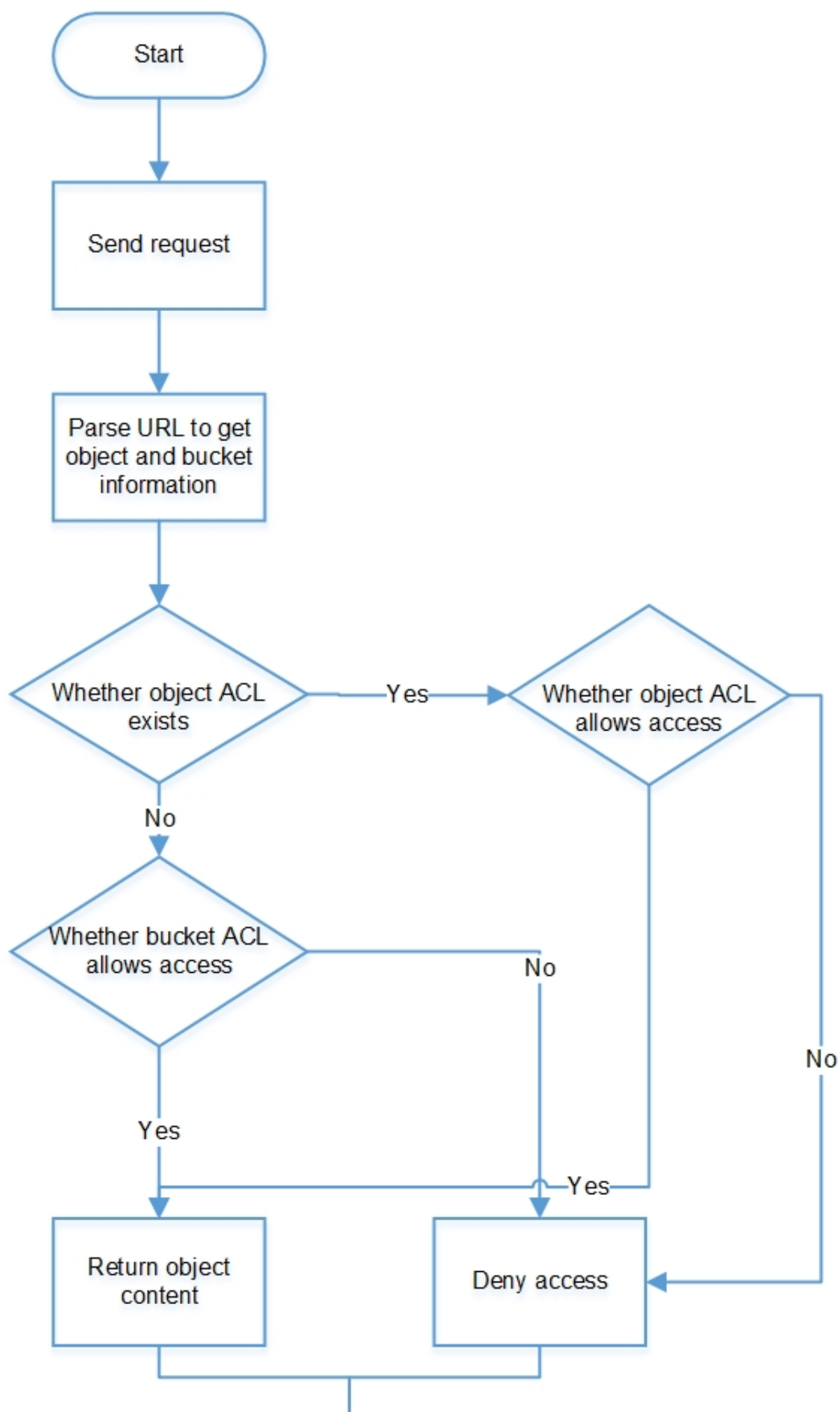
Security of OSS access

Based on whether the identity authentication information is included or not, HTTP requests sent to OSS are divided into two types: requests with identity authentication information and anonymous requests without identity authentication information. The identity authentication information in requests comes in two forms:

- Authorization contained in the request header, in the format of OSS + AccessKeyId + signature string
- OSS AccessKeyId and a signature contained in the request URL

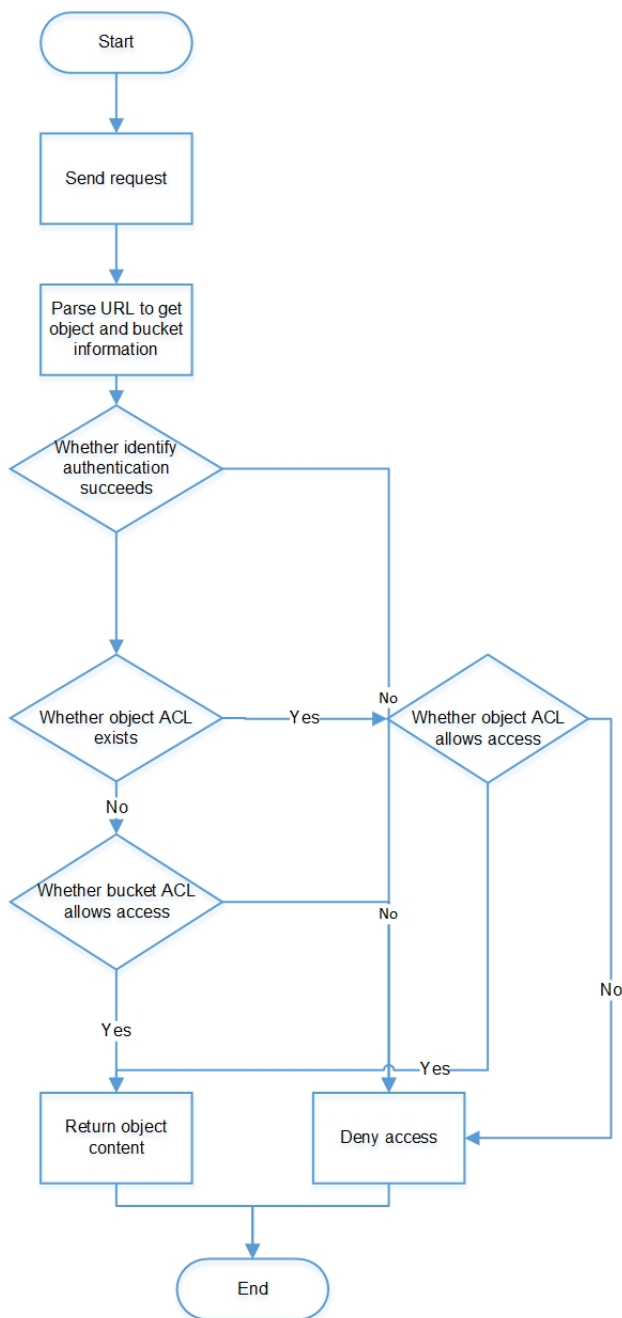
OSS access process

Access process for anonymous requests



1. The user's request is sent to the HTTP server of OSS.
2. OSS parses the URL to get the bucket and object information.
3. OSS checks whether the object has ACL settings.
 - If no ACL is set for the object, go to step 4.
 - If an ACL is set for the object, OSS checks whether the ACL permits anonymous access.
 - If the ACL permits anonymous access, go to step 5.
 - If not, the request is rejected and the process ends.
4. OSS checks whether the bucket ACL permits anonymous access.
 - If the ACL permits anonymous access, go to step 5.
 - If not, the request is rejected and the process ends.
5. The request passes the access verification, and the object content is returned to the user.

Access process for requests with identity authentication information



1. The user's request is sent to the HTTP server of OSS.
2. OSS parses the URL to get the bucket and object information.
3. Based on the AccessKeyId of the request, OSS obtains the identity information of the request sender to perform authentication.
 - If the information is not obtained, the request is rejected and the process ends.
 - If the information is obtained, but the request sender is not permitted to access the resource , the request is rejected and the process ends.

- If the information is obtained, but the signature calculated based on the request HTTP parameters does not match the signature contained in the request, the request is rejected and the process ends.
 - If the authentication succeeds, go to step 4.
4. OSS checks whether the object has ACL settings.
- If no ACL is set for the object, go to step 5.
 - If an ACL is set for the object, OSS checks whether the object ACL permits the user's access.
 - If the ACL permits the user's access, go to step 6.
 - If not, the request is rejected and the process ends.
5. OSS checks whether the bucket ACL permits the user's access.
- If the ACL permits the user's access, go to step 6.
 - If not, the request is rejected and the process ends.
6. The request passes the access verification, and the object content is returned to the user.

Three methods for OSS access with identity authentication information

- Access OSS by using the console: The identity authentication process is invisible to users in the console. When users access OSS through the console, they do not have to concern about the details of this process.
- Access OSS by using SDKs: OSS provides SDKs for multiple programming languages. A signature algorithm is implemented in an SDK, where users only need to input the AK information.
- Access OSS by using APIs: If you want to write your own code to call the RESTful APIs, you must implement a signature algorithm to calculate the signature. For more information about the signature algorithm, see [Add a signature to the header](#) and [Add a signature to the URL](#) in the *OSS API Reference*.

For more information about AccessKey and identity authentication, see [Access control](#).

3.2 Add a signature to the header

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

Calculation of the Authorization field

```
Authorization = "OSS " + AccessKeyId + ":" + Signature
```

```
Signature = base64(hmac-sha1(AccessKeySecret,  
    VERB + "\n"  
    + Content-MD5 + "\n"  
    + Content-Type + "\n"  
    + Date + "\n"  
    + CanonicalizedOSSHeaders  
    + CanonicalizedResource))
```

- The `AccessKeySecret` indicates the key required for a signature.
- `VERB` indicates the HTTP request method, including PUT, GET, POST, HEAD, and DELETE.
- `\n` is a line break.
- `Content-MD5` The Content-MD5 is the MD5 value of requested content data. The message content (excluding the header) is calculated to obtain an MD5 value, which is a 128-bit number. This number is encoded with Base64 into a Content-MD5 value. The request header can be used to check the message validity, that is, whether the message content is consistent with the sent content, such as "eB5eJF1ptWaXm4bijSPyxw==". The request header may be empty. For more information, see [RFC2616 Content-MD5](#).
- `Content-Type` indicates the requested content type, such as "application/octet-stream". It content type may be empty.
- `Date` indicates the time that the operation takes. It must be in GMT format, such as "Sun, 22 Nov 2015 08:16:38 GMT".
- The `CanonicalizedOSSHeaders` indicates an assembly of HTTP headers whose prefixes are "x-oss-".
- The `CanonicalizedResource` indicates the OSS resource that the user wants to access.

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

Construct CanonicalizedOSSHeaders

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

1. Convert the names of all HTTP request headers whose prefixes are x-oss- into lowercase letters. For example, convert `X-OSS-Meta-Name:TaoBao` to `x-oss-meta-name:TaoBao`.
2. If the request is sent with the `AccessKeyID` and `AccessKeySecret` obtained by the STS, you must also add the obtained security-token value to the signature string in the form of `x-oss-security-token:security-token`.
3. Sort all acquired HTTP request headers in a lexicographically ascending order.

4. Delete any space on either side of a separator between the request header and content. For example, convert `x-oss-meta-name: TaoBao` to `x-oss-meta-name:TaoBao`.
5. Separate all the content and headers with the `\n` separator to form the final CanonicalizedOSSHeaders.

**Note:**

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

Construct CanonicalizedResource

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

1. Set CanonicalizedResource into a null character string (`""`);
2. Add the OSS resource to be accessed in the following format: `/BucketName/ObjectName`. (If ObjectName does not exist, CanonicalizedResource is `/BucketName/`. If BucketName does not exist either, CanonicalizedResource is `/`.)
3. If the requested resource includes sub-resources (SubResource), sort all the sub-resources in a lexicographically ascending order and separate the sub-resources using the separator `&` to generate a sub-resource string. Add `?` and the sub-resource string to the end of the CanonicalizedResource string. In this case, CanonicalizedResource is like: `/BucketName/ObjectName?acl&uploadId=UploadId`
4. If the user request specifies the query string (QueryString, also called HTTP Request Parameters), sort these query strings and request values in a lexicographically ascending order, separate the query strings and request values using the separator `&`, and add them to CanonicalizedResource based on the parameters. In this case, CanonicalizedResource is like: `/BucketName/ObjectName?acl&response-content-type=ContentType&uploadId=UploadId`.

**Note:**

- The sub-resources supported by OSS currently include: `acl`, `uploads`, `location`, `cors`, `logging`, `website`, `referer`, `lifecycle`, `delete`, `append`, `tagging`, `objectMeta`, `uploadId`, `partNumber`, `security-token`, `position`, `img`, `style`, `styleName`, `replication`, `replicationProgress`, `replicationLocation`,

cname, bucketInfo, comp, qos, live, status, vod, startTime, endTime, symlink, x-oss-process, response-content-type, response-content-language, response-expires, response-cache-control, response-content-disposition, and response-content-encoding.

- Three types of sub-resources are available:
 - Resource identifiers, such as acl, append, uploadId, and symlink sub-resources. For more information, see [Bucket-related operations](#) and [Object-related operations](#).
 - Specify response header fields such as response-***. For more information, see the Request Parameters section of [GetObject](#).
 - Object handling methods, such as x-oss-process. It is used as the object handling method, such as [Image Processing](#).

Rules to calculate a signature header

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

Example signature

Assume that AccessKeyId is 44CF9590006BF252F707 and AccessKeySecret is OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV.

Request	Signature string calculation formula	Signature string
PUT /nelson HTTP/1.0 Content-MD5: eB5eJF1ptW aXm4bijSPyxw== Content- Type: text/html Date: Thu, 17 Nov 2005 18:49:58 GMT Host: oss-example.oss-cn-hangzhou .aliyuncs.com X-OSS-Meta- Author: foo@bar.com X-OSS- Magic: abracadabra	Signature = base64(hmac- sha1(AccessKeySecret, VERB + "\n" + Content-MD5 + "\n" + Content-Type + "\n" + Date + "\n" + CanonicalizedOSSHead ers+ CanonicalizedResource))	"PUT\n eB5eJF1ptW aXm4bijSPyxw==\n text/ html\n Thu, 17 Nov 2005 18 :49:58 GMT\n x-oss-magic :abracadabra\n x-oss-meta- author:foo@bar.com\n /oss- example/nels

The signature calculation method is as follows:

Python sample code:

```
import base64
import hmac
import sha
h = hmac.new("OtxrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
             "PUT\n0DBG0ERFMDMzQTczRUY3NUE3NzA5QzdFNUYzMDQxNEM=\ntext\n/html\nThu, 17 Nov 2005 18:49:58 GMT\nx-oss-magic:abracadabra\nx-oss-meta-author:foo@bar.com\n/oss-example/nelson", sha)
Signature = base64.b64encode(h.digest())
print("Signature: %s" % Signature)
```

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

```
PUT /nelson HTTP/1.0
Authorization:OSS 44CF9590006BF252F707:26NBxoKdsyly4EDv6inkoDft/yA=
Content-Md5: eB5eJF1ptWaXm4bijSPyxw==
Content-Type: text/html
Date: Thu, 17 Nov 2005 18:49:58 GMT
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
X-OSS-Meta-Author: foo@bar.com
X-OSS-Magic: abracadabra
```

Detail analysis

- If the input AccessKeyID does not exist or is inactive, the error 403 Forbidden is returned. Error code: InvalidAccessKeyId.
- If the authorization value format in the user request header is incorrect, the error 400 Bad Request is returned. Error code: InvalidArgument.

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

```
<? xml version="1.0" ? >
<Error>
  <Code>
    SignatureDoesNotMatch
  </Code>
  <Message>
    The request signature we calculated does not match the
    signature you provided. Check your key and signing method.
  </Message>
  <StringToSignBytes>
    47 45 54 0a 0a 0a 57 65 64 2c 20 31 31 20 4d 61 79 20 32 30 31
    31 20 30 37 3a 35 39 3a 32 35 20 47 4d 54 0a 2f 75 73 72 65 61 6c 74
    65 73 74 3f 61 63 6c
  </StringToSignBytes>
  <RequestId>
    1E446260FF9B10C2
  </RequestId>
  <HostId>
    oss-cn-hangzhou.aliyuncs.com
  </HostId>
  <SignatureProvided>
    y5H7yzPsA/tP4+0tH1HHvPEwUv8=
  </SignatureProvided>
  <StringToSign>
    GET
    Wed, 11 May 2011 07:59:25 GMT
    /oss-example? acl
  </StringToSign>
  <OSSAccessKeyId>
    AKIAIVAKMSMOY7V0MRWQ
  </OSSAccessKeyId>
</Error>
```

**Note:**

OSS SDK has implemented the signature. You do not need to worry about the signature issue when you use the OSS SDK. To learn more about the signature implementations of specific

languages, see the OSS SDK code. The files for implementing OSS SDK signature are shown in the following table:

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

Content-MD5 calculation method

Content-MD5 calculation

The message content "123456789" is used as an example. The Content-MD5 value of the string is calculated as follows:

The algorithm defined in related standards can be simplified to the following:

Calculate the MD5-encrypted 128-bit binary array.

Encode the binary array (instead of the 32-bit string code) with Base64.

Python is used as an example.

The correct calculation code is:

```
>>> import base64,hashlib
>>> hash = hashlib.md5()
>>> hash.update("0123456789")
>>> base64.b64encode(hash.digest())
'eB5eJF1ptWaXm4bijSPyxw=='
```

Note:

The correct code is: `hash.digest()`, used to calculate a 128-bit binary array

```
>>> hash.digest()
'\x\x1e^\$]i\xb5f\x97\x9b\x86\xe2\x8d#\xf2\xc7'
```

The common error is to base 64 the computed 32-Bit String encoding directly.

An incorrect example: `hash.hexdigest()`, and a visible 32-bit string is calculated.

```
>>> hash.hexdigest()
'781e5e245d69b566979b86e28d23f2c7'
```

Result of encoding the incorrect MD5 value with Base64:

```
>>> base64.b64encode(hash.hexdigest())
```

```
'NzgxZTVlMjQ1ZDY5YjU2Njk3OWI4NmUyOGQyM2YyYzc='
```

3.3 Add a signature to a URL

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

Implementation

URL signature example:

```
http://oss-example.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?OSSAccessKeyId=nz2pc56s936**9l&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

The URL signature must include at least the following three parameters: **Signature**, **Expires**, and **OSSAccessKeyId**.

- The **Expires** parameter indicates the time-out period of a URL. The value of this parameter is UNIX time (which is the number of seconds that have elapsed since 00:00:00 UTC, January 1, 1970. For more information, see [Wikipedia](#)). If the time when OSS receives the URL request is later than the value of the Expires parameter and is included in the signature, an error code request timed-out is returned. For example, if the current time is 1141889060, to create a URL that is scheduled to expire in 60 seconds, you can set the value of Expires to 1141889120.
- **OSSAccessKeyId** refers to the AccessKeyID in the key.
- **Signature** indicates the signature information. For all requests and header parameters that OSS supports, the algorithm for adding a signature to a URL is basically the same as that of [Adding a signature to a header](#).

```
Signature = urlencode(base64(hmac-sha1(AccessKeySecret,
    VERB + "\n"
    + CONTENT-MD5 + "\n"
    + CONTENT-TYPE + "\n"
    + EXPIRES + "\n"
    + CanonicalizedOSSHeaders
    + CanonicalizedResource)))
```

The difference is listed as follows:

- When a signature is added to a URL, the Expires parameter replaces the Date parameter.
- Signatures cannot be included in a URL and the Header at the same time.
- If more than one incoming Signature, Expires, or AccessKeyId value is available, the first of each incoming value is used.

- Whether the request time is later than the Expires time, is verified first before verifying the signature.
- When you put the signature string into a URL, remember to perform the UriEncode for a URL.
- When you add a signature to a temporary user URL, the `security-token` must also be entered. The format is as follows:

```
http://oss-example.oss-cn-hangzhou.aliyuncs.com/oss-api.pdf?
OSSAccessKeyId=nz2pc56s936**9l&Expires=1141889120&Signature=
vjbyPxybdZaNmGa%2ByT272YEAiv4%3D&security-token=SecurityToken
```

Sample code

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

```
import base64
import hmac
import sha
import urllib
h = hmac.new("0txrzxIsfpFjA7SwPzILwy8Bw21TLhquhboDYROV",
             "GET\n\n1141889120\n/oss-example/oss-api.pdf",
             sha)
urllib.quote (base64.encodestring(h.digest()).strip())
```

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

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

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

Detail analysis

- If you adopt the approach of adding a signature to a URL, the authorized data is exposed on the Internet before the authorization period expires. We recommend that you must assess the usage risks in advance.
- The PUT and GET requests both support adding a signature in a URL.
- When a signature is added to a URL, the sequence of Signature, Expires, and AccessKeyId can be swapped. If one or more Signature, Expires, or AccessKeyId parameter is missing, the error 403 Forbidden is returned. Error code: AccessDenied.
- If the current access time is later than the Expires time set in the request, the error 403 Forbidden is returned. Error code: AccessDenied.
- If the format of the Expires time is incorrect, the error 403 Forbidden is returned. Error code: AccessDenied.
- If the URL includes one or more Signature, Expires, or AccessKeyId parameter and the header also includes signature information, the error 400 Bad Request is returned. Error code: InvalidArgument.
- When the signature string is generated, the Date parameter is replaced by the Expires parameter, but the headers such as content-type and content-md5 defined in the preceding section are still included. (Though the Date request header still exists in the request, you can skip adding it to the signature string.)

4 Access and control

4.1 Access control

Send an OSS access request

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

- According to the role of the visitors, the access to OSS resources is classified into owner access and third-party access. Here, the owner refers to the bucket owner, also known as the "developer". Third-party users are any users who access resources in a bucket.
- According to the identity of visitors, the access to OSS resources is classified into anonymous access and signature-based access. In the OSS, a request that does not contain any identification information is considered anonymous access. According to the rules in the OSS API documentation, signature-based access refers to the requests that contain signature information in the request header or URL.

Types of AccessKeys

Currently, the three types of [AccessKey](#) for OSS access are as follows:

- Alibaba Cloud account AccessKeys

These are the AccessKeys of the bucket owners. The AccessKey provided by each Alibaba Cloud account has full access to its own resources. Each Alibaba Cloud account can simultaneously have zero to five active or inactive AccessKey pairs (AccessKeyID and AccessKeySecret).

You can add or delete AccessKey pairs in the [AccessKey console](#).

Each AccessKey pair is either active or 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.

**Note:**

The AccessKey of the Alibaba Cloud account must not be used directly unless it is required.

- RAM account AccessKeys

Resource Access Management (RAM) is a resource access control service provided by Alibaba Cloud. RAM account AKs are the AccessKeys 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 grant limited permissions to RAM users so that they have only the read permission on a bucket. RAM accounts are subordinate to the Alibaba Cloud account and cannot own any actual resources. All resources belong to the Alibaba Cloud account.

- STS account AccessKeys

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

Implementation of identity authentication

The three methods of authentication are:

- AK authentication
- RAM authentication
- STS authentication

Before sending a request to OSS as an individual identity, a user must generate a signature string for the request according to the format specified by OSS, and then encrypt the signature string using the AccessKeySecret to generate a verification code.

After receiving the request, 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 similar to the verification code provided, the request is assumed valid. If not, OSS rejects the request and returns an HTTP 403 error.

Users can directly use the SDKs provided by OSS with different AccessKeys for different types of identity authentication.

Permission control

OSS provides the following permission control to access its stored objects:

- Bucket-level permissions
- Object-level permissions

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

Bucket-level permissions

- Bucket permission types

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 must be borne by the owner of the bucket. Use this permission with caution.
public-read	Only the owner of the 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 the bucket can read, write, and delete the objects in the bucket. Other people cannot access the objects in the bucket without authorization.

- Set and get bucket ACL
 - API: [PutBucketACL](#)
 - SDK: Java SDK - [Manage a bucket](#)
 - Console: [Create a bucket](#)
 - API: [GetBucketACL](#)
 - SDK: Java SDK - [Manage a bucket](#)

Object-level permissions

- Object permission types

OSS ACL 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	Anyone can read and write the object.
public-read	Anyone can read the object, but only the object owner can write the object.
private	Only the object owner can read and write the object. Other people have no permission to operate the object.
default	The object inherits the permission of the bucket.

**Note:**

- If no ACL is configured for an object, the object uses the default ACL, indicating that the object has the same ACL as the bucket where the object is stored.
 - If an ACL is configured for an object, the object ACL 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.
- Set and get object ACL
 - API: [PutObjectACL](#)
 - SDK: [Set the object ACL](#) in Java SDK
 - API: [GetObjectACL](#)
 - SDK: [Get the object ACL](#) in Java SDK

Account-level permissions (RAM)

- Scenario

If you have purchased cloud resources and multiple users in your organization use them, these users have to share the AccessKey of your Alibaba Cloud account. However, the following issues may occur:

- If the key is shared by many people, a high risk of data leakage is involved.
- You cannot determine which resources can be accessed by the users.

To address these issues, you can create RAM users with their own AccessKeys for your Alibaba Cloud account. In this case, your Alibaba Cloud account is the primary account and the

RAM user accounts are the subaccounts. Subaccounts can only use their AccessKeys for the operations and resources authorized by the primary account.

- Configuration

For more information about how to create a RAM user, how to grant permissions, and group access management, see the [RAM User Guide](#).

For more information on how to configure policies required for authorization, see the **RAM and STS authorization policy configuration** section of this document.

Temporary account permissions (STS)

- Scenario

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 called as the federated users. Additionally, users can also be the applications you create that have access to your Alibaba Cloud resources.

Considering the federated users, short-term access permission management is provided to 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 logon password and AccessKey) of your Alibaba Cloud account (or RAM users), but must create a short-term access credential for a federated user. The access permission and validity of this credential is limited to you. You must not care about permission revocation. The access credentials automatically becomes invalid when it expires.

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

- Configuration

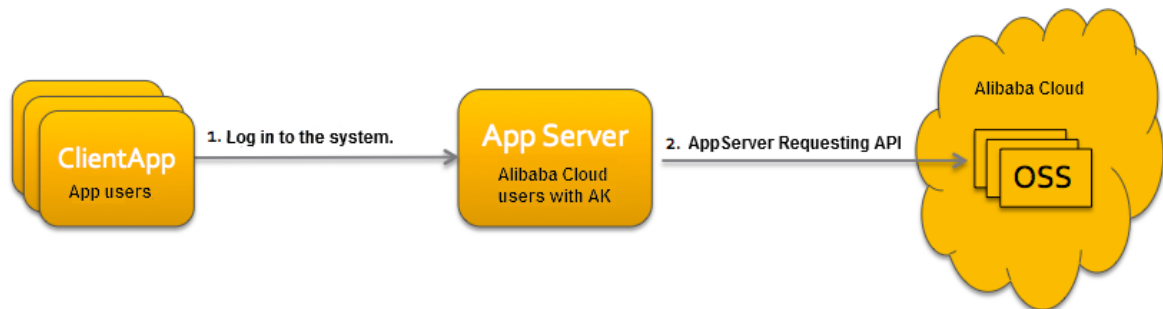
For more information on STS, see [Roles](#) in the *RAM User Guide*. You can call [AssumeRole](#) of the STS interface or directly use the STS SDK to obtain valid access credentials.

RAM and STS scenario practices

How the access identity is verified may vary with scenarios. This sections describes two methods of identity verification in typical scenarios.

A mobile app is used as an example. Suppose that you are a mobile app developer. You attempt to use the Alibaba Cloud OSS to store end user data for that app. Make sure that the data is isolated between app users.

- Mode 1: Using AppServer for data transit and data isolation

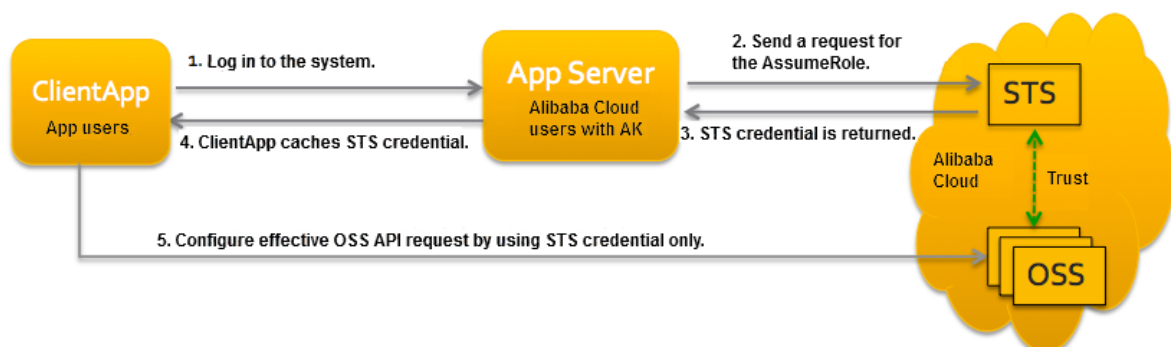


As shown in the preceding figure, you develop an AppServer. Only the AppServer can access cloud data. The ClientApp can read or write data only through the AppServer. The AppServer guarantees the data isolation of different users.

In this method, you can use the AccessKey provided by your Alibaba Cloud account or RAM account for signature verification. In case of any security issues, do not directly use the AccessKey of your Alibaba Cloud account to access OSS.

- Mode 2: Using STS for direct access to OSS

The STS solution is shown as follows:



Procedure

1. Log on 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 on. For each valid app user, the AppServer must define the minimum access permission.
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 more information, see Roles 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 AccessKey (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 must 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

Following is the example of the RAM or STS authorization policy configuration:

- 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 the authorization policy and it can be used to grant permissions for users through RAM or STS. The policy has a Statement (one policy can have multiple Statements). In this 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.

- 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 contains description of the Action, Effect, Resource, and Condition. The request system checks each statement one by one, for a match. All successfully matched statements are classified into 2 categories namely Allow and Deny based on the difference of Effect settings, and Deny is given priority. If the all the matches are Allow, the request passes the authentication. If one of the matches is Deny or no matches, this request is denied access.

- Action

Actions fall into three categories.

- Service-level action is `GetService`, which is used to list all buckets belongs to the user.
- 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`.

To authorize actions for a type of object, you can select one or more of the preceding actions. Additionally, all action names must be prefixed with "oss:", as mentioned in the preceding example. Action is a list and there can be multiple Actions. The mapping between Actions and APIs is as follows:

- Service-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

API	Action
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
RestoreObject	oss:RestoreObject

• Resource

Resource stands for a specific resource or resources on 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 render it as `acs:oss:{region}:{bucket_owner}:{bucket_name}`. Resource is a list and multiple resources are allowed. Here, the region field is currently not supported and set to `*`.

- Effect

Effect indicates the authorization result of the Statement. Two value options are available: Allow and Deny. When multiple Statement matches are observed, then the Deny option is given 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 preceding 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
<code>acs:SourceIp</code>	Specifying the IP address segment	Common IP address, wildcard (*) supported
<code>acs:UserAgent</code>	Specifying the http useragent header	String

Condition	Function	Valid value
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

More examples

For more examples of authorization policy configuration, see [Tutorial: control access to buckets and folders](#) and [OSS FAQ](#).

For the online policy graphical configuration tool, click [here](#).

You can use the Simple Policy function of ossbrowser to grant RAM users the access rights to specific buckets or directories. For more information, see [Grant permissions with a simple policy](#).

Best practices

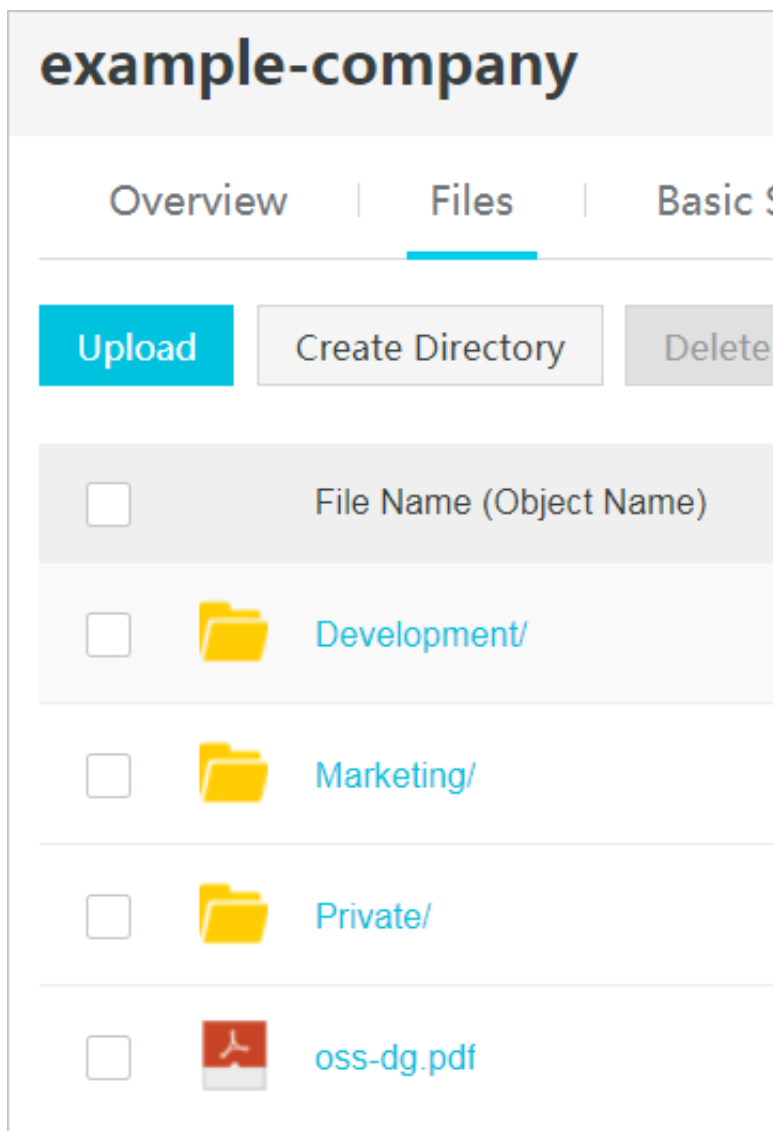
- [RAM and STS User Guide](#)

4.2 Tutorial: Control access to buckets and folders

This tutorial explains how to grant and control user access to OSS buckets and folders. In this tutorial, we create a bucket with folders, and then create Resource Access Management (RAM) users in your Alibaba Cloud primary account and grant these users incremental permissions to access your OSS bucket and its folders.

Buckets and folders

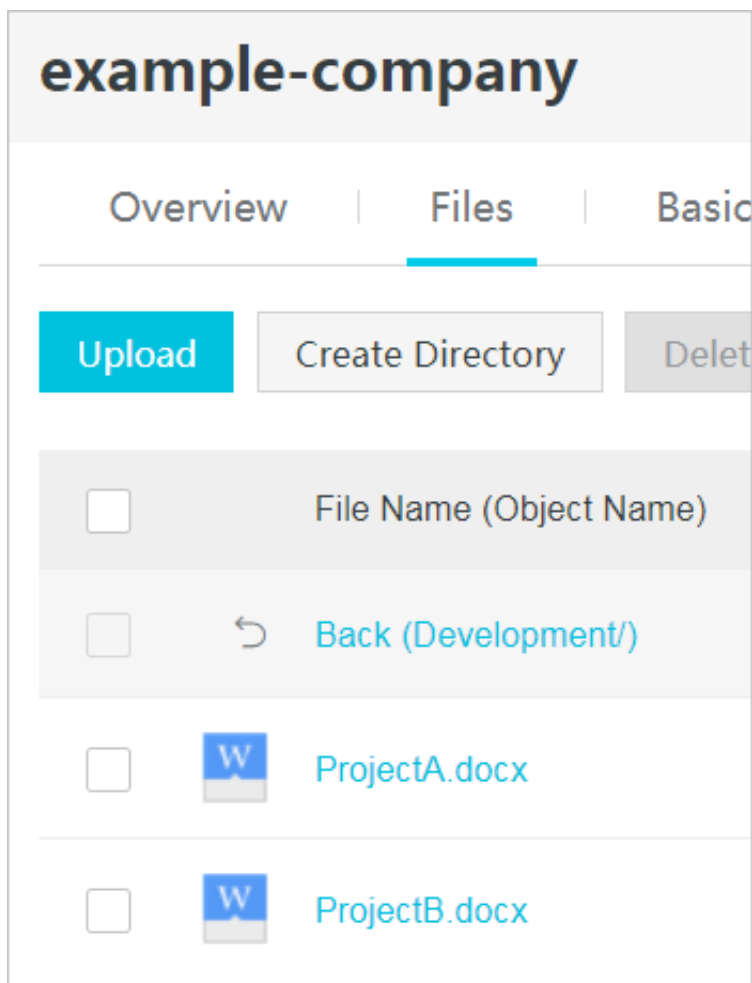
The data model structure of Alibaba Cloud OSS is flat instead of hierarchical. All objects (files) are directly related to their corresponding buckets. Therefore, OSS lacks the hierarchical structure of directories and subfolders as in a file system. However, you can emulate a folder hierarchy in the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



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

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

In the key, `Development` is the prefix and `/` is the delimiter. You can retrieve a list of all objects with a specific prefix and delimiter from a bucket. In the console, you click the **Development** folder, and the console lists the objects in the folder, as shown in the following figure.

**Note:**

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

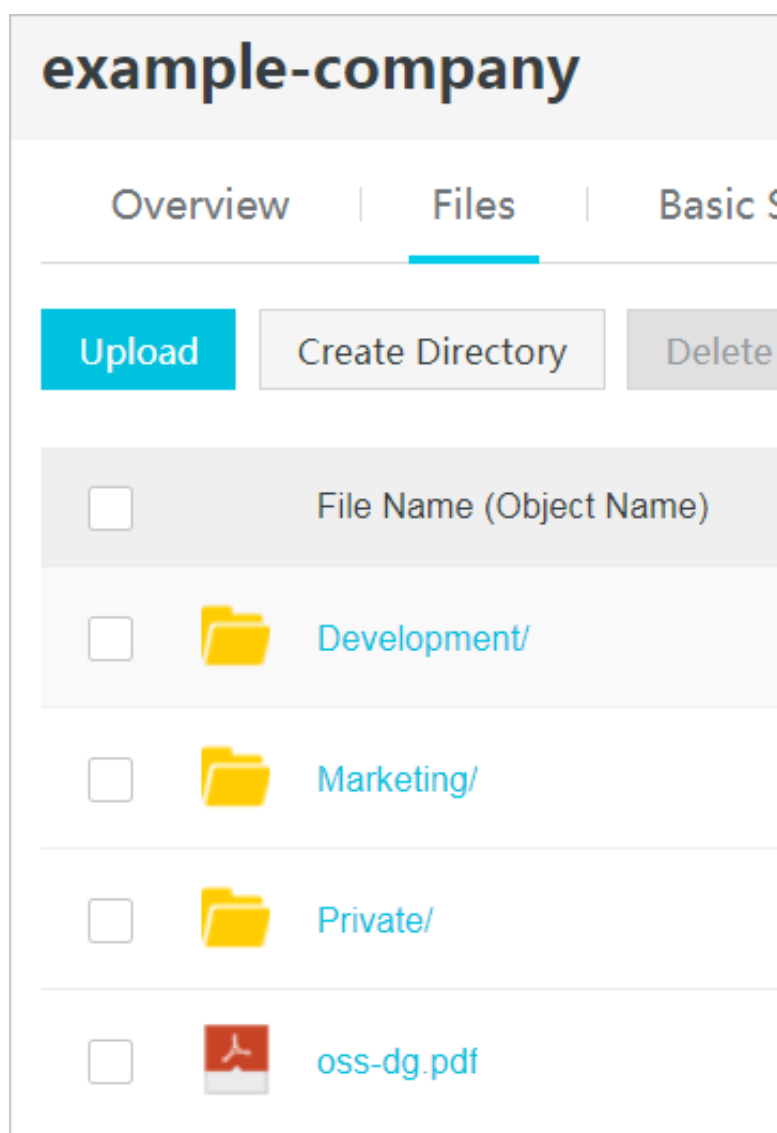
The console uses object keys to resemble a logical hierarchy. When you create a logical hierarchy of objects, you can manage access to individual folders, as described later in this tutorial.

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

- `Development/Alibaba Cloud.pdf`
- `Development/ProjectA.docx`
- `Development/ProjectB.docx`

- Marketing/data2016.xlsx
- Marketing/data2016.xlsx
- Private/2017/images.zip
- Private/2017/promote.pptx
- oss-dg.pdf

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



Request and response of OSS

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

When a user clicks a bucket name, the console sends the [GetBucket](#) request to OSS. This request includes the following parameters:

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

An example request is as follows:

```
GET /? prefix=&delimiter=/ HTTP/1.1
Host: example-company.oss-cn-hangzhou.aliyuncs.com
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqxo2oawuk53otfjbyc:DNrn7xHk3sgysx7I8U9
I9IY1vY=
```

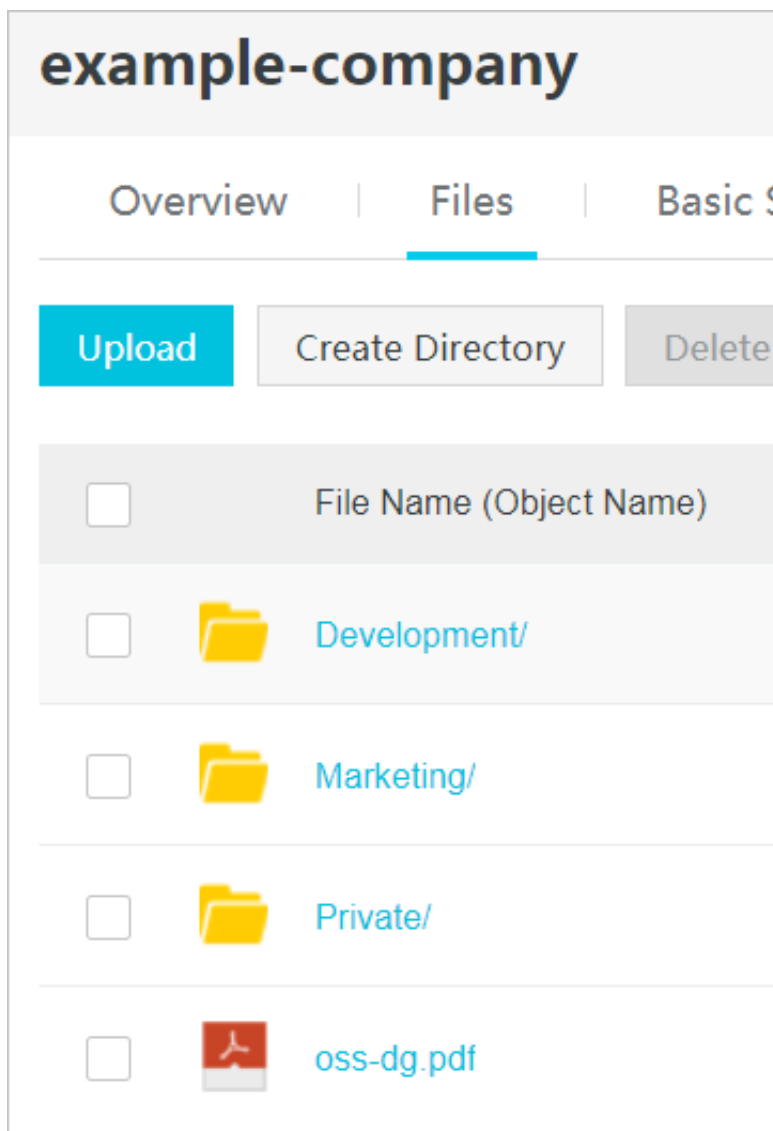
OSS returns a response that includes the `ListBucketResult` element:

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
  <Name>example-company</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>100</MaxKeys>
  <Delimiter></Delimiter>
    <IsTruncated>>false</IsTruncated>
    <Contents>
      <Key>oss-dg.pdf</Key>
      ...
    </Contents>
    <CommonPrefixes>
      <Prefix>Development</Prefix>
    </CommonPrefixes>
    <CommonPrefixes>
      <Prefix>Marketing</Prefix>
    </CommonPrefixes>
    <CommonPrefixes>
      <Prefix>Private</Prefix>
    </CommonPrefixes>
  </ListBucketResult>
```

The key `oss-dg.pdf` does not contain the `/` delimiter, so OSS returns the key in the `<Contents/>` element. All other keys in the bucket `example-company` contain the `/` delimiter, so OSS groups

these keys and returns a `CommonPrefixes/` element for each of the prefix values `Development/`, `Marketing/`, and `Private/`. The `CommonPrefixes/` element includes a substring of the key name, which starts from the beginning of the key name and ends with (but does not include) the first occurrence of the specified `/` delimiter.

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



Now, if a user clicks the **Development** folder, the console sends the [GetBucket](#) request to OSS.

This request includes the following parameters:

- `prefix` with `Development/` as its value.
- `delimiter` with `/` as its value.

An example request is as follows:

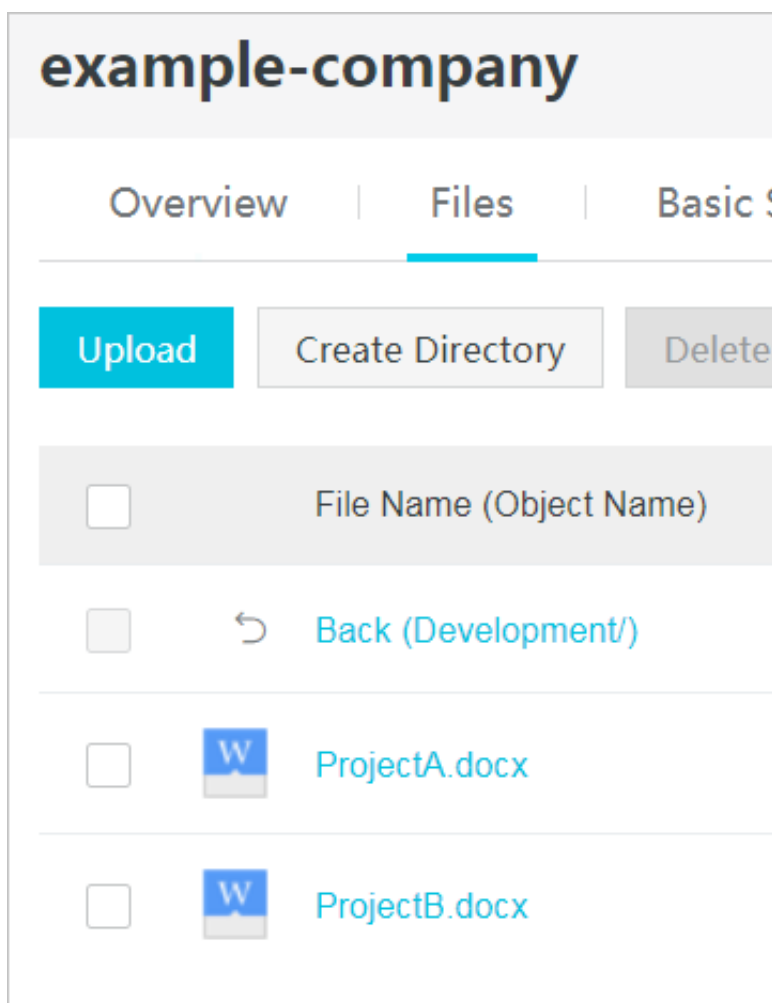
```
GET /? prefix=Development/&delimiter=/ HTTP/1.1
Host: oss-example.oss-cn-hangzhou.aliyuncs.com
```

```
Date: Fri, 24 Feb 2012 08:43:27 GMT
Authorization: OSS qn6qrrqx02oawuk53otfjbyc:DNrn7xHk3sgysx7I8U9
I9IY1vY=
```

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

```
HTTP/1.1 200 OK
x-oss-request-id: 534B371674E88A4D8906008B
Date: Fri, 24 Feb 2012 08:43:27 GMT
Content-Type: application/xml
Content-Length: 712
Connection: keep-alive
Server: AliyunOSS
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.oss-cn-hangzhou.aliyuncs.com">
  <Name>example-company</Name>
  <Prefix>Development/</Prefix>
  <Marker></Marker>
  <MaxKeys>100</MaxKeys>
  <Delimiter></Delimiter>
    <IsTruncated>>false</IsTruncated>
    <Contents>
      <Key>ProjectA.docx</Key>
      ...
    </Contents>
    <Contents>
      <Key>ProjectB.docx</Key>
      ...
    </Contents>
</ListBucketResult>
```

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



Tutorial example

The tutorial example is as follows:

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

**Note:**

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

Prepare for the tutorial

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

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

**Note:**

For more information about RAM, see [Log on with a RAM user account](#).

Provide a logon link for RAM users

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

Step 1. Create a bucket

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

1. Log on to the [OSS console](#).
2. Create a bucket named **example-company**.

For detailed procedures, see [Create a bucket](#) in the *OSS Console User Guide*.

3. Upload one file to the bucket.

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

For detailed procedures, see [Upload files](#) in the *OSS Console User Guide*.

4. Create three folders named Development, Marketing, and Private.

For detailed procedures, see [Create a folder](#) in the *OSS Console User Guide*.

5. Upload one or two files to each folder.

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

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

Step 2. Create RAM users and a group

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



Note:

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

For detailed procedures on creating a RAM user, see [Create a RAM user](#) in the RAM Quick Start. Remember to create a logon password for each RAM user.

For detailed procedures on creating a group, see the Create a group section of [Groups](#) in the RAM User Guide.

Step 3: Verify that RAM users have no permissions

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

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

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

Step 4: Grant group-level permissions

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

- List all buckets owned by the primary account.

To do this, Anne and Leo must have permission for the `oss:ListBuckets` action.

- List root-level items, folders, and objects, in the example-company bucket.

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

Step 4.1: Grant permissions to list all buckets

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

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

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

```
{
  "Effect": "Allow",
  "Action": [
    "oss:ListBuckets"
  ],
  "Resource": [
    "acs:oss:*:*:*"
  ]
}
```

**Note:**

A policy is a JSON document. In the policy, the Statement attribute is an array of objects, and each object describes a permission using a collection of name value pairs. The preceding policy describes one specific permission. The Effect attribute value determines whether a specific permission is allowed or denied. The Action attribute specifies the type of access. In the policy, the `oss:ListBuckets` is a predefined OSS action, which returns a list of all buckets owned by the authenticated sender.

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

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

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

4. Test the permission.

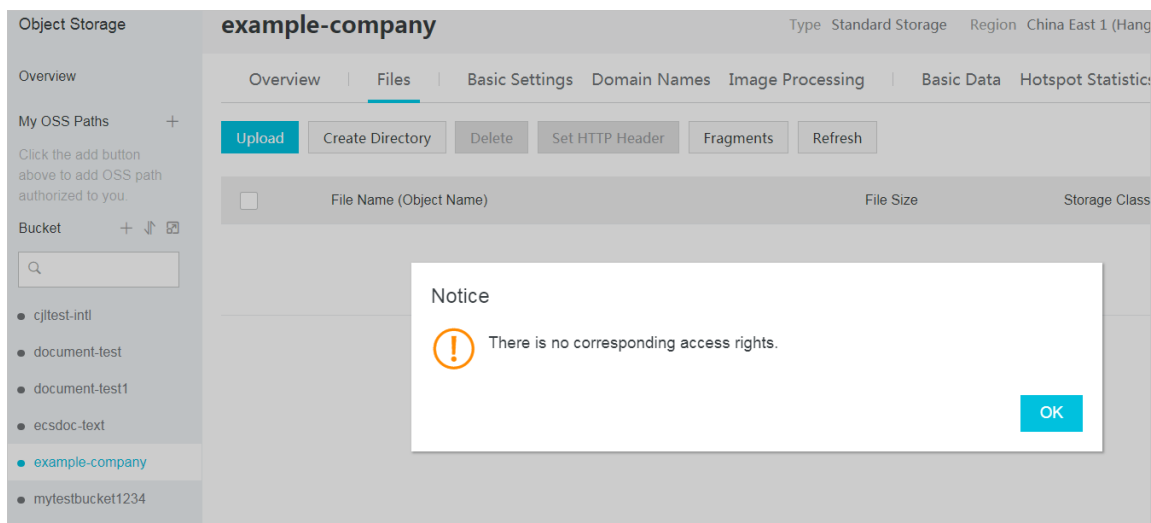
a. Open the RAM user logon page, and log on to the RAM console with Anne's or Leo's credentials.

b. Open the OSS console.

The console lists all of the buckets.

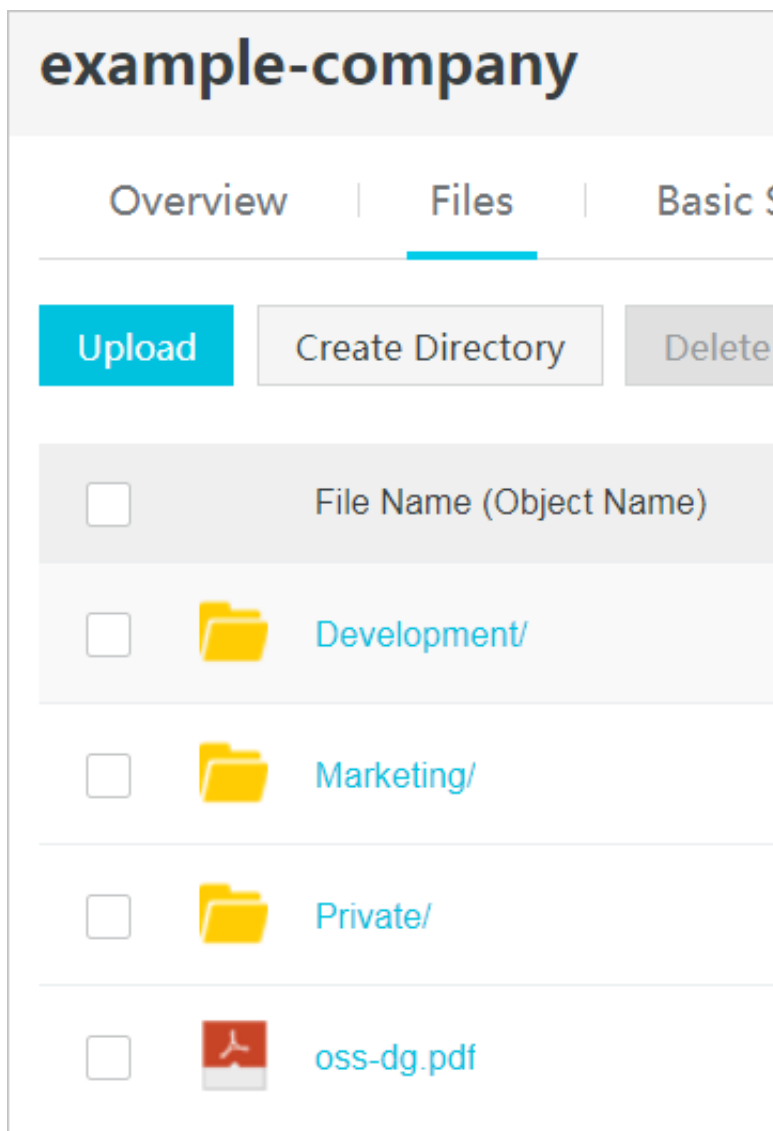
c. Click the example-company bucket, and then click the **Files** tab.

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



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

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



1. Log on to the [RAM console](#) with your primary account credentials.
2. Replace the existing policy **AllowGroupToSeeBucketListInConsole** that is attached to the Staff group with the following policy. The following policy also allows the `oss:ListObjects` action. Remember to replace `example-company` in the policy Resource with the name of your bucket.

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

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

```

    "Action": [
      "oss:ListBuckets",
      "oss:GetBucketAcl"
    ],
    "Resource": [
      "acs:oss:*:*:*"
    ],
    "Condition": {}
  },
  {
    "Effect": "Allow",
    "Action": [
      "oss:ListObjects"
    ],
    "Resource": [
      "acs:oss:*:*:example-company"
    ],
    "Condition": {
      "StringLike": {
        "oss:Prefix": [
          ""
        ],
        "oss:Delimiter": [
          "/"
        ]
      }
    }
  }
]
}

```

**Note:**

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

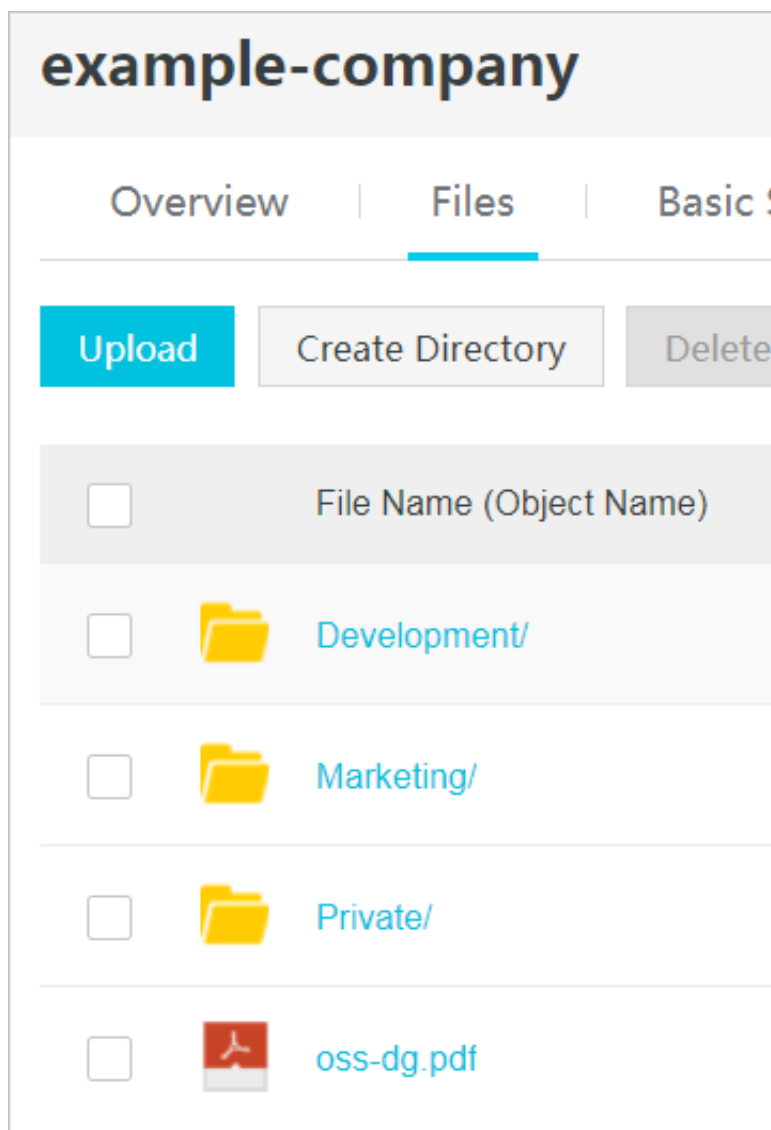
3. Test the updated permissions.

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

The console lists all of the buckets.

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

The console lists all the root-level items.



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

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

Summary of the group policy

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

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

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

- Provide Anne the ability to get and put objects in the Development folder.
- Provide Bob the ability to get and put objects in the Finance folder.

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

Step 5: Grant RAM user Anne specific permissions

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

Step 5.1: Grant RAM user Anne permission to list the Development folder content

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

1. Log on to the [RAM console](#) with your primary account credentials.
2. Create a policy **AllowListBucketsIfSpecificPrefixesIncluded** that grants the RAM user Anne permission to list the Development folder content.
 - a. From the left-side navigation pane, click **Policies**, and then click **Create Authorization Policy**.
 - b. Click **Blank Template**.
 - c. In the **Authorization Policy Name** field, enter **AllowListBucketsIfSpecificPrefixesIncluded**.
 - d. In the **Policy Content** field, copy and paste the following policy.

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

```
]
}
```

3. Attach the policy to the RAM user Anne.

For detailed procedures on attaching a policy, see [Attach policies to a RAM user](#) in the RAM Quick Start.

4. Test Anne's permissions.

- a. Open the RAM user logon page, and log on to the RAM console with Anne's credentials.
- b. Open the OSS console. The console lists all of the buckets.
- c. Click the example-company bucket, and then click the **Files** tab. The console lists all the root-level items.
- d. Click the Development/ folder. The console lists the objects in the folder.

Step 5.2 Grant RAM User Anne permissions to get and put objects in the Development folder

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

1. Log on to the [RAM console](#) with your primary account credentials.
2. Replace the policy **AllowListBucketsIfSpecificPrefixIsIncluded** you created in the previous step with the following policy.

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

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

```
{
  "Effect": "Allow",
  "Action": [
    "oss:GetObject",
    "oss:PutObject",
    "oss:GetObjectAcl"
  ],
  "Resource": [
    "acs:oss:*:*:example-company/Development/*"
  ],
  "Condition": {}
}
```

**Note:**

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

3. Test the updated policy.

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

The console lists all of the buckets.

- c. In the OSS console, verify that Anne can now add an object and download an object in the Development folder.

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

RAM user Anne can now list the root-level content in the example-company bucket, and get and put objects in the Development folder. If you want to strictly restrict the access permissions, you can explicitly deny Anne's access to any other folders in the bucket. If other policies grant Anne's access to any other folders in the bucket, this explicit policy overrides those permissions.

You can add the following statement to the RAM user Anne's policy **AllowListBucketsIfSpecificPrefixIsIncluded**. The following statement requires all requests that Anne sends to OSS to include the prefix parameter, and the parameter value can be either `Development/*` or an empty string.

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

```

        "Condition": {
            "StringNotLike": {
                "oss:Prefix": [
                    "Development/*",
                    ""
                ]
            }
        }
    }
}

```

Follow the preceding step to update the policy **AllowListBucketsIfSpecificPrefixIsIncluded** that you created for RAM user Anne. Copy and paste the following policy to replace the existing one.

```

{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringLike": {
          "oss:Prefix": [
            "Development/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:GetObject",
        "oss:PutObject",
        "oss:GetObjectAcl"
      ],
      "Resource": [
        "acs:oss:*:*:example-company/Development/*"
      ],
      "Condition": {}
    },
    {
      "Effect": "Deny",
      "Action": [
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringNotLike": {
          "oss:Prefix": [
            "Development/*",
            ""
          ]
        }
      }
    }
  ]
}

```

```
    ]
  }
```

Step 6: Grant RAM user Leo specific permissions

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

Step 7: Secure the Private folder

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

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

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

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

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

```

    }
  }
]

```

- Replace the Staff group policy **AllowGroupToSeeBucketListInConsole** with an updated policy that includes the preceding deny statements. After the updated policy is applied, none of the users in the group can access the Private folder in your bucket.
1. Log on to the [RAM console](#) with your primary account credentials.
 2. Replace the existing policy **AllowGroupToSeeBucketListInConsole** that is attached to the Staff group with the following policy. Remember to replace example-company in the policy Resource with the name of your bucket.

```

{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListBuckets",
        "oss:GetBucketAcl"
      ],
      "Resource": [
        "acs:oss:*:*:*"
      ],
      "Condition": {}
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListObjects"
      ],
      "Resource": [
        "acs:oss:*:*:example-company"
      ],
      "Condition": {
        "StringLike": {
          "oss:Prefix": [
            ""
          ],
          "oss:Delimiter": [
            "/"
          ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "oss:*"
      ],
      "Resource": [
        "acs:oss:*:*:example-company/Private/*"
      ],
      "Condition": {}
    }
  ]
}

```

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

Cleanup

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

For detailed procedures, see [Delete a RAM user](#) section of **Users** in the *RAM User Guide*.

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

5 Access OSS

5.1 OSS-based app development

Development sequence diagram

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

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

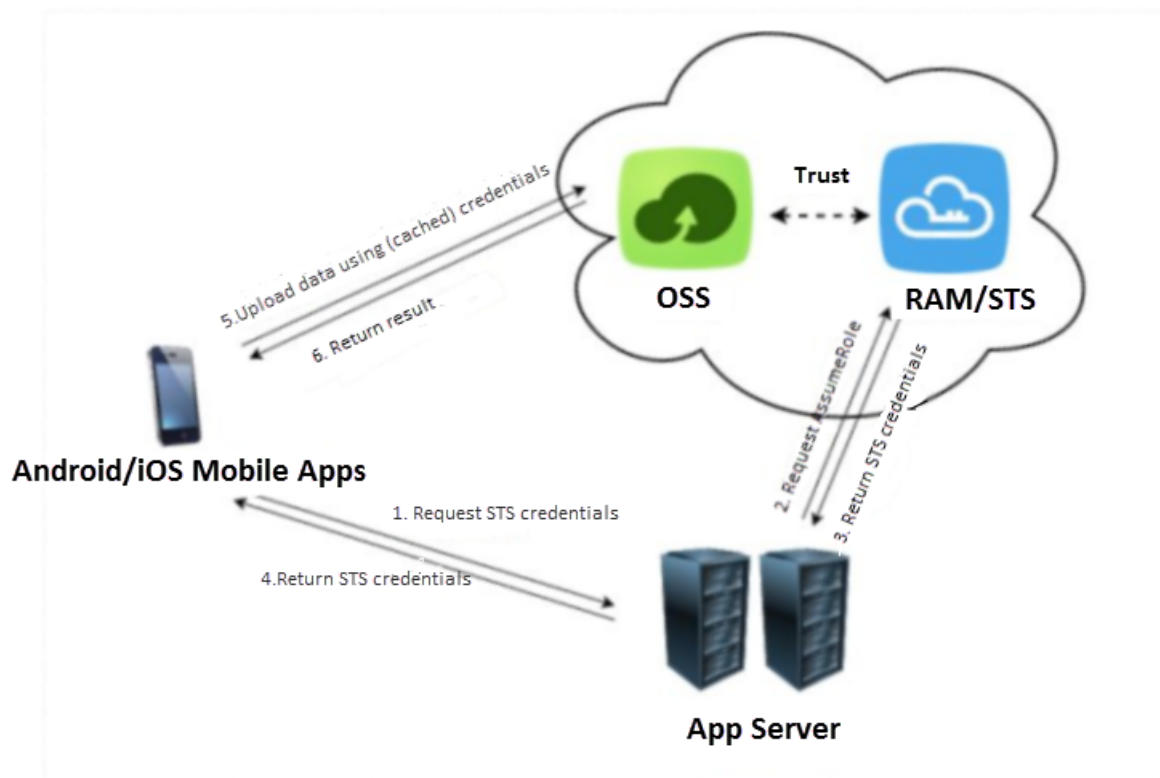
Best practices

- [Set up direct data transfer for mobile apps](#)
- [Set up data callback for mobile apps](#)
- [Permission control](#)

Service development process

- Data upload with temporary credential authorization

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



The description of the process is as follows:

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

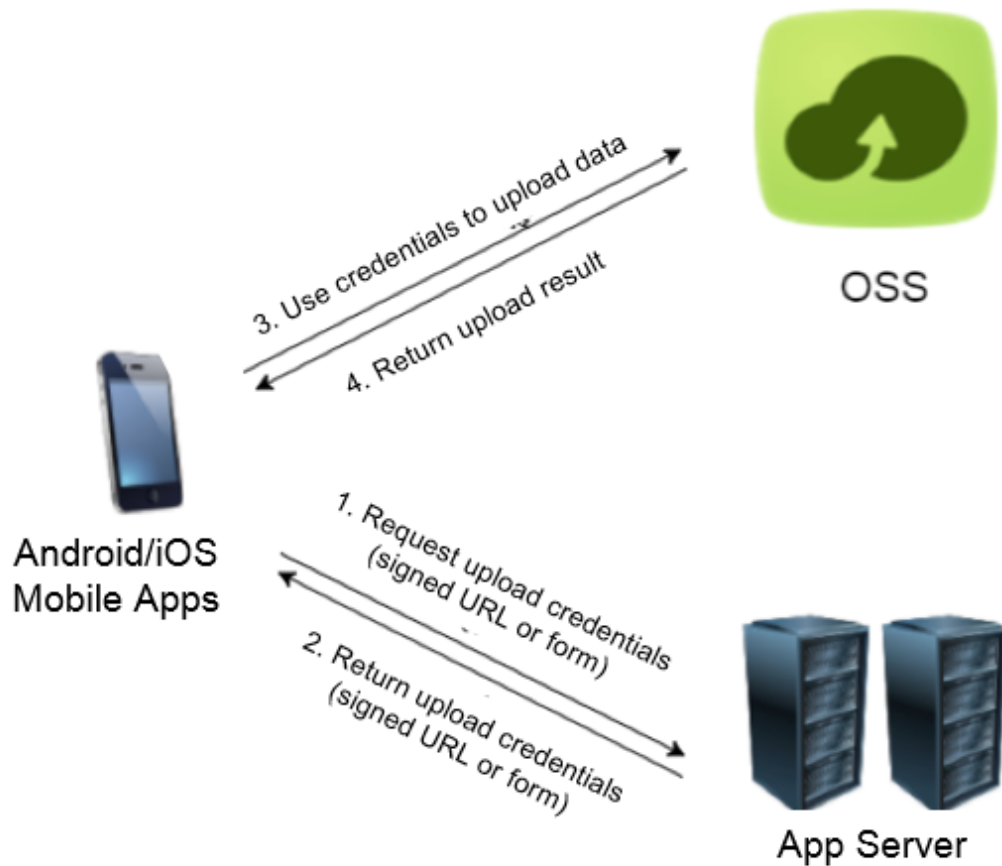
Note that:

- The client does not have to request authorization from the application server for each upload attempt. Each time the authorization is obtained, the client caches temporary credentials returned by STS until it expires.
- STS provides fine-grained access control for upload, which restricts client access permissions at the object level. This completely isolates the objects uploaded to OSS by different clients, and thus greatly enhances application security.

For more information, see [Authorized third-party upload](#).

- Data upload with signed URL or form authorization

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



The description of the process is as follows:

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

For more information, see [Authorized third-party upload](#).

- Data download with temporary credential authorization

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

1. The client sends the application server the request of downloading data from OSS.

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

Note that:

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

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

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



Note:

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

Best practices

- [What is RAM and STS](#)

Reference

- Android SDK: [Upload objects](#)
- iOS SDK: [Upload objects](#)

5.2 Quick start

1. Log on to the [OSS console](#).
2. Create a bucket.
3. Upload and download files.

For more information, see [Get started with Alibaba Cloud OSS](#).

Get familiar with OSS upload and download

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

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

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

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

OSS also provides different download methods. For more information, see [Simple download](#) and [Multipart download](#).

General process of using OSS SDKs

1. Obtain the AccessKeyId and AccessKeySecret from the console.
2. Download the OSS SDKs in your preferred programming language from GitHub.
3. Perform uploads, downloads, and other operations.

For more information about how to use the OSS SDKs for different programming languages, see [OSS SDK Reference](#).

6 Manage buckets

6.1 Set bucket read and write permissions

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

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

For more information about ACL, see [Access control](#).

Reference

Set bucket ACL:

- Console: [Set ACL](#)
- SDK: Java SDK-[Set bucket ACL](#)
- API: [Put BucketACL](#)

Get bucket ACL:

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

6.2 Obtain bucket region information

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

Reference

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

6.3 View the bucket list

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

Reference

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

Additional links

- [Create a bucket](#)

6.4 Set a WORM strategy

A Write Once Read Many (WORM) strategy is used to specify the protection period for files in the bucket. No one can modify or delete the files during the protection period.



Note:

Currently, OSS only supports the WORM strategy at bucket level.

Strategy description

Currently, OSS allows only one time-based WORM strategy, and the protection period (lifecycle) is 1 day to 70 years.

Assume that you created a bucket named examplebucket on June 1, 2013, and then uploaded file1.txt, file2.txt, and file3.txt to the bucket at different times. After that, a WORM strategy was

created on July 1, 2014 with a protection period of 5 years on the bucket. The specific upload time for the preceding three files and their corresponding WORM strategy expiration dates are as follows:

File name	Upload time	Expiration date of WORM strategy
file1.txt	June 1, 2013	May 31, 2018
file2.txt	July 1, 2014	June 30, 2019
file3.txt	September 30, 2018	September 29, 2023

You can also specify the effective rules, modification rules and deletion rules for a time-based WORM strategy.

- Effective rules

When a time-based WORM strategy is created, the policy is in the InProgress state by default, and the state is valid for 24 hours. Within 24 hours of the validity period, the bucket resources under this policy are protected.

- Within 24 hours after the creation of the WORM strategy: The bucket owner and authorized users can modify or delete the strategy if it is not locked. If the WORM strategy is locked, then the policy cannot be modified and deleted, but you can extend the protection period.
- 24 hours later after the creation of the WORM strategy: If the WORM strategy is not locked, then the policy automatically expires.

- Modification rules

For the InProgress and Locked state of the time-based WORM strategy, two kinds of modification rules are provided:

- When the WORM strategy is in InProgress state, it can be deleted only.
- When the WORM strategy is in Locked state, the protection period (Lifecycle) can be extended, but the strategy cannot be deleted.

- Deletion rules

A time-based WORM strategy is the metadata property of a bucket. When you delete a bucket, its corresponding WORM strategy and access policy are also deleted.

Reference

Console : [Set a WORM strategy](#)

6.5 Bind a custom domain name

You can bind a custom domain name to your OSS bucket in the OSS console and then add a CNAME record in the Domain Name System (DNS). After the CNAME resolution, OSS automatically processes the access requests to the custom domain name.

Use case

Suppose that John has a website with the domain name `abc.com` in Hangzhou. The website contains a page with the link `http://img.abc.com/logo.png`. John wants the requests for `http://img.abc.com/logo.png` to be serviced from the OSS content. CNAME is particularly well suited for this scenario. The process is as follows:

1. John creates a bucket named `abc-img` in China East 1 (Hangzhou), and uploads the image `logo.png`.
2. John binds the domain name `img.abc.com` to the bucket `abc-img` in the OSS console.
3. OSS maps the domain name `img.abc.com` to the bucket `abc-img`.
4. John adds a CNAME record in the Domain Name System (DNS) to map the domain name `img.abc.com` to the OSS domain name of the bucket `abc-img`, that is, `abc-img.oss-cn-hangzhou.aliyuncs.com`.
5. When the request for `http://img.abc.com/logo.png` is sent, OSS resolves the mapping between `img.abc.com` and `abc-img`, and directs the request to `http://abc-img.oss-cn-hangzhou.aliyuncs.com/logo.png`.

The comparison between the process before CNAME resolution and that after CNAME resolution is as follows:

Before CNAME resolution	After CNAME resolution
1. A request to access <code>http://img.abc.com/logo.png</code> is received. 2. DNS resolves the request to the user's server IP address. 3. Access to <code>logo.png</code> on the user's server is achieved.	1. A request to access <code>http://img.abc.com/logo.png</code> is received. 2. DNS resolves the request to <code>abc-img.oss-cn-hangzhou.aliyuncs.com</code>. 3. Access to <code>logo.png</code> in the OSS bucket <code>abc-img</code> is achieved.

Reference

For more information about how to bind a custom domain name in the OSS console, see [Manage a domain name](#) in the *OSS Console User Guide*.

6.6 Delete a bucket

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

Reference

- API: [Delete Bucket](#)
- SDK: Java SDK - [Delete a bucket](#)
- Console: [Delete a bucket](#)

7 Upload files

7.1 Simple upload

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

Set object metadata when uploading an object

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

Upload restrictions

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

Upload large objects

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

Security and authorization

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

Further operations

After uploading objects to OSS, you may want to:

- Initiate a callback request to a specified application server. For more information, see [Upload callback](#).
- Process the uploaded images. For more information, see [Image processing](#).

Reference

- API: [PutObject](#)
- Java SDK: [Simple upload](#)
- Console: [Upload objects](#)

Best practices

- [RAM and STS User Guide](#)
- [Web client direct data transfer and upload callback](#)

7.2 Form upload

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

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

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

Upload restrictions

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

Advantages of form upload

If the form upload is not used, files are uploaded to the web server first, and then the web server forwards the files to OSS. In case of huge uploads, the web server becomes the bottleneck and needs to be scaled up. If the form upload is used, files are uploaded directly from the client to OSS

without the forwarding of the web server. OSS handles all upload requests and guarantees the service quality.

Security and authorization

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

To grant upload permissions to a third party, you can use the PostObject interface. For more information, see [PostObject](#).

Procedures for form upload

1. Construct a Post policy.

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

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

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

2. Encode the policy string using Base64.

3. Use the OSS AccessKeySecret to sign the Base64-encoded policy.

4. Construct an HTML page for uploads.

5. Open the HTML page and select and upload files.

A complete Python code example is as follows:

```
#coding=utf8
import md5
import hashlib
import base64
import hmac
from optparse import 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
    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 example as `post_object.py` and run it by using `python post_object.py`.

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:**

In the constructed form,

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

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

Usage

- API: [PostObject](#)
- Java SDK: [Form upload](#)

Best practices

- [Web client direct upload](#)
- [Cross-origin Resource Sharing \(CORS\)](#)

7.3 Multipart upload

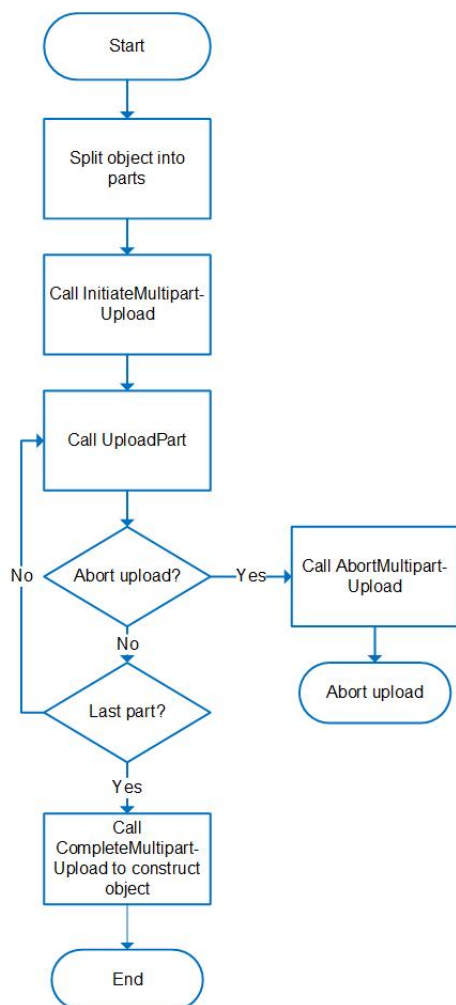
For the object larger than 5 GB, you can use the multipart upload to split it into multiple data blocks (called parts in OSS) and upload them separately. When you have uploaded all parts, OSS constructs the object from the uploaded parts.

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

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

Workflow

The workflow of the multipart upload is shown as follows:



The description of the workflow is as follows:

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

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

- All the parts, except the last one, must not be smaller than 100 KB. Otherwise, the call to the [CompleteMultipartUpload](#) interface fails.

- After the object is split into parts, the parts are ordered by the partNumbers specified during the upload. The upload speed does not correlate to the number of parts uploaded concurrently, because both the network conditions and the device load must be considered.
- By default, when the upload is complete but the call to the [CompleteMultipartUpload](#) interface fails, the uploaded parts will not be deleted automatically. You can call the [AbortMultipartUpload](#) interface to terminate the upload and save the storage space. To automatically delete the uploaded parts, see [Manage object lifecycle](#).

Resumable upload

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

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

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

Restrictions

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

Security and authorization

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

OSS also provides account-level authorization for third-party uploads. For more information, see [Authorized third-party uploads](#).

Further operations

After uploading objects to OSS, you may want to:

- Initiate a callback request to a specified application server. For more information, see [Upload callback](#).
- Process the uploaded data. For more information, see [Cloud data processing](#).

Usage

- API:
 - [MultipartUpload](#)
 - [InitiateMultipartUpload](#)
 - [UploadPart](#)
 - [UploadPartCopy](#)
 - [CompleteMultipartUpload](#)
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [ListParts](#)
- SDK: Java SDK- MultipartUpload in [Upload objects](#)

Best practices

- [RAM and STS best practices](#)
- [Web client direct upload](#)

7.4 Append object

Applicable scenarios

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

This feature makes many application scenarios inconvenient, such as video monitoring and live video broadcast, since video data is constantly produced in real time. Using other upload methods, users must slice the video stream into small pieces and then upload them as new objects. In actual use, these methods have obvious defects:

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

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

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

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

Upload restrictions

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

Upload security and authorization

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

Post-upload Operations

To process uploaded images, users can use [Image Processing](#). For audio/video file format conversion, users can use [Media Processing](#).

Reference for using the function

- API: [Append Object](#)
- Java SDK: [Append object](#)

**Note:**

Append object method does not support upload callback.

Best practices

- [RAM and STS User Guide](#)

7.5 Authorized third-party upload

Applicable scenarios

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

To resolve this issue, OSS provides an authorized third-party upload function. This means each client can directly upload files to the OSS, bypassing the need for a server. This reduces the cost for application servers and takes full advantage of the OSS's ability to process massive data volumes.

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

URL signature

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

Temporary access credentials

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

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

Best practices

- [RAM and STS User Guide](#)
- [Web client direct data transfer and upload callback](#)

7.6 Upload callback

Use cases

When an object upload is completed, the OSS can provide a callback to the application server. To implement the callback, you only need to attach the relevant Callback parameter to the request sent to the OSS. APIs that currently support callbacks include PutObject, PostObject, and CompleteMultipartUpload.

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

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

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

**Note:**

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

Reference

- API: [Callback](#)
- SDK: iOS [Callback notification after upload](#)
- [Upload callback](#)

Best practices

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

8 Download files

8.1 Simple download

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

A user may access a certain object in two conditions:

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

For more information about object and bucket access permission control, see [Access control](#).

To authorize a third-party user to download an object from a private bucket, see [Authorized third-party download](#).

To use multipart download, see [Multipart download](#).

Reference

- API: [GetObject](#)
- SDK: Java SDK-[Object](#)
- Console: [Get object URL](#)

Best practices

- [RAM and STS User Guide](#)

8.2 Multipart download

OSS provides a "start object download from specified point" function. This allows users to split large objects into multiple downloads, which improves speed and reliability of downloads. If a download is paused or interrupted, it resumes at the point of interruption once restarted.

Similar to a simple upload, the user must have read permission for the object. Multipart downloads are supported when the Range parameter is set. If the Range parameter is specified in the request header, the returned message contains the length of the entire file and the range returned

in this response. For example, Content-Range: bytes 0-9/44 indicates that the length of the entire file is 44, and the range in the response body is 0–9. If the range requirement is not met, the system transfers the entire file and does not include Content-Range in the result. The return code is 206.

Reference

- API: [GetObject](#)

8.3 Authorized third-party download

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

URL signature

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

- Implementation method

Example URL that includes a signature:

```
http://<bucket>.<region>.aliyuncs.com/<object>?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 expected URL expiration time.
- Signature, which is the developer's signature string. For more information, see [Add a signature to a URL](#).



Note:

This link must undergo URL encoding.

- Reference
 - API: [Get Object](#)
 - SDK: Java SDK-[Using URL Signature to Authorize Access](#)

— Console: [Get object URL](#)

**Note:**

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

Temporary access credentials

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

Implementation method

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

Reference

- API: [Temporary Access Credentials](#)
- SDK: Use STS temporary authorization in Java SDK-[Object](#)
- Console: [Get object URL](#)

Best practices

- [RAM and STS User Guide](#)

9 Manage files

9.1 Object Meta

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

- HTTP standard attributes

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

- User Meta

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

Set object Meta when uploading objects

You can set object Meta when uploading objects.

Reference:

- API: [PutObject](#)
- SDK: Java SDK-**Set the HTTP header and User-defined metadata** in [Upload objects](#)

You can set object Meta when using multipart uploads.

Reference:

- API: [InitiateMultipartUpload](#)
- SDK: Java SDK-[Initialize multipart upload](#)

Modify object Meta after uploading objects

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

Reference

- API: [CopyObject](#)
- SDK: Java SDK-[Use CopyObjectRequest to copy objects](#)

Retrieve object Meta

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

Reference:

- API: [HeadObject](#)
- SDK: Java SDK-[Get object metadata](#)

9.2 View the object list

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

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

Name	Function
	specified, the default value is 100. The MaxKeys value cannot exceed 1,000.
Prefix	Indicates that only the objects whose keys contain the specified prefix are returned. Note that keys returned from queries using a prefix still contains the prefix.

Folder simulation

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

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

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

```
For example:
In this example, the OSS bucket oss-sample, contains the following
objects:
File D
Directory A/File C
Directory A/File D
Directory A/Directory B/File B
Directory A/Directory B/Directory C/File A
Directory A/Directory C/File A
Directory A/Directory D/File B
Directory B/File A
1. List first-level 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: [GetBucket](#)
- SDK: Java SDK-[List objects in a bucket](#)

9.3 Copy an object

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

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

Before copying an object, note the following precautions:

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

Reference:

- API: [CopyObject](#)

- SDK: Java SDK-[CopyObject](#)

Copy large objects

The OSS supports the function of copying large files similar to [Multipart upload](#).

The only difference is that the process [UploadPart](#) is replaced by the process [UploadPartCopy](#).

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

Reference:

- API: [UploadPartCopy](#)
- SDK: Java SDK-[Copy a large object](#)

9.4 Delete an object

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

- Single deletion, in which only a specified object is deleted.
- Batch deletion, in which up to 1,000 objects can be deleted at one time.
- Auto deletion, in which large numbers of objects can be deleted according to certain rules.

For example, to regularly delete objects that are created a specified number of days ago, or to regularly empty the entire bucket, we recommend that you use [Lifecycle management](#). Once the rules are specified, OSS uses these rules to recycle expired objects. This reduces the number of user attempts at deletion requests, and helps streamline the deletion process.

Reference

- API: [Delete Object](#) and [Delete Multiple Objects](#)
- SDK: Java SDK [Delete Files](#)
- Console: [Delete Files](#)

9.5 Manage object lifecycle

OSS provides Object Lifecycle Management to manage objects for you. 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 only applies 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 includes 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. The following 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 preceding 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 are deleted. Objects modified after this time are not deleted.
 - **<Days>** indicates that files that were last modified more than 10 days ago are deleted.

In the first rule, the OSS deletes 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 are deleted, but the rule does 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 returns 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 deletes matched objects after midnight on January 1, 2014.
- If, in a rule to delete objects, the number of days is specified, OSS sums up the last update time (Last-Modified) and the specified number of days, and then round the sum to the midnight UTC timestamp. For example, if the last update time of an object is 01:00 a.m. on April 12, 2014 and the number of days specified in the matched rule is 3, the expiry time is 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 is the time of the last Put operation. If an object was copied to itself, the last update time is the time at when the object was last copied.

Reference

- API: [PutBucketLifecycle](#)
- Console: [Set lifecycle](#)

9.6 Cross-region replication

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

Use cases

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

- **Compliance requirements:** Although, by default, OSS stores multiple copies of each object on a physical disk, compliance requirements may dictate that you store a copy of the data at a further distance. Cross-region replication allows you to replicate data between distant OSS data centers to satisfy these compliance requirements.
- **Minimize latency:** Your customers are in two geographic locations. To minimize latency in accessing objects, you can maintain object copies in OSS data centers that are geographically closer to your users.
- **Data backup and disaster recovery:** You have high requirements on data security and availability, and you want to explicitly maintain copies of all written data in a second data center. In case one OSS data center is damaged by a catastrophic event like earthquake and tsunami, you can use backup data from the other one.
- **Data replication:** For business reasons, you may need to migrate data from one OSS data center OSS to another.
- **Operational reasons:** You have computing clusters in two different data centers that analyze the same set of objects. You may choose to maintain object copies in these regions.

Instructions

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

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

Restrictions

- For two buckets that are in synchronization, because you can operate on both buckets at the same time, copying an object from the source bucket may overwrite the object with the same name in the target bucket. Be cautious when using this feature.
- Because Bucket Replication uses an asynchronous copying method, it usually takes several minutes or hours to copy data to the target bucket, depending on the size of data.
- Cross-region synchronization only works when no synchronization to a third bucket is enabled for the two buckets to be synchronized. For example, if synchronization to Bucket B is enabled for Bucket A, you can no longer enable synchronization to Bucket C for Bucket A, unless you delete the former configuration first. Similarly, if synchronization to Bucket B is enabled for Bucket A, it is not allowed to enable synchronization from Bucket C to Bucket B.
- Synchronization is supported only between two buckets in different regions.
- Currently, the cross-region replication feature is only supported between different regions in Mainland China and between Eastern and Western United States.

Reference

- Console: [Cross-region replication](#)

9.7 Manage back-to-origin settings

Back-to-origin allows for multiple origin retrieval 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 then specifies an origin retrieval method. A maximum of five rules can be configured. Requests are compared to the rules in a set sequence, until matched to a valid rule. The specified method can be either mirroring or redirection.

Mirroring



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

Example scenario

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

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

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

If the configured origin site is an IP address, after the domain name is migrated to the OSS, data can still be mirrored to the origin site.

- However, if it is a domain name, no mirroring can be produced because the domain name is resolved to the OSS or CDN. In this situation, the user can apply for another domain name to mirror the origin site.

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

Usage rules

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

```
Content-Type
Content-Encoding
Content-Disposition
Cache-Control
Expires
Content-Language
```

Access-Control-Allow-Origin

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

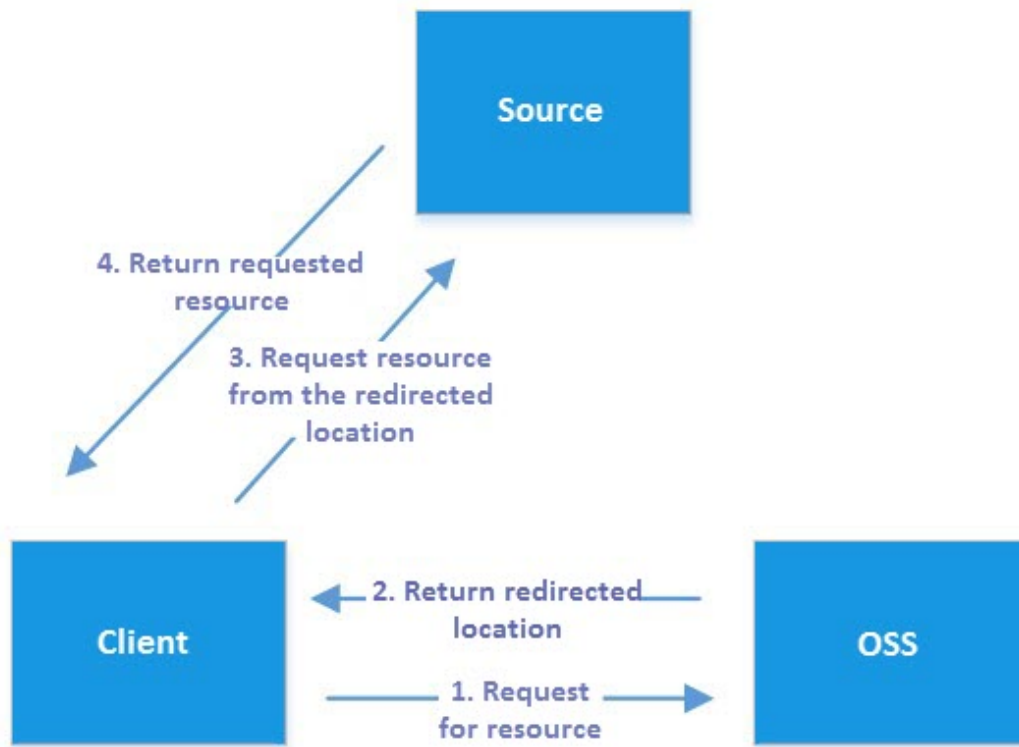
```
x-oss-tag:MIRROR http%3a%2f%2fwww.example-domain.com%2fdir1%2fimage%2fexample_object.jpg
```

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

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



Application scenarios

- Migrate data sources to OSS

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

- Configure page redirect function

If a user wants to hide objects using a certain header prefix, a customized page can be displayed to visitors.

- Configure the redirected page when a 404 or 500 error occurs

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

Reference

- Console: [Origin retrieval Rule Management](#)

10 Security management

10.1 Anti-leech settings

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

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

For example, for a bucket named oss-example, set its referer whitelist to `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 is performed only when users access objects through URL signatures or anonymously. When the request header contains the "Authorization" field, anti-leech verification is not performed.
- A bucket supports multiple referer fields, which are separated by the Enter key on the console, and by the comma "," on API.
- The referer field supports the wildcard "*" and "?".
- Users can set whether to allow access requests with empty referer fields.
- When the whitelist is empty, the system checks if the referer field is null (otherwise, all requests get rejected).
- When the whitelist is not empty and the rules do not allow null referer fields, only requests with referers in the whitelist are allowed. Other requests (including null referer requests) are then rejected.
- If the whitelist is not empty and the rules allow empty referer fields, requests with empty referer and with the referers in the whitelist are allowed. Other requests get rejected.
- The three bucket permissions (private, public-read, and public-read-write) check the referer field.

Wildcard details:

- Asterisk "*": The asterisk can be used to represent 0 or multiple characters. If you are looking for an object name prefixed with AEW but have forgotten the remaining part, you can enter AEW* to search for all types of files starting with AEW, such as AEW.txt, AEW.EXE and

AEWI.dll. If you want to narrow down the search scope, you can enter AEW*.txt to search for all .txt files starting with AEW, such as AEWIP.txt and AEWDF.txt.

- Question mark “?”: The question mark can be used to represent one character. If you enter love?, all types of files starting with love and ending with one character get displayed, such as lovey and lovei. If you want to narrow the search scope, you can enter love?.doc to search for all .doc files starting with love and ending with one character, such as lovey.doc and loveh.doc.

Reference

- API: [PutBucketReferer](#)
- Console: [Set anti-leech](#)

10.2 Cross-origin resource sharing

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

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

Cross-origin

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

1. By using a header containing the origin of the HTTP request. As in the earlier example, origin header contains www.a.com.
2. After receiving the request, the server judges based on certain rules whether the request must be accepted or not. If yes, then the server attaches the Access-Control-Allow-Origin header in the response. The header contains www.a.com, indicating that cross-origin access is allowed. If the server accepts all the cross-origin requests, set the Access-Control-Allow-Origin header to *.

3. The browser determines whether the cross-origin request is successful or not based on whether the corresponding header has been returned or not. If no corresponding header is attached, the browser blocks the request. If the request is not a simple one, the browser initially send an OPTIONS request to obtain the CORS configuration of the server. If the server does not support the following operations, the browser blocks the following requests.

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

Key points

- Attaching relevant CORS headers and other actions are automatically executed by the browser , and no additional action is required by the user. Only in the browser environment could the CORS operations be meaningful.
- Whether a CORS request is accepted is completely independent of OSS authentication and other such measures. The OSS CORS rule is only used to determine whether to attach the relevant CORS headers. Whether the request is required to be blocked or not, this can be exclusively determined by the browser.
- When using cross-origin requests, make sure the browser's cache function is enabled. For example, the same cross-origin resource have been requested by two webpages running on the same browser (originated from www.a.com and www.b.com) at the same time respectively. If the request of www.a.com is received by the server initially, the server returns to the user the resource with the Access-Control-Allow-Origin header "www.a.com". When www.b.com initiates its request, the browser returns its previous cached request to the user. As the header content does not match the CORS request, the subsequent request fails.

Reference

- API: [Introduction](#)
- SDK: Java SDK-[CORS](#)
- Console: [Set CORS](#)

10.3 Server-side encryption

This article describes how to use the server-side encryption feature of OSS to encrypt and protect the persistent data in OSS:

- [Server-side encryption options](#)

- [Fully managed by OSS](#)
- [CMK managed by KMS](#)
- [Server-side encryption API usage](#)

OSS supports server-side encryption for the data uploaded by users: When a user uploads data, OSS encrypts the user data and permanently stores the data with encryption; when the user downloads the data, 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.

Main application scenarios of OSS server-side encryption

OSS protects static data by using server-side encryption, which applies to scenarios with high security or compatibility requirements for storage, such as the storage of deep learning sample files and online collaboration document data. The following server-side encryption methods are available for different application scenarios:

- Using KMS for encryption and decryption

When uploading a file, you can use a specified CMK ID or the CMK fully managed by KMS to encrypt and decrypt a large amount of data. You do not need to send user data to the KMS service side through networks for encryption and decryption, and therefore is a low-cost method.

- Using encryption fully managed by OSS

Encryption fully managed by OSS is a property of objects. When creating an object, you only need to add the HTTP header, `x-oss-server-side-encryption`, to the Put Object request and specify its value as `AES256` to encrypt and store the object on the server. This method is ideal for bulk data's encryption and decryption.

Server-side encryption options

OSS server-side encryption provides the following options for users to choose from (based on the key management policy):

- Fully managed by OSS: The generation and management of data encryption keys are conducted by OSS, which provides strong and multi-factor security measures to protect data. The data encryption algorithm adopted is the strong encryption algorithm of AES-256 (256-bit advanced encryption standard).

Server-side encryption - Fully managed by OSS

The OSS server-side encryption is a property of objects. When creating an object, the user only needs to add the HTTP header, `x-oss-server-side-encryption`, to the Put Object request and specify its value as AES256 to encrypt and store the object on the server.

The following operations support the addition of the HTTP header to requests:

- Put Object: simple upload
- Copy Object: copy an object
- Initiate Multipart Upload: multipart upload

Server-side encryption - CMK managed by KMS

When creating an object, the user only needs to add the HTTP header, `x-oss-server-side-encryption` to the request and specify its value as `askms`, to encrypt and store the object on the server (KMS service is used for key management). With the KMS service, OSS can encrypt and protect the objects stored in OSS by using strong security measures such as envelope encryption and AES-256 encryption algorithm.

KMS is a secure and easy-to-use management service provided by Alibaba Cloud. With the help of KMS, users no longer have to spend a great deal to protect the confidentiality, integrity, and availability of keys. Users can use keys securely and conveniently, and focus on developing encryption/decryption function scenarios. Users can view and manage KMS keys in the KMS console.

The following operations support adding the HTTP header to requests

- Put Object: simple upload
- Copy Object: copy an object
- Initiate Multipart Upload: multipart upload



Note:

- Users must activate the KMS service. In this mode, when a user adds objects for encryption to a bucket in a region for the first time, a default KMS key is automatically created in this region (used as CMK), which is used for server-side encryption (KMS mode).
- Users can view the information about this key in the KMS console. See [KMS User Guide](#).

Server-side encryption - RestAPI usage

Operation requests

The following APIs support adding the header `x-oss-server-side-encryption` to requests:

- Put Object: simple upload
- Copy Object: copy an object
- Initiate Multipart Upload: multipart upload

HTTP header format

Name	Description	Example
<code>x-oss-server-side-encryption</code>	Note: This option specifies the server-side encryption mode Valid values: AES256 ,KMS	<code>x-oss-server-side-encryption:KMS</code> indicates the server-side encryption mode is CMK managed by KMS



Note:

- If any other request received by OSS contains the header `x-oss-server-side-encryption`, OSS directly returns HTTP Status Code 400, with the error code in the message body being `InvalidArgument`.
- If a user specifies an invalid value for the header `x-oss-server-side-encryption`, OSS directly returns HTTP Status Code 400, with the error code in the message body being `InvalidEncryptionAlgorithmError`.

Operation response

For objects stored after server-side encryption, OSS returns the header `x-oss-server-side-encryption` for the following API requests:

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

Meta information

For objects stored in the mode of CMK managed by KMS, their Meta information includes the following fields:

Name	Description	Example
x-oss-server-side-encryption	Specifies the server-side encryption mode	x-oss-server-side-encryption: KMS
x-oss-server-side-encryption-key-id	The ID of the user KMS key used for encrypting the object	x-oss-server-side-encryption-key-id: 72779642-7d88-4a0f-8d1f-1081a9cc7afb

Related APIs

- [AppendObject](#)
- [PutObject](#)
- [CopyObject](#)
- [PostObject](#)

10.4 Set security token

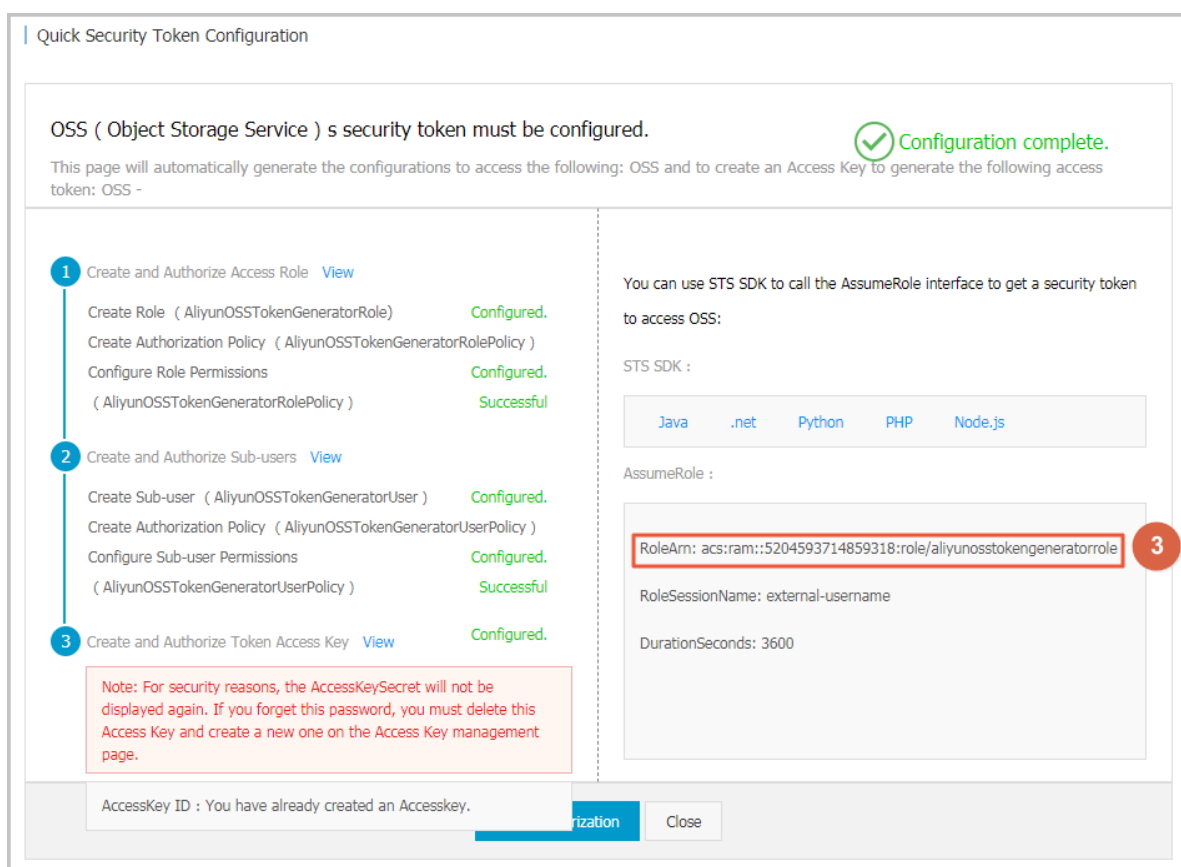
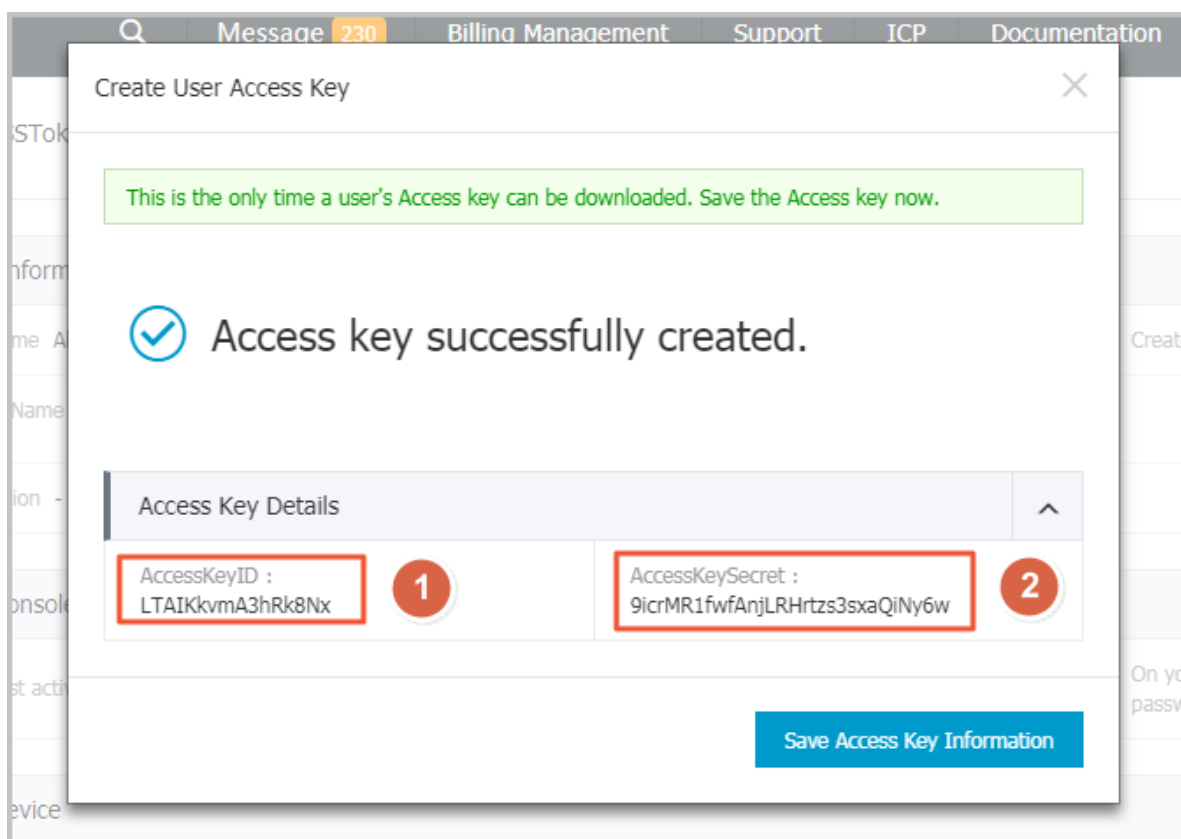
Procedure

1. Log on to the [OSS console](#). In the OSS **Overview** page, find the **Basic Settings** area, and click **Security Token**.

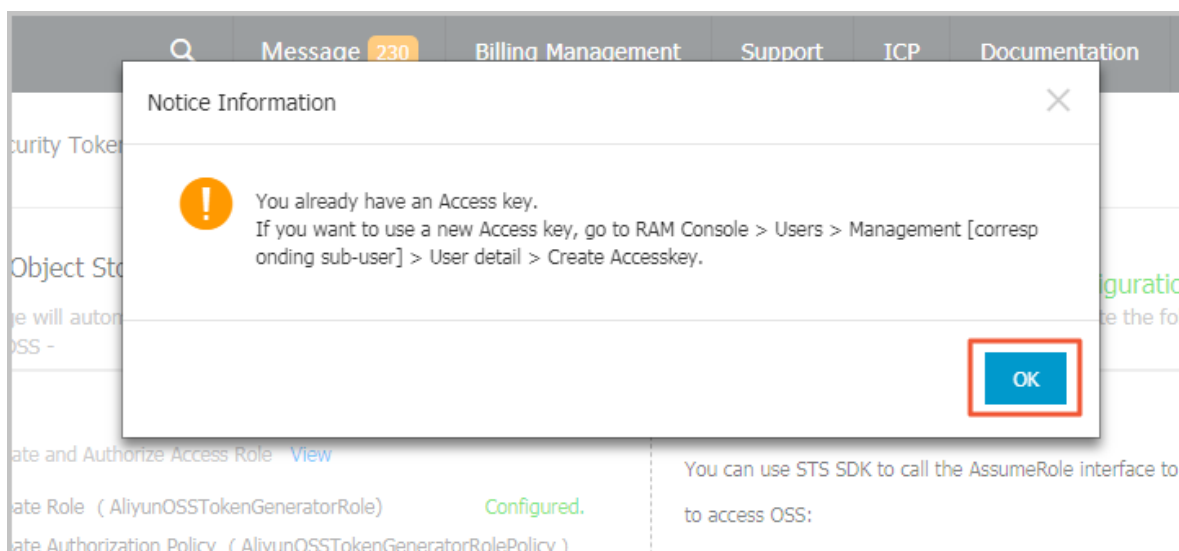
2. Enter the **Quick Security Token Configuration** page.

If RAM has not been activated yet, a dialog box to activate RAM appears. Click **Activate** and perform real-name verification. After the verification is completed, the following page appears. Click **Start Authorization**.

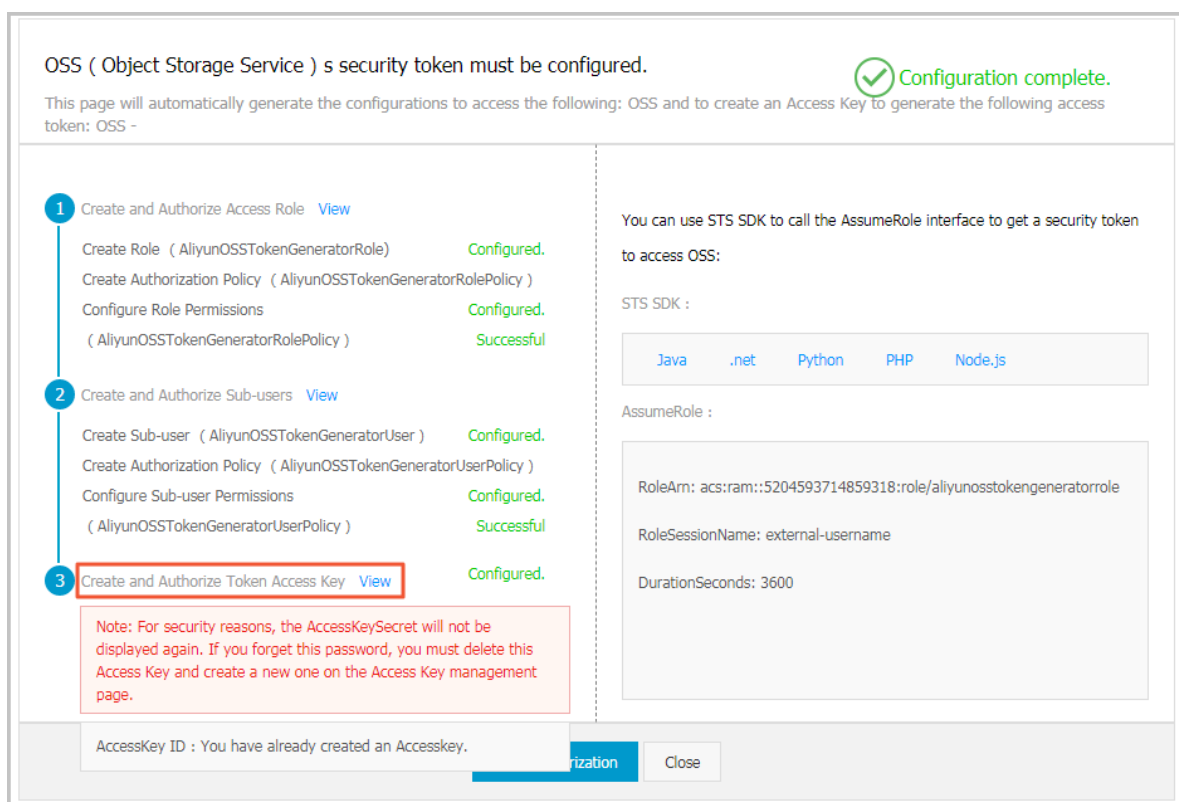
3. The system performs authorization automatically. Be sure to save the parameters in the three red boxes in the following figures. Click **Save Access Key Information** to close the dialog box and complete STS activation.



4. If you have already created an AccessKeyID/AccessKeySecret, the following dialog box appears:



Click **View**, as shown in the following figure.



5. Click **Create Access Key**, as shown in the following figure.

User Details

User Authorization P...

User Groups

Basic Information

Edit Basic Information

User Name AliyunOSSTokenGeneratorUser UID 202721711754921046 Created At 2017-11-27 11:55:21

Display Name

Description -

Web Console Logon Management

Enable Console Logon

You must activate MFA Close Last Logon Time: On your next logon you must reset the password. Close

MFA Device

Type Introduction Enabling Status Actions

VMFA Device Application calculates a 6-digit verification code using the TOTP standard algorithm. Not Enabled Enable VMFA Device

User Access Key

Create Access Key

AccessKey ID Status Created At Actions

LTAIBpjam9pwG3DA Enable 2018-01-29 10:04:07 Disable Delete

Record parameters 1, 2, and 3.

Create User Access Key

This is the only time a user's Access key can be downloaded. Save the Access key now.

✓ Access key successfully created.

Access Key Details

AccessKeyID : LTAIKkvmA3hRk8Nx 1

AccessKeySecret : 9icrMR1fwfAnjLRHrtzs3sxaQiNy6w 2

Save Access Key Information

Quick Security Token Configuration

OSS (Object Storage Service) s security token must be configured.

Configuration complete.

This page will automatically generate the configurations to access the following: OSS and to create an Access Key to generate the following access token: OSS -

1

Create and Authorize Access Role [View](#)

Create Role (AliyunOSSTokenGeneratorRole)

Configured.

Create Authorization Policy (AliyunOSSTokenGeneratorRolePolicy)

Configured.

Configure Role Permissions (AliyunOSSTokenGeneratorRolePolicy)

Successful

2

Create and Authorize Sub-users [View](#)

Create Sub-user (AliyunOSSTokenGeneratorUser)

Configured.

Create Authorization Policy (AliyunOSSTokenGeneratorUserPolicy)

Configured.

Configure Sub-user Permissions (AliyunOSSTokenGeneratorUserPolicy)

Successful

3

Create and Authorize Token Access Key [View](#)

Configured.

Note: For security reasons, the AccessKeySecret will not be displayed again. If you forget this password, you must delete this Access Key and create a new one on the Access Key management page.

You can use STS SDK to call the AssumeRole interface to get a security token to access OSS:

STS SDK :

Java

.net

Python

PHP

Node.js

AssumeRole :

RoleArn: acs:ram::5204593714859318:role/aliyunostokengeneratorrole

RoleSessionName: external-username

DurationSeconds: 3600

AccessKey ID : You have already created an Accesskey.

ization

Close

Once you save all the three parameters, the STS gets successfully activated.

11 Static website hosting

11.1 Configure static website hosting

In the [OSS console](#), you can set up your buckets to work in static website hosting mode.

If your selected bucket is located in Hangzhou, after the configuration takes effect, the endpoint of the static website is as follows:

```
http://<Bucket>.oss-cn-hangzhou.aliyuncs.com/
```

**Note:**

When you use an OSS endpoint in Mainland China regions or the Hongkong region to access a web file through the Internet, the Content-Disposition: 'attachment=filename;' is automatically added to the Response Header, and the web file is downloaded as an attachment. If you access OSS with a user domain, the Content-Disposition: 'attachment=filename;' will not be added to the Response Header. For more information about using the user domain to access OSS, see [Bind a custom domain name](#).

For users to manage static websites hosted on the OSS more easily, the OSS provides two functions:

- Index Document Support

The index document refers to the default index document (such as index.html) that is returned by the OSS when a user directly accesses the root domain name of the static website. If static website hosting mode is set for a bucket, you must specify the index document as an object in that bucket. This setting is required.

- Error Document Support

The error document refers to the error page the OSS returns to a user if the HTTP 4XX error (such as 404 "NOT FOUND") occurs when the user attempts to access the static website but fails. If static website hosting mode is set for a bucket, you must specify the error document as an object in that bucket. This setting is optional.

For example, if a user sets:

- The index document support as index.html
- The error document support as error.html
- The bucket as oss-sample

- The endpoint as `oss-cn-hangzhou.aliyuncs.com`

Then:

- When a user accesses `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/` and `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/directory/`, it is the same as accessing `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/index.html`.
- When a user accesses `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/object`, and the object does not exist, OSS returns `http://oss-sample.oss-cn-hangzhou.aliyuncs.com/error.html`.

Detail analysis

- Static websites are websites with web pages composed of static content, including scripts such as JavaScript executed on the client. OSS does not support content that needs to be processed by the server, such as PHP, JSP, and ASP.NET content.
- For access to a bucket-based static website through a user-defined domain name, see [Bind custom domain names](#).
- When you set a bucket to static website hosting mode, you must specify an index page, the error page is optional.
- When you set a bucket to static website hosting mode, the specified index page and error page must be an object in the bucket.
- After a bucket is set to static website hosting mode, the OSS returns the index page for anonymous access to the root domain name of the static website, and returns Get Bucket results for signed access to the root domain name of the static website.
- After a bucket is set to static website hosting mode, and the user accesses the root domain name of a static website or a nonexistent object, the OSS returns a specified object to the user and bills the return traffic and requests to the bucket owner.

Reference

- API: [PutBucketWebsite](#)
- Console: [Static website hosting](#)
- Java SDK: [Static website hosting](#)

11.2 Tutorial: Host a static website using a custom domain name

Suppose that you want to host a static website on Alibaba Cloud Object Storage Service (OSS). You have registered a domain (for example, `examplewebsite.com`), and you want the requests for `http://examplewebsite.com` and `http://www.examplewebsite.com` to be serviced from your OSS content. Whether you have an existing static website that you want to host on OSS, or you are starting from scratch, you can use this example and learn how to host websites on Alibaba Cloud OSS.

Prerequisites

This tutorial covers the following services:

- Domain name registration

If you do not have a registered domain name, such as `examplewebsite.com`, select a registrar to register one. Alibaba Cloud also provides domain name registration service. For more information, see [Alibaba Cloud Domain service](#).

- Alibaba Cloud OSS

You use Alibaba Cloud OSS to create buckets, upload a sample website page, configure permissions to let others see the content, and then configure the buckets for website hosting. In this example, because you want to allow requests for `http://examplewebsite.com` and `http://www.examplewebsite.com`, you create two buckets. You host content in only one bucket, and configure the other bucket to redirect requests to the bucket that hosts the content.

- Alibaba Cloud DNS

As your Domain Name System (DNS) provider, you configure Alibaba Cloud DNS. In this example, you add your domain name to Alibaba Cloud DNS and define a CNAME record so that you can use your domain name instead of the OSS assigned access domain name to access your OSS buckets.

In this example, we use Alibaba Cloud DNS. We recommend that you use Alibaba Cloud DNS. However, you can use various registrars to define a CNAME record that points to an OSS bucket.

**Note:**

This tutorial uses `examplewebsite.com` as a domain name. Replace this domain name with the one that you have registered.

Step 1: Register a domain

If you already have a registered domain, you can skip this step. If you have never hosted a website, your first step is to register a domain, such as `examplewebsite.com`. You can use Alibaba Cloud Domain service to register a domain.

For more information, see [Buy a domain name](#) in the Alibaba Cloud Domain Quick Start.

Step 2: Create and configure buckets and upload data

You create two buckets to support requests from both the root domain such as `examplewebsite.com` and subdomain such as `http://www.examplewebsite.com`. One bucket is used to store the content, and the other bucket is used to redirect requests to the bucket that stores the content.

Step 2.1: Create two buckets

In this step, you log on to Alibaba Cloud OSS console with your Alibaba Cloud account credentials and create the following two buckets:

- **originbucket**: to store the content
- **redirectbucket**: to redirect requests to **originbucket**

1. Log on to the [OSS console](#).
2. Create two buckets, for example, **originbucket** and **redirectbucket**, one to store the content, and the other to redirect requests to the bucket that stores the content. Set the access control list (ACL) of the two buckets to Public Read so that everyone can see the content of the buckets.

For detailed procedures, see [Create a bucket](#).

3. Make yourself a note of the access domain name of the **originbucket** and **redirectbucket**. You will use them in later steps. You can find the access domain name of a bucket on the **Overview** tab page of the bucket, as shown in the following figure.

Object Storage **originbucket** Type: Standard Storage Region: China North 2 (Beijing) Created At: 02/28/2018, 10:17 [Delete Bucket](#)

Overview | **Overview** | Files | Basic Settings | Domain Names | Image Processing | Basic Data | Hotspot Statistics | API Statistics | Object Access Statistics

Basic Data

Data in the Overview page and Bucket Overview page is not in real time. It is delayed for two to three hours.

Storage Used	Total Used	Internet Traffic This Month	Inbound	Requests This Month	PUT	Files	File Fragments
355 Byte		0 Byte		0		2	0
Month-On-Month -- Day-On-Day 0.00%		Internet Traffic Last Month 3.16KB		Requests Last Month 7			

Access Domain Name

	Endpoint	Access Domain Name	HTTPS
Internet Access	oss-cn-beijing.aliyuncs.com	originbucket.oss-cn-beijing.aliyuncs.com	Supported
ECS Address for Classic Network Access (Intranet)	oss-cn-beijing-internal.aliyuncs.com	originbucket.oss-cn-beijing-internal.aliyuncs.com	Supported
ECS Address for VPC Network Access (Intranet)	oss-cn-beijing-internal.aliyuncs.com	originbucket.oss-cn-beijing-internal.aliyuncs.com	Supported

4. Upload your website data to **originbucket**.

You will host your content out of the root domain bucket **originbucket**, and you will redirect requests for the subdomain bucket **redirectbucket** to the root domain bucket **originbucket**. You can store content in either bucket.

For this example, you host content in the **originbucket** bucket. The content can be any type of files, such as text files, photos, and videos. If you have not yet created a website, then you only need two files for this example. One file is used as the homepage of the website, and the other file is used as the error page of the website.

For example, you can create one file named `index.html` using the following HTML and upload it to the bucket. In a later step, you use this file name as the default homepage for your website.

```
<html>
  <head>
    <title>Hello OSS! </title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>Now host on Alibaba Cloud OSS</p>
    <p>This is the index page</p>
  </body>
</html>
```

You create another file named `error.html` using the following HTML and upload it to the bucket. This file is used as the 404 error page of a website. In a later step, you use this file name as the default 404 page for your website.

```
<html>
<head>
  <title>Hello OSS! </title>
  <meta charset="utf-8">
</head>
<body>
```

```
<p>This is the 404 error page</p>
</body>
</html>
```

Step 2.2: Configure buckets for website hosting

When you configure a bucket for website hosting, you can access the website using the OSS assigned access domain name.

In this step, you configure originbucket as a website.

1. Log on to the [OSS console](#).
2. From the bucket name list, select originbucket.
3. Click the **Basic Settings** tab and find the **Static Page** area.
4. Click **Edit**, and then enter the following information:
 - **Default Homepage:** The index page (equivalent to index.html of the website). Only HTML files that have been stored in the bucket can be used. For this example, enter `index.html`.
 - **Default 404 Page:** The default 404 page returned when an incorrect path is accessed. Only HTML and image files that have been stored in the bucket can be used. If this field is left empty, the default 404 page is disabled. For this example, enter `error.html`.
5. Click **Save**.

Step 2.3: Configure the index page for redirect

Now that you have configured the default homepage and error page of the **originbucket**, you also need to configure the default homepage of **redirectbucket**.

To configure the index page for redirect, follow these steps:

1. Log on to the [OSS console](#).
2. From the bucket name list, select redirectbucket.
3. Click the **Basic Settings** tab and find the **Static Page** area.
4. Click **Edit**, and then enter `index.html` in the **Default Homepage** text box.
5. Click **Save**.

Step 3: Bind your domain name to your OSS buckets

Now that you have your root domain `examplewebsite.com` and your OSS bucket **originbucket**, bind your domain to your OSS buckets so that you can access the OSS buckets using your own domain name instead of the domain name assigned by OSS.

In this example, before you bind your domain `examplewebsite.com` to your OSS bucket **originbucket**, you have to use the OSS assigned domain name `originbucket.oss-cn-beijing.aliyuncs.com` to access your bucket `originbucket`. After you bind your domain `examplewebsite.com`, you can use this `examplewebsite.com` to access your OSS bucket.

Similarly, you also need to bind your subdomain `www.examplewebsite.com` to your OSS bucket **redirectbucket**, so that you can use `www.examplewebsite.com` instead of the OSS assigned domain name `originbucket.oss-cn-beijing.aliyuncs.com` to access your OSS bucket.

To bind your root domain `examplewebsite.com` to your OSS bucket **originbucket**, follow these steps:

1. Log on to the [OSS console](#).
2. From the bucket name list, select `originbucket`.
3. Click the **Domain Names** tab.
4. Click **Bind User Domain** to open the Bind User Domain dialog box.
5. In the **User Domain** text box, enter the root domain `examplewebsite.com`.
6. Define a CNAME record to `originbucket`.
 - If your domain name has been resolved under your Alibaba Cloud account, you can open the **Add CNAME Record Automatically** switch. Then click **Submit**.
 - If your domain name has not been resolved under your Alibaba Cloud primary account, the **Add CNAME Record Automatically** switch is disabled. Follow these steps to add a CNAME record manually, and then click **Submit**.
 1. Add your domain name in Alibaba Cloud DNS.
 - If your domain name is registered with Alibaba Cloud, it is automatically added to the Alibaba Cloud DNS list. You can skip this step.
 2. In the Alibaba Cloud DNS console, find your domain name.
 3. Click the domain name or click the **Configure** link.
 4. Click **Add Record**.
 5. In the **Add Record** dialog box, select CNAME from the **Type** drop-down box, and enter the OSS domain name of the bucket in the **Value** text box. In this example, enter **originbucket.oss-cn-beijing.aliyuncs.com**.
 6. Click **Confirm**.
7. Follow the preceding steps to bind your sub domain `www.examplewebsite.com` to your OSS bucket **redirectbucket**.

Step 4: Configure your website redirect

Now that you have configured your bucket for website hosting and bound your custom domain to your OSS bucket, configure the **redirectbucket** to redirect all requests for `http://www.examplewebsite.com` to `http://examplewebsite.com`.

To configure your website redirect, follow these steps:

1. Log on to the [OSS console](#).
2. From the bucket name list, select **redirectbucket**.
3. Click the **Basic Settings** tab and find the **Back-to-Origin** area.
4. Click **Edit**, and then click **Create Rule**.
5. Create the 404 redirect rule as follows:
 - a. In the Back-to-Origin Type area, select **Redirect**.
 - b. In the **Back-to-Origin When** area, set HTTP Status Code to **404**.
 - c. In the **Back-to-Origin URL** area, select **Add a prefix or suffix**, enter your domain name of the **originbucket**. In this example, enter **examplewebsite.com**.
 - d. Click **OK**.

Step 5: (Optional) Speed up your website with Alibaba Cloud CDN

You can use Alibaba Cloud Content Delivery Network (CDN) to improve the performance of your website. CDN makes your website files (such as HTML, images, and video) available from data centers around the world. These are called edge nodes. When a visitor requests a file from your website, Alibaba Cloud CDN automatically redirects the request to a copy of the file at the nearest edge node. This results in faster download times than if the visitor had requested the content from a data center that is located farther away.

Alibaba Cloud CDN caches content at edge nodes for a period of time that you specify. If a visitor requests content that has been cached for longer than the expiration date, CDN checks the origin server to see if a later version of the content is available. If a later version is available, CDN copies the new version to the edge node. Changes that you make to the original content are replicated to edge nodes as visitors request the content. However, before the expiration date, the content is still in the earlier version. We recommend that you open the **Auto Refresh CDN Cache** switch, so that changes you make to the original content are automatically refreshed in CDN cache in real time.

To speed up **originbucket** with CDN, follow these steps:

1. Add a CDN domain in the CDN console. For detailed procedures, see Step 2. Add a CDN domain in [CDN quick start](#).
2. Define a CNAME record in the Alibaba Cloud DNS console. For detailed procedures, see [Configure Alibaba Cloud DNS](#).
3. Open the **Auto Refresh CDN Cache** switch in the OSS console.
 - a. Log on to the [OSS console](#).
 - b. From the bucket name list, select `originbucket`.
 - c. Click the **Domain Names** tab.
 - d. Open the **Auto Refresh CDN Cache** switch.
4. Follow the preceding steps to speed up `redirectbucket` with CDN.

Step 6: Test the website

To verify that the website is working correctly, in your browser, try the following URLs:

URL	Result
<code>http://examplewebsite.com</code>	Displays the index document in the originbucket .
The URL of a file that does not exist in the originbucket , for example, <code>http://examplewebsite.com/abc</code>	Displays the error document in the originbucket .
<code>http://www.examplewebsite.com</code>	Redirects your request to <code>http://examplewebsite.com</code> .
<code>http://www.examplewebsite.com/abc</code>	Redirects your request to <code>http://examplewebsite.com/abc</code> .



Note:

In some cases, you may need to clear the cache of your web browser to see the expected behavior.

Cleanup

If you created your static website as a learning exercise only, remember to delete the Alibaba Cloud resources that you allocated to avoid unnecessary fees charged to your account. After you delete your Alibaba Cloud resources, your website is no longer available.

To clean up, follow these steps:

1. Disable and then remove the domain name in the Alibaba Cloud CDN console.

- 2.** Delete the CNAME records in the Alibaba Cloud DNS console.
- 3.** Delete the OSS files and buckets in the Alibaba Cloud OSS console.

12 Monitoring service

12.1 Monitoring service overview

The OSS monitoring service details metric data, including basic system operation statuses, performance, and metering. It also provides a custom alarm service to track requests, analyze usage, collect statistics on business trends, and promptly discover and diagnose system problems.

OSS metric indicators are classified into indicator groups such as basic service indicators, performance indicators, and metering indicators. For more information, see [Monitoring indicators reference](#).

High real-time performance

Real-time performance monitoring can expose potential peak/valley problems, display actual fluctuations, and provide insights into the analysis and evaluation of business scenarios. OSS real-time metric indicators (excluding the metering indicator) enable minute-level collection and aggregation of metric data with an output delay of less than one minute.

Metering indicator description

The metering indicator uses the following features:

- Metering entries are collected, aggregated, and output at the hour-level.
- However, the output delay can be up to thirty minutes.
- The time of metering refers to the start time of the relevant statistical period.
- The metering acquisition cutoff time is the end time of the last metering data statistical period for the current month. If no metering data is produced during the current month, the metering data acquisition cutoff time is 00:00 on the first day of the current month.
- A maximum amount of metering entries is pushed for presentation. For precise metering data, go to Billing Management and click [Usage Records](#).

For example, suppose that the user just uses the request to upload data, an average of 10 times a minute. So at 2016-05-10 08:00:00 to 2016-05-10 09:00:00 this hour of time, the measured data value of the user's number of put class requests is 600 times (10*60 minutes), data time 2016-05-10 At 08:00:00, the data will be output at about 09:30:00. If this data is from 2016-05-01 From 00:00:00 to the present, the last measure monitoring data, then the cut-off time of the month for the

measure data acquisition is 2016-05-10 09:00:00. If the user did not produce any measurement data in May 2016, the cut off time for the measurement collection is 2016-05-01 00:00:00.

OSS alarm service

You can set up to 1,000 alarm rules.

Alarm rules can be configured for other metric indicators, which can then be added to alarm monitoring. Additionally, multiple alarm rules may be configured for a single metric indicator.

- For information about the alarm service, see [Alarm service overview](#).
- For instructions about how to use the OSS alarm service, see [Alarm service user guide](#).
- For more information about OSS metric indicators, see [Monitoring indicators reference](#).

Metric data retention policy

Metric data is retained for 31 days and is automatically cleared upon expiration. To analyze metric data offline, or download and store historical metric data for longer than 31 days, use the appropriate tool or input code to read the data storage of CloudMonitor. For more information, see [Metric data access through the API](#).

The console displays metric data up until the past seven days. To view historical metric data earlier than seven days, use the CloudMonitor SDK. For more information, see [Metric data access through the API](#).

Metric data access through the API

The API of CloudMonitor allows you to access OSS metric data. For usage information, see:

- [CloudMonitor API Reference](#)
- [Cloud monitoring SDK Reference](#)
- [Metric item reference](#)

Monitoring, diagnosis, and troubleshooting

The following documentation provides monitoring, diagnosis, and troubleshooting details related to OSS management:

- [Real-time service monitoring](#)

Describes how to use the monitoring service to monitor the running status and performance of OSS.

- [Tracking and diagnosis](#)

Describes how to use the OSS monitoring service and logging function to diagnose problems, and how to associate relevant information in log files for tracking and diagnosis.

- [Troubleshooting](#)

Describes typical problems and corresponding troubleshooting methods.

Considerations

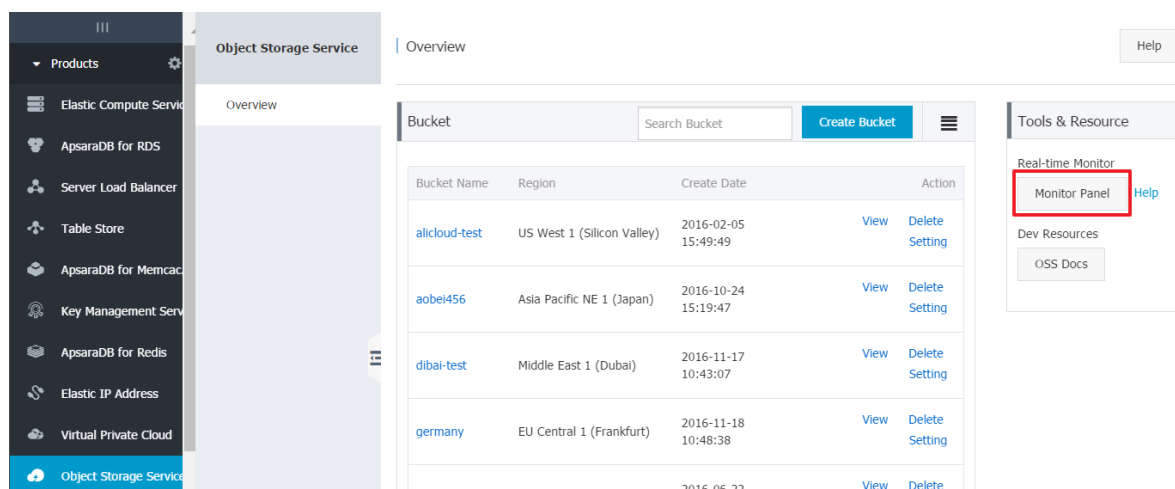
OSS buckets must be globally unique. If, after deleting a bucket, you create another bucket with the same name as the deleted bucket, the monitoring and alarm rules set for the deleted bucket are applied to the new bucket.

12.2 Monitoring service user guide

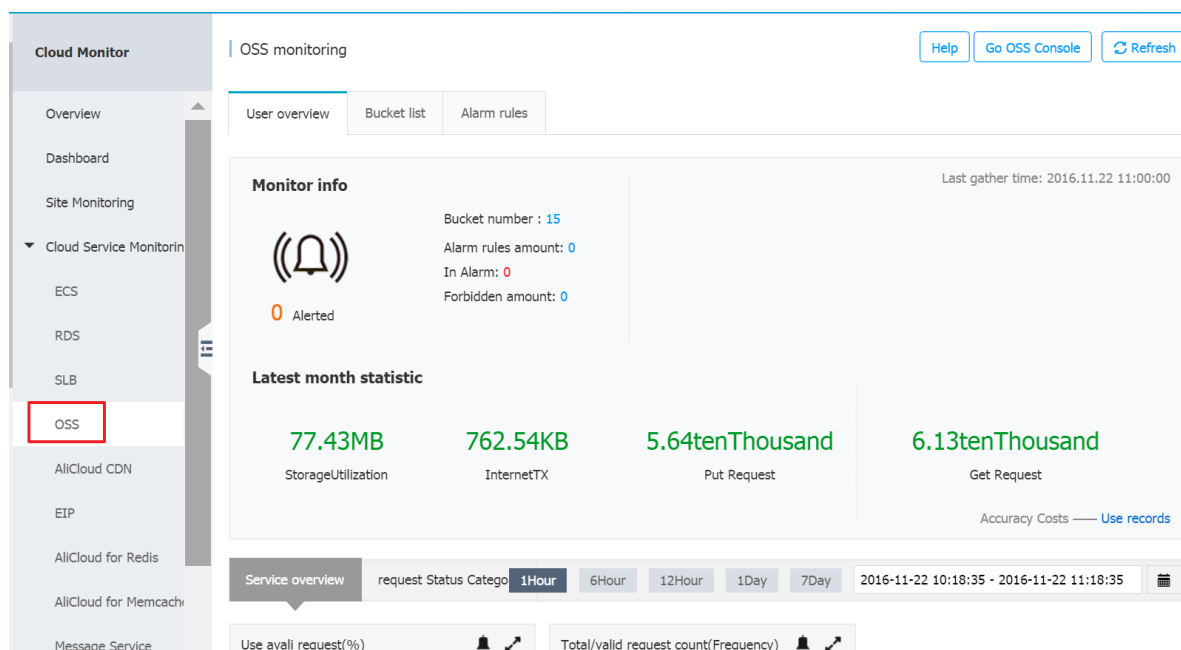
OSS monitoring entry

The OSS monitoring service is available on the Alibaba Cloud Console. You can access the OSS monitoring service in either of the following ways:

- Log on to the [OSS console](#) and then click **Set alarm rules** on the right side of OSS overview page.



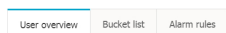
- Log on to the [CloudMonitor console](#) to view the OSS monitoring service.



OSS monitoring page

The OSS monitoring page consists of the following three tabs:

- 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.

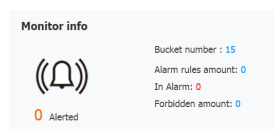


User overview

The User overview page displays monitoring information at the user level, including User monitoring information, Latest month statistics and User-level metric indicators.

- User monitoring information

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

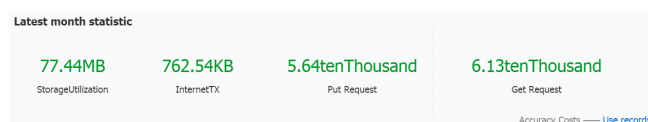


- ■ The parameters are as follows: Click the number next to **Bucket number** to display all the buckets you have created.
- Click the number next to Alarm rules amount, In Alarm, Forbidden amount, or Alerted to display the following information: **Alarm rules amount** refers to total number of alarm rules you have set.
- **In Alarm** refers to alarms in alarm state.
- **Forbidden amount** refers to alarms that are currently disabled.
- Alerted refers to alarms recently changed to alarm state
- Latest month statistics

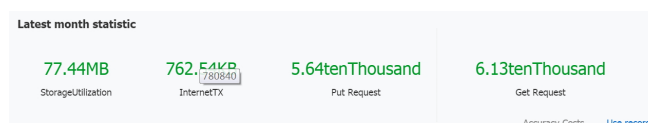
This module displays information about charged OSS resources that you have used during the period from 00:00 on the first day of the current month, to the metering acquisition cutoff time.

The following indicators are displayed:

- Storage utilization
- Internet traffic
- Put requests
- Get requests



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



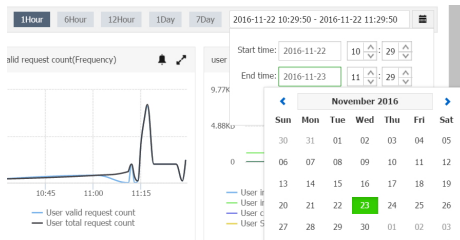
- User-level metric indicators

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



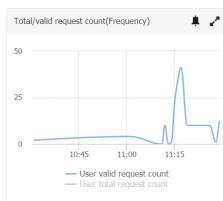
You can select a pre-determined time range, or define a time range in the custom time boxes, to display the corresponding metric chart or table.

- 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.

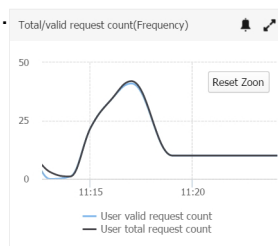


Metric charts/tables support the following display modes:

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



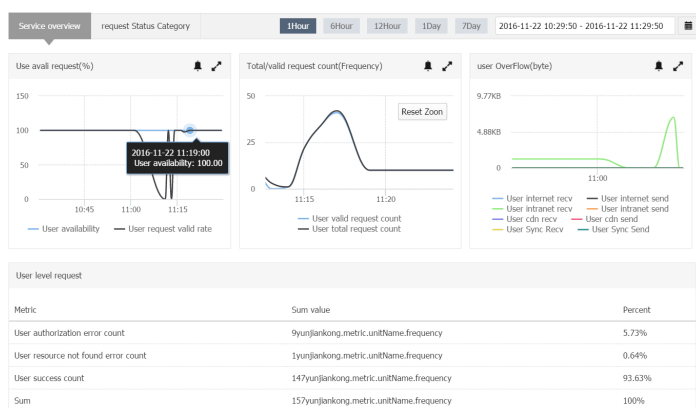
- Click the  icon in the upper-right corner of a metric chart to zoom in on the chart. Be note that tables cannot be zoomed in.
- Click the  icon in the upper-right corner of a metric chart to configure alarm rules for the displayed metric indicators. For more information, see the [Alarm Service User Guide](#). Be note that you cannot set alarm rules for tables and metering reference indicators.
- Place the cursor inside the curve area of a chart, and press and hold the left button on the mouse while dragging the mouse to extend the time range. Click **Reset Zoom** to restore the original time range.



- 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.

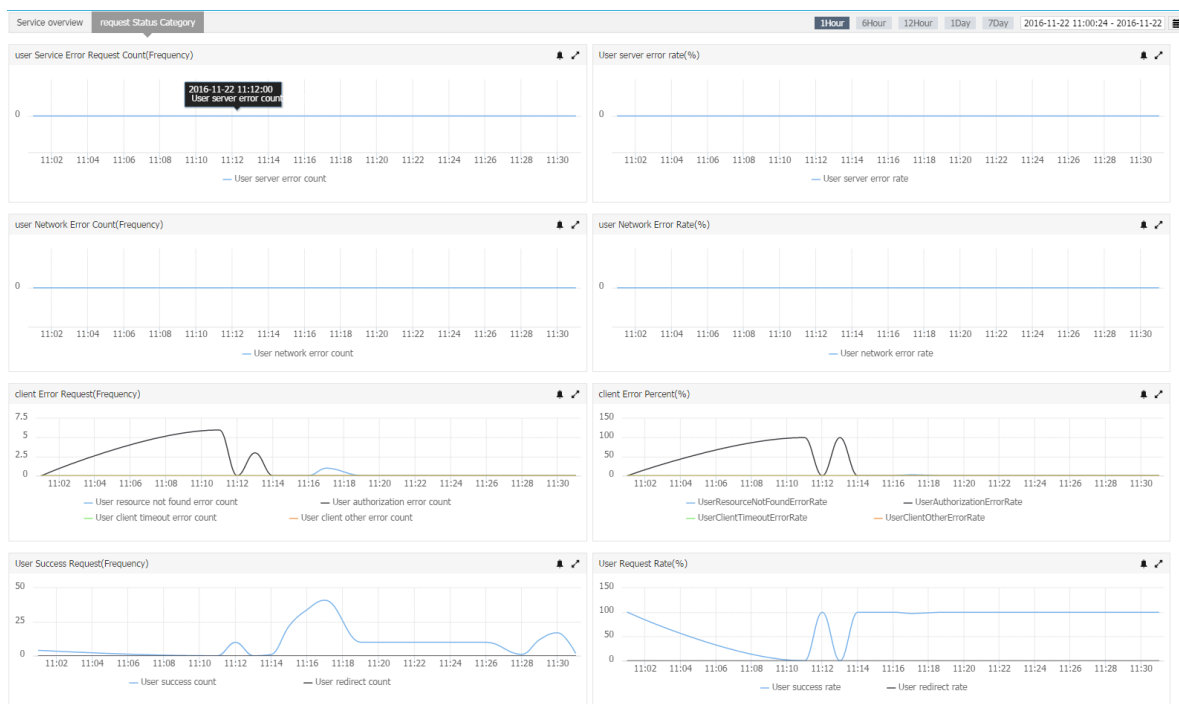


- Request status category

The "Request State Details" page shows the metric data of request state distribution through the following main 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 time-out 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 time-out 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.

OSS monitoring

Go OSS Console

Refresh

User overview

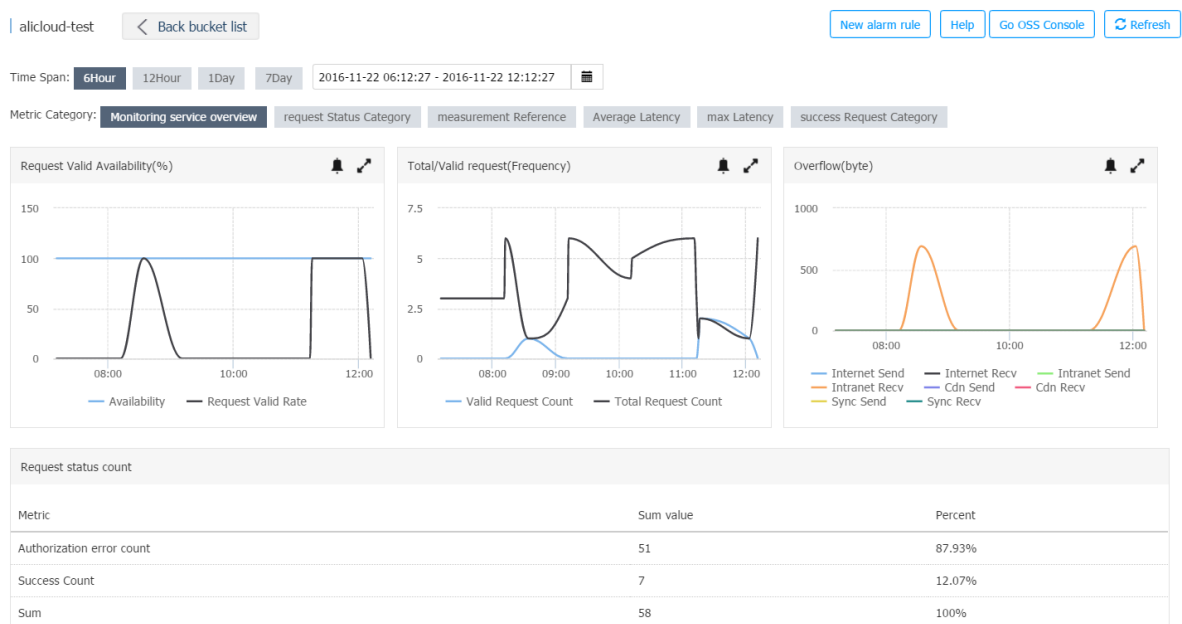
Bucket list

Alarm rules

Bucket name	Region	Created time	Store Size	Internet out	Get request	Monthly Data (Deadline: 2016.11.22 11:00:00)		Actions
						Put request		
 alicloud-test	USA West 1	2016-02-05 15:49:49	863.58KB	0B	123yunjankong.metric.unittName.frequency	385yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 adcenf56	Japan	2016-10-24 15:19:47	381.03KB	0B	105yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 dbbar-test	Dubai	2016-11-17 10:43:07	0B	0B	39yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 germany	Germany 1	2016-11-18 10:48:38	0B	0B	19yunjankong.metric.unittName.frequency	1yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 hang-ep	East China 1	2016-06-22 14:05:57	68.23MB	0B	6.05yunjankong.metric.unittName.tenThousand	5.60yunjankong.metric.unittName.tenThousand		Monitoring chart Alarm rules
 helloglobal	North China 2	2016-04-13 12:15:47	9B	9B	72yunjankong.metric.unittName.frequency	11yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 leo-test-bid	East China 2	2016-07-14 13:37:00	7.87MB	0B	103yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 raymondtest	Hong Kong	2016-09-19 12:22:14	0B	0B	105yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 real-test-dba	Dubai	2016-11-17 15:42:19	130.35KB	0B	21yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
 real-test2	East China 2	2016-11-17 15:43:17	0B	0B	18yunjankong.metric.unittName.frequency	0yunjankong.metric.unittName.frequency		Monitoring chart Alarm rules
Setting Custom monitor alarm rules.						Total: 15	10	1 2

- Display parameters are as follows: The metering statistics of the current month display the storage size, Internet outbound traffic, Put request count, and Get request count for each bucket.
- Click Monitoring chart or the corresponding bucket name to go to the bucket monitoring view page.
- Click Alarm rules next to your expected bucket, or go to the Alarm rules tab to display all alarm rules of the bucket.
- Enter the expected bucket name in the search box in the upper left-corner to display the bucket (fuzzy match is supported).
- Select the check boxes before the expected bucket names and click Setting custom monitor alarm rules to batch set alarm rules. For more information, see the [Alarm Service User Guide](#).
- Bucket-level monitoring view

Click **Monitoring chart** next to the expected bucket name in the bucket list to go to the bucket monitoring view.



The bucket monitoring view displays metric charts based on the following six indicator groups:

- Monitoring service overview
- Request status category
- Measurement reference
- Average latency

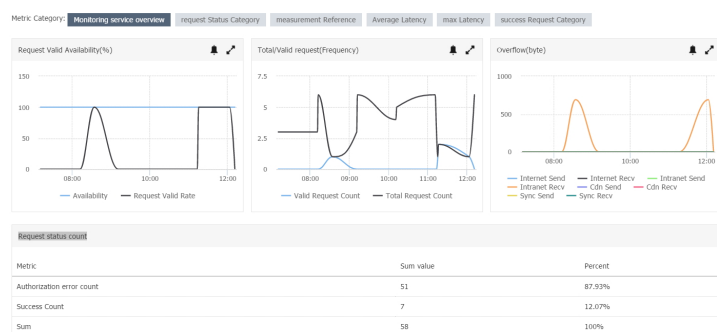
- Maximum latency
- Success request category

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.

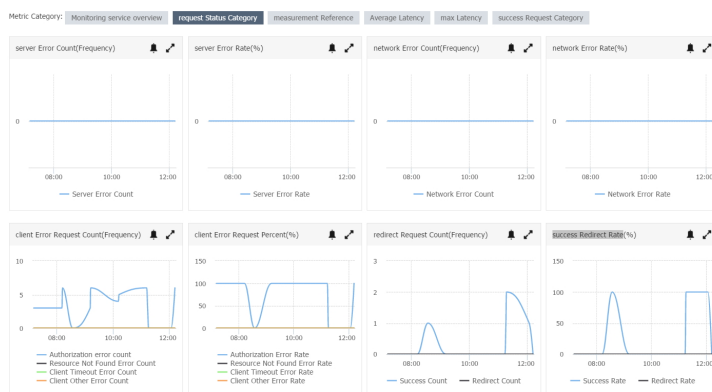


— Request status category

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:

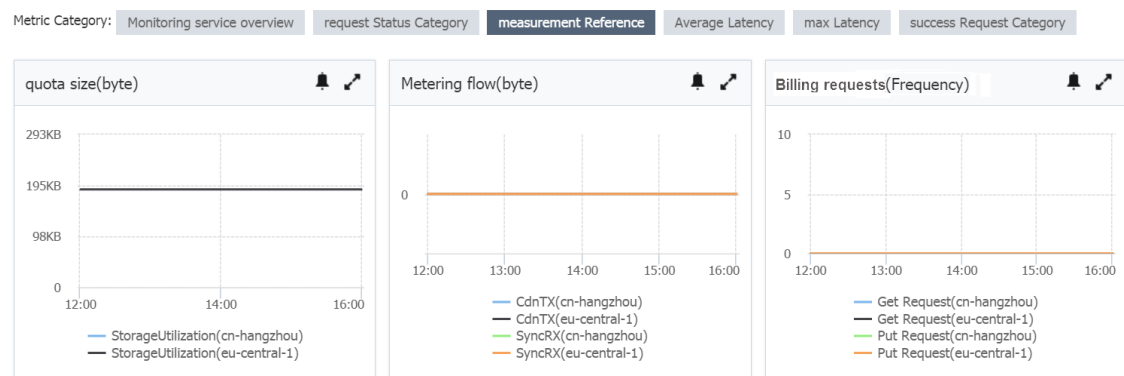
- Server error count
- Server error rate
- Network error count

- Network error request rate
- Client error request count, which includes four metric indicators: number of error requests indicating resource not found, number of authorization error requests, number of client-site time-out error requests, and number of other client-site error requests
- Client error request percent, which includes four metric indicators: percentage of error requests indicating resource not found, percentage of authorization error requests, percentage of client-site time-out error requests, and percentage of other client-site error requests
- Redirect request count, which includes two metric indicators: number of successful requests and number of redirect requests
- Success redirect rate, which includes two metric indicators: percentage of successful requests and percentage of redirect requests



— Measurement reference

The metering reference group shows metering indicators with an hourly collection and representation granularity, as shown in the following figure:



The metering metric charts include:

- Quota size

- Overflow
- Billing requests, which includes the Get request count and Put request count.

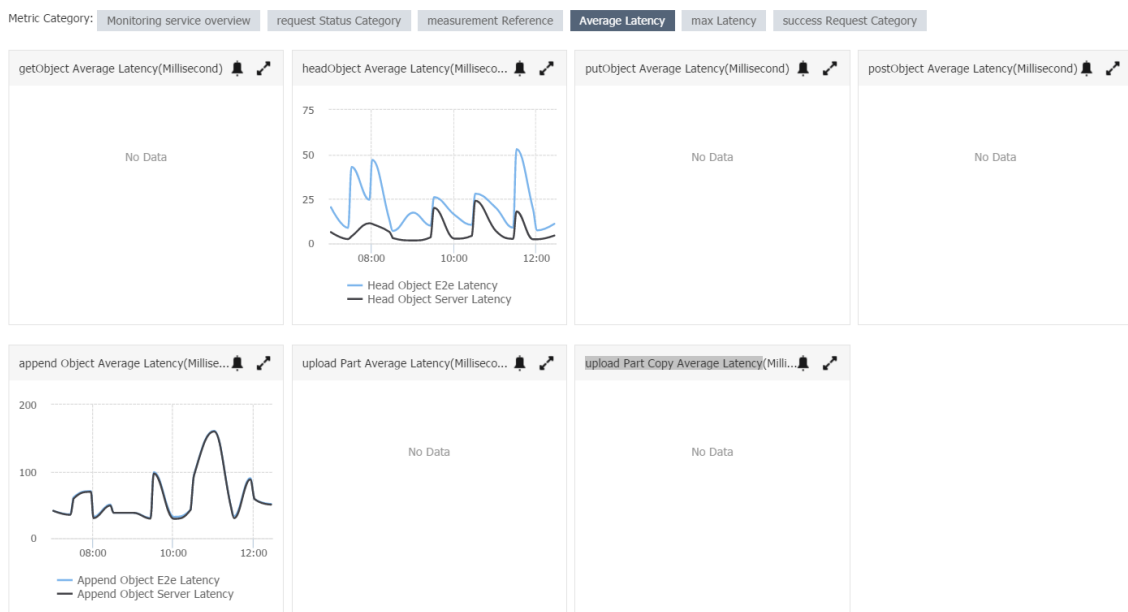
After a bucket is created, new data is collected in the next hour on the hour following the current time point, and the collected data will be displayed within 30 minutes.

— Average latency

This indicator group contains the average latency indicators of API monitoring. The metric charts include:

- getObject Average Latency
- headObject Average Latency
- putObject Average Latency
- postObject Average Latency
- append Object Average Latency
- upload Part Average Latency
- upload Part Copy Average Latency

Each metric chart shows the corresponding average E2E latency and average server latency. See the figure below:



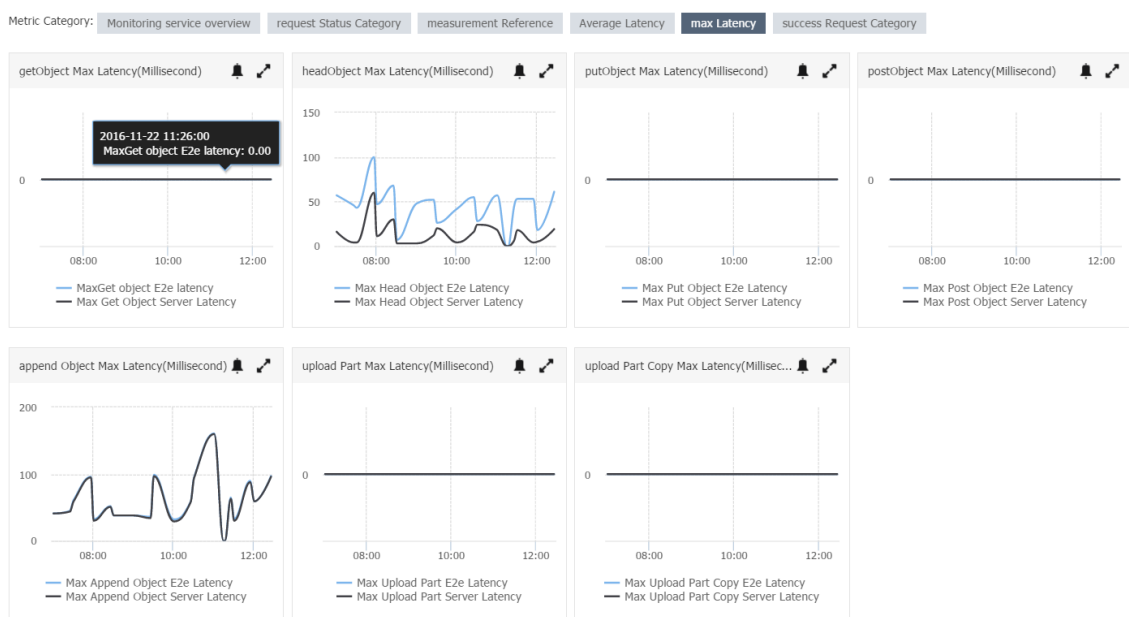
— Maximum latency

This indicator group contains the maximum latency indicators of API monitoring. The metric charts include:

- getObject Max Latency(Millisecond)

- headObject Max Latency
- putObject Max Latency
- postObject Max Latency
- append Object Max Latency
- upload Part Max Latency
- upload Part Copy Max Latency

Each metric chart shows the corresponding maximum E2E latency and maximum server latency. See the following figure:

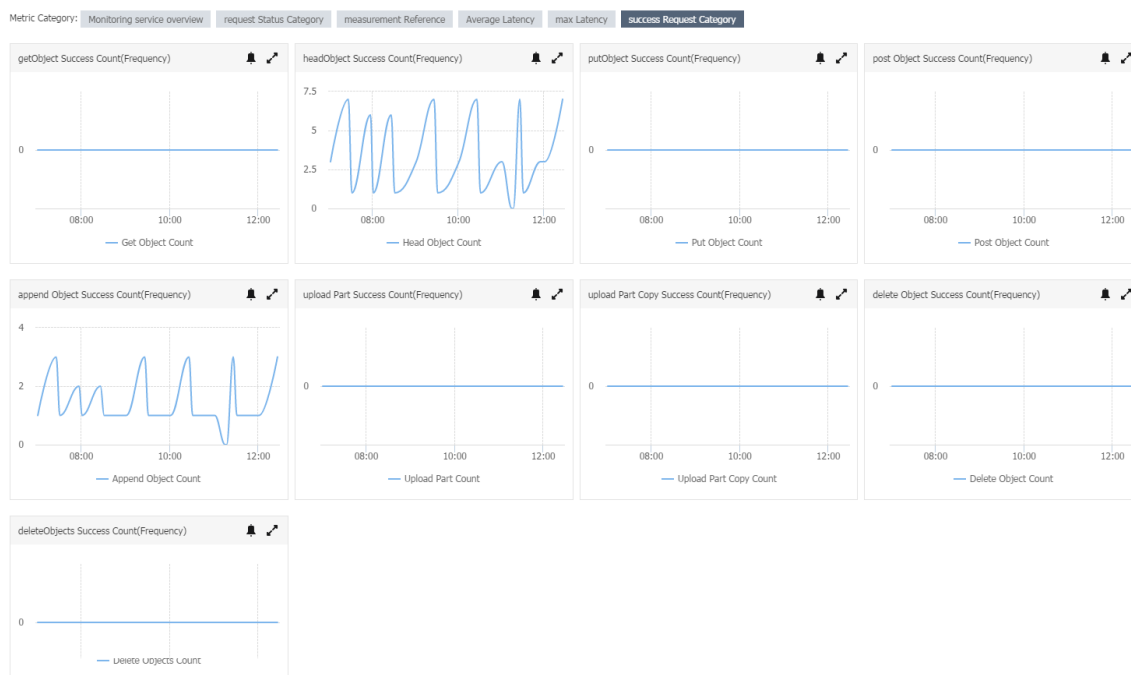


— Success request category

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

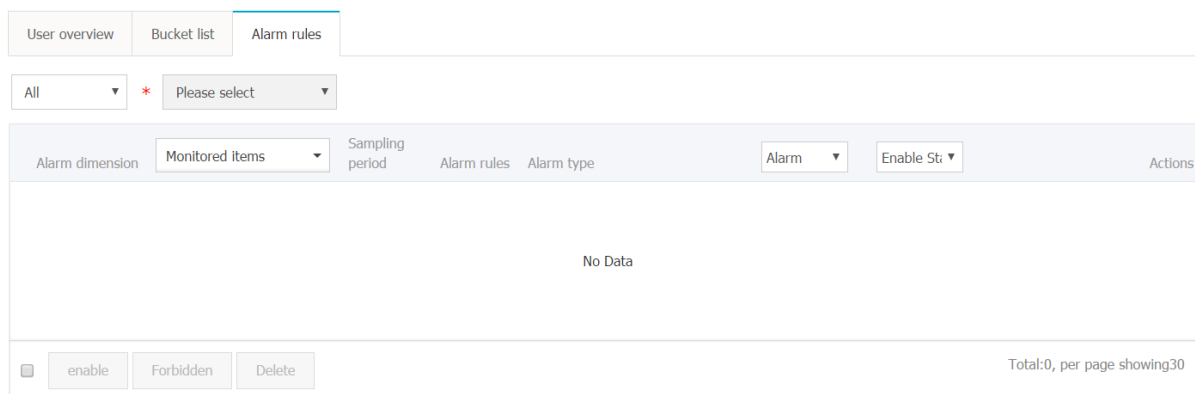
- getObject Success Count
- headObject Success Count
- putObject Success Count
- post Object Success Count
- append Object Success Count
- upload Part Success Count
- upload Part Copy Success Count
- delete Object Success Count
- deleteObjects Success Count

See the following figure:



Alarm rules

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



For the description and usage of the "Alarm Rules" tab page, see the [Alarm Service User Guide](#).

Additional links

For more information regarding the important points and user guide of the monitoring service, see the related chapter in [Monitoring, diagnosis, and troubleshooting](#).

12.3 Alarm service user guide

To help familiarize yourself with the basic concepts and configurations of alarm contacts and alarm contact groups, we recommend that the following documents are read before this user guide:

- [Alarm service overview](#)
- [Manage alarm contact](#)

Additionally, OSS alarm rules are developed in accordance with OSS metric items. This means they are categorized by dimensions similar to those of OSS metric items. Two alarm dimensions are available: user-level and bucket-level.

Alarm rule page

The alarm rule page is where you can view, modify, activate, deactivate, and delete alarm rules related to OSS monitoring alarms. You can also view historical alarms of the different alarm rules. An example screenshot is as follows:

Alarm dimension	Monitored items	Sampling period	Alarm rules	Alarm type	Alarm	Enable St	Actions
☐ User level	User internet send	5 minutes	Monitoring value it alarms1times continuously,> 9byte	aobeitest2... View	OK	Yes	Alarm history Modify Suspend Delete
☒ User level	User success count	5 minutes	Monitoring value it alarms1times continuously,> 15yunjankong.metric.unitName.frequency	aobeitest2... View	OK	Yes	Alarm history Modify Suspend Delete
☐ User level	User total request count	5 minutes	Monitoring value it alarms1times continuously,> 10yunjankong.metric.unitName.frequency	aobeitest2... View	OK	Yes	Alarm history Modify Suspend Delete
☐ User level	User valid request count	5 minutes	Monitoring value it alarms1times continuously,> 10yunjankong.metric.unitName.frequency	aobeitest2... View	OK	Yes	Alarm history Modify Suspend Delete

enable Forbidden Delete

Total:4, per page showing 30

- Click **Modify** next to the expected alarm rule to modify it.
- Click **Delete** next to the expected alarm rule to delete it. You can also select multiple alarm rules and then click **Delete** at the bottom of the table to delete alarm rules in batches.
- If an alarm rule is in the **Enable** status, click **Suspend** next to the expected alarm rule to deactivate it. Once the alarm rule is suspended, you no longer receive alarm information for this rule. You can also select multiple alarm rules and then click **Forbidden** at the bottom of the table to deactivate alarm rules in batches.
- If an alarm rule is in the **Forbidden** status, click **Enable** next to the expected alarm rule to activate it. The rule is then be resumed to detect exceptions and send alarm information. You can also select multiple alarm rules and then click **Enable** at the bottom of the table to activate alarm rules in batches.
- Click **Alarm history** next to the expected alarm rule to view information on past alarms corresponding to this rule.

History

Select a time range:

2016-11-22

-

2016-11-22

Search

Time	Trigger status	field	Alarm values	Alarm methods
2016-11-22 13:43:49	Alarm		26	Silent channel
2016-11-22 13:28:49	Alarm		18	Alarm groups:aobeitest222,jiangmi,test1,

Total:2, per page showing10

10

<

1

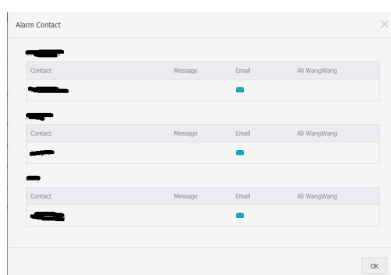
>

Close

Alarm history concepts

- Alarm history refers to past status changes of a selected alarm rule. Operations such as switching from normal status to alarm status, or switching from alarm status to normal status , are considered status changes. Additionally, a status change called channel silence is also available.
- Channel silence** occurs when a triggered alarm has remained active for 24 hours and has not returned to a normal status. In this case, no new alarm notifications are sent for 24 hours.
- Historical alarm information is retained for one month, and can be queried at a maximum of three days' data at one time within this time period. Alarm information older than one month is automatically deleted, and cannot be queried.

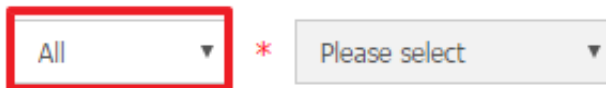
To view details about an alarm, such as the alarm contact list and contact details, click **View** next to the expected alarm. An example screenshot displaying specific details is as follows:



Search for alarm rules

Based on the control information at the bottom of the alarm rule page, you can quickly find alarm rules you have searched for:

- Alarm dimension drop-down box: All and Bucket Level. If you select All, all user-level and bucket-level alarm rules are displayed.



- Bucket drop-down box: If you select Bucket Level in the alarm dimension drop-down box, this box lists the buckets of the current user. Select a bucket to display all the alarm rules for this bucket:



- Monitored items drop-down box lists all OSS metric items, including user-level and bucket level metric items. If you select Monitored items, user-level and bucket-level alarm rules for all monitored items are displayed.
- Alarm status drop-down box lists alarm status, including OK and Alarm .
- Enable state drop-down box lists the enable status, including Enabled and Forbidden .
- View alarm rules

Click the **Alarm rules** tab to display all alarm rules. You can also select Bucket Level in the drop-down box and then select the name of the expected bucket to see alarm rules for that bucket. You can then filter returned information using selections in the drop-down box such as **Metric Item**, **Alarm status** and **Activation status**.

- View alarm rules for a specific bucket

If you want to view the alarm rules of a specific bucket, select Bucket Level in the alarm dimension drop-down box and then select the name of the target bucket in the bucket drop-down box. Select **Alarm Rules** for the target bucket in the **Bucket List** to go to the alarm tab. This tab displays all the alarm rules for this bucket. With the **Metric item**, **Alarm status** and **Activation status** drop-down boxes, you can better filter the alarm rules that match certain conditions in the current dimension.

- View alarm rules related to a specific metric item

Select a specific metric item in the metric item drop-down box to display all the alarm rules for this metric item.

- View alarm rules in a certain alarm status

Choose an alarm status in the alarm status drop-down box, such as Alarm, to display all the alarm rules currently in this status.

- View alarm rules in a certain activation status

Choose an activation status in the activation status drop-down box, such as Deactivated, to display all the alarm rules currently in this status.

Add alarm rules

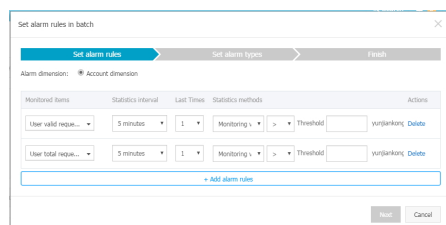
After specifying a bucket in the Bucket List Tab, click **Set Alarm Rule** to set an alarm rule.

Alternatively, click the alarm icon  in a metric chart in the User Overview tab or the Monitoring

View tab of a specific bucket to open the **Batch Set Alarm Rules** page to set multiple alarm rules.

The following example describes how to set alarms at the user-level. To learn more about the terms and concepts used later, see the [Alarm service overview](#) of CloudMonitor.

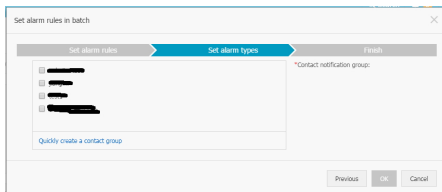
1. Set parameters for **Alarm rules** as follows:



- Alarm dimension specifies the monitoring dimension of the alarm rule to set. If the dimension is set to bucket-level, the expected bucket with which to set the alarm rule for must be specified.
- Monitored items specifies all the metric items for the selected alarm dimension. You can use the quick search box to easily find metric items:
- Statistics interval specifies the length of the interval between statistical measurements. The default setting is 5 minutes.
- Last times specifies the number of statistical cycles for which an alarm which is triggered when the value of the metric item continuously exceeds the threshold value in several consecutive statistical cycles.

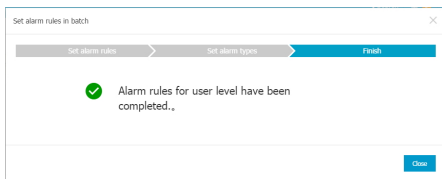
- Statistics method: specifies the statistical indicator calculated for this metric item. For the OSS monitoring service, the statistical method is set as `Monitoring Value`.
- Click **+ Add alarm rules** to set additional metric item alarm rules.
- Click **Delete** next to the expected alarm rule to delete it.

2. Click **Next**, the page to **Set the alarm types** is then displayed.



If you have set alarm contract groups following the [Manage alarm contact](#), they are displayed on the interface. If you have not set alarm contact groups, click **Quickly create a contact group** and follow the prompts to create a group.

3. Click **OK**.



- Add alarm rules in the Bucket list

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



Note:

During batch setting, the alarm dimension is bucket-level and the metric item must be a bucket-level metric item.

- Add alarm rules in a metric chart

In the **User overview** or Monitoring chart tab, for the expected bucket, click  in the upper-right corner of a metric chart to set alarm rules for the metric item associated with this metric chart.



Note:

If you click the alarm icon in a metric chart, the alarm dimension displayed on the alarms rule page is pre-determined and you can only set alarm rules for the metric item corresponding to the metric chart.

Considerations

Currently, alarm rules can be created without requiring prior association to a bucket. If you delete a bucket, any associated alarm rules are not deleted. Before deleting a bucket, we recommend that you delete any corresponding alarm rules first.

12.4 Metric item reference

This chapter provides parameter references to use with the API, or the CloudMonitor SDK, to access the metric data of the OSS monitoring service.

Project

The OSS monitoring service metric data uses the same project name: `acs_oss`.

Sample code written by the Java SDK:

```
QueryMetricRequest request = new QueryMetricRequest();  
Request.setproject ("acs_oss ");
```

StartTime and EndTime

The value range of the time parameters for CloudMonitor is in the format of [StartTime, EndTime]. The data that is attributed to StartTime is not collected, whereas the data that is attributed to EndTime can be accessed.

The CloudMonitor retention policy specifies that data is retained for 31 days. This means the interval between StartTime and EndTime cannot exceed 31 days, and data outside the 31 day collection period cannot be accessed.

For more information about other time parameters, see [CloudMonitor API Reference](#).

Sample code written by the Java SDK:

```
request.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

Metric	Metric item name	Unit	Level
UserCdnSend	User-level CDN outbound traffic	Byte	User level
UserCdnRecv	User-level CDN inbound traffic	Byte	User level
UserSyncSend	User-level outbound traffic of cross-region replication	Byte	User level
UserSyncRecv	User-level inbound traffic of cross-region replication	Byte	User level
UserServerErrorCount	User-level server-site error requests	Times	User level
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 time-out error request	Times	User level

Metric	Metric item name	Unit	Level
UserClientOtherErrorRate	User-level client-site time-out 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 request rate	%	User level
UserRedirectCount	User-level redirect requests	Times	User level
UserRedirectRate	User-level redirect request rate	%	User level
Availability	Availability	%	Bucket level
RequestValidRate	Valid request rate	%	Bucket level
TotalRequestCount	Requests	Times	Bucket level
ValidRequestCount	Valid requests	Times	Bucket level
InternetSend	Internet outbound traffic	Byte	Bucket level
InternetRecv	Internet inbound traffic	Byte	Bucket level
IntranetSend	Intranet outbound traffic	Byte	Bucket level
IntranetRecv	Intranet inbound traffic	Byte	Bucket level
CdnSend	CDN outbound traffic	Byte	Bucket level
CdnRecv	CDN inbound traffic	Byte	Bucket level
SyncSend	Outbound traffic of cross-region replication	Byte	Bucket level
SyncRecv	Inbound traffic of cross-region replication	Byte	Bucket level

Metric	Metric item name	Unit	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 resource not found	Times	Bucket level
ResourceNotFoundErrorRate	Client-site error request rate indicating resource not found	%	Bucket level
ClientTimeoutErrorCount	Client-site time-out error requests	Times	Bucket level
ClientOtherErrorRate	Client-site time-out error request rate	%	Bucket level
ClientOtherErrorCount	Other client-site error requests	Times	Bucket level
ClientOtherErrorRate	Other client-site error request rate	%	Bucket level
SuccessCount	Successful requests	Times	Bucket level
SuccessRate	Successful request rate	%	Bucket level
RedirectCount	Redirect requests	Times	Bucket level
RedirectRate	Redirect request rate	%	Bucket level
GetObjectE2eLatency	Average E2E latency of GetObject requests	Millisecond	Bucket level

Metric	Metric item name	Unit	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

Metric	Metric item name	Unit	Level
MaxPostObjectE2eLatency	Maximum E2E latency of PostObject requests	Millisecond	Bucket level
MaxPostObjectServerLatency	Maximum server latency of PostObject requests	Millisecond	Bucket level
AppendObjectE2eLatency	Average E2E latency of AppendObject requests	Millisecond	Bucket level
AppendObjectServerLatency	Average server latency of AppendObject requests	Millisecond	Bucket level
MaxAppendObjectE2eLatency	Maximum E2E latency of AppendObject requests	Millisecond	Bucket level
MaxAppendObjectServerLatency	Maximum server latency of AppendObject requests	Millisecond	Bucket level
UploadPartE2eLatency	Average E2E latency of UploadPart requests	Millisecond	Bucket level
UploadPartServerLatency	Average server latency of UploadPart requests	Millisecond	Bucket level
MaxUploadPartE2eLatency	Maximum E2E latency of UploadPart requests	Millisecond	Bucket level
MaxUploadPartServerLatency	Maximum server latency of UploadPart requests	Millisecond	Bucket level
UploadPartCopyE2eLatency	Average E2E latency of UploadPartCopy requests	Millisecond	Bucket level
UploadPartCopyServerLatency	Average server latency of UploadPartCopy requests	Millisecond	Bucket level

Metric	Metric item name	Unit	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

Metric	Metric item name	Unit	Level
			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

Metric	Metric item name	Unit	Level
			data belongs to the user level.

Sample code written by the Java SDK:

```
request.setMetric("UserAvailability");
```

12.5 Monitoring indicators reference

OSS indicators can be monitored at the user level or the bucket level based on application scenarios.

In addition to common chronological metric indicators, the system analyzes and collects statistics on the existing metric indicators for easy user observation of metric data and matching of billing policy. Statistical indicators over a specified period of time are provided, such as request status distribution and metering statistics of the month. This reference guide describes the indicators in detail.

All indicators are collected at the minute-level (per minute) except for metering and statistical indicators. Metering indicators are collected at the hour-level (per hour).

User-level indicators

The user level indicator refers to the indicator information that monitors the overall situation of the OSS system used from the user's account level, and is a summary of all bucket related monitoring data under the account. User-level indicators consist of three monitoring indicator details: current-month metering statistics, service monitoring overview, and request state details.

Service monitoring overview

Indicators in service monitoring overview are basic service indicators. Details of service monitoring overview indicators are as follows:

Indicator	Unit	Description
Availability	%	An indicator showing the system availability of using the storage service. It is obtained through the equation: $1 - \text{percentage of requests with server-end errors}$ (indicated by a return

Indicator	Unit	Description
		code 5xx) in all requests.
Valid requests rate	%	Percentage of valid requests in all requests. For more information about valid requests, see the following description.
Requests	Times	Total number of requests received and processed by the OSS server
Valid requests	Times	Total number of requests whose return code is 2xx or 3xx.
Internet outbound traffic	Byte	Downstream Internet traffic
Internet inbound traffic	Byte	Upstream Internet traffic
Intranet outbound traffic	Byte	Downstream intranet traffic of the service system
Intranet inbound traffic	Byte	Upstream intranet traffic of the service system
CDN outbound traffic	Byte	Downstream CDN traffic when CDN acceleration service is activated, that is, the origin retrieval traffic
CDN inbound traffic	Byte	Upstream CDN traffic when CDN acceleration service is activated
Outbound traffic of cross-region replication	Byte	Downstream traffic generated in the data replication process when the cross-region 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

In addition to the above specific monitoring indicators, statistics are also provided for the distribution of request status over a period of time. The statistics are mainly based on the status codes of

the returned status codes or OSS error codes (the total number and the percentage of requests within the observed time period).

Request state details

Request state details indicators are requested monitoring information based on the return status code, or OSS error code, associated with the different requests. Details of request state details indicators are as follows:

Indicator	Unit	Description
Server-site error requests	Times	Total number of requests with system-level errors indicated by a return code 5xx
Server-site error requests rate	%	Percentage of requests with server-end errors in all requests
Network error requests	Times	Total number of requests whose <i>HTTP status code is 499</i>
Network error requests rate	%	Percentage of requests with network errors in all requests
Client-end authorization error requests	Times	Total number of requests with a return code 403
Client-end authorization error requests rate	%	Percentage of requests with client-end authorization errors in all requests
Client-end error requests indicating resource not found	Times	Total number of requests with a return code 404
Client-end error requests rate indicating resource not found	%	Percentage of requests with client-end errors indicating resource not found in all requests
Client-end time-out error requests	Times	Total number of requests whose return status code is 408 or return OSS error code is RequestTimeout
Client-end time-out error requests rate	%	Percentage of requests with client-end time-out errors in all requests

Indicator	Unit	Description
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

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.

Indicator	Unit	Description
Get requests	Times	Total number of Get requests of the user from 00:00 of the first day of the current month to the metering statistic collection deadline.

Bucket-level indicators

Bucket-level indicators are used to monitor OSS operations of specific buckets and are applicable for business scenarios. In addition to current-month metering statistics and basic service indicator items such as service monitoring overview and request state details (which can be monitored at the account level), bucket-level indicators include metering indicators and performance indicators such as metering reference, latency, and successful request operation categories.

Service monitoring overview

Similar to the user-level description, the service monitoring overview indicators are basic indicators, but use metric data that is displayed at the bucket-level.

Request state details

Similar to the user-level description, the request state details indicators use metric data that is displayed at the bucket-level.

Current-month metering statistics

Statistical methods are similar to those listed in current-month metering statistics at the user level , but the former collects resource usage statistics at the bucket level. Details of current-month metering statistics at the bucket-level are as follows:

Indicator	Unit	Description
Storage size	Byte	Size of storage occupied by a specified bucket before the metering statistic collection deadline.
Internet outbound traffic	Byte	Total Internet outbound traffic of a specified bucket from 00:00 of the first day of the current month to the metering statistic collection deadline.
Put requests	Times	Total number of Put requests of a specified bucket from 00:

Indicator	Unit	Description
		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.
Get requests	Times	Total number of Gut requests of a specified bucket in an hour.

Latency

Latency Request latency directly reflects the system performance and is monitored using two indicators: average latency and maximum latency. The indicators are collected and aggregated at the minute-level. Moreover, indicators can be classified based on the OSS API request operation type to more specifically reflect the performance of the system responding to different operations. Only APIs involving data operations in bucket-related operations (excluding meta operations) are monitored currently.

Besides, in order to facilitate analyzing performance hotspots and environmental problems, latency monitoring indicators are collected from two different links of E2E and the server, in which:

- E2E latency refers to the E2E latency of sending a successful request to OSS, including the processing time OSS requires to read the request, send a response, and receive a response confirmation message.
- Server latency is the latency of OSS processing a successful request, excluding the network delay involved in E2E latency.

Note that performance indicators are used to monitor successful requests (with a return status code 2xx).

The following table lists specific metric indicator items:

Indicator	Unit	Description
Average E2E latency of GetObject requests	Millisecond	Average E2E latency of successful requests whose request API is GetObject
Average server latency of GetObject requests	Millisecond	Average server latency of successful requests whose request API is GetObject
Maximum E2E latency of GetObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is GetObject
Maximum server latency of GetObject requests	Millisecond	Maximum server latency of successful requests whose request API is GetObject
Average E2E latency of HeadObject requests	Millisecond	Average E2E latency of successful requests whose request API is HeadObject
Average server latency of HeadObject requests	Millisecond	Average server latency of successful requests whose request API is HeadObject
Maximum E2E latency of HeadObject requests	Millisecond	Maximum E2E latency of successful requests whose request API is HeadObject
Maximum server latency of HeadObject requests	Millisecond	Maximum server latency of successful requests whose request API is HeadObject
Average E2E latency of PutObject requests	Millisecond	Average E2E latency of successful requests whose request API is PutObject

Indicator	Unit	Description
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

Indicator	Unit	Description
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 UploadPartCopy requests	Millisecond	Maximum server latency of successful requests whose request API is UploadPartCopy

Successful request operation categories

In conjunction with latency monitoring, the monitoring of successful requests reflects the system capability of processing access requests to a certain extent. Similarly, only APIs involving data operations in bucket-related operations are monitored currently. The following lists specific indicator items:

Indicator	Unit	Description
Successful GetObject requests	Times	Number of successful requests whose request API is GetObject
Successful HeadObject requests	Times	Number of successful requests whose request API is HeadObject

Indicator	Unit	Description
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

12.6 Service monitoring, diagnosis, and troubleshooting

Despite reducing users' costs of infrastructure construction and O&M cloud applications compared to traditional applications, cloud applications have complicated monitoring, diagnosis, and troubleshooting. The OSS storage service provides a wide array of monitoring and log information , helping you fully understand program behavior and promptly discover and locate problems.

Overview

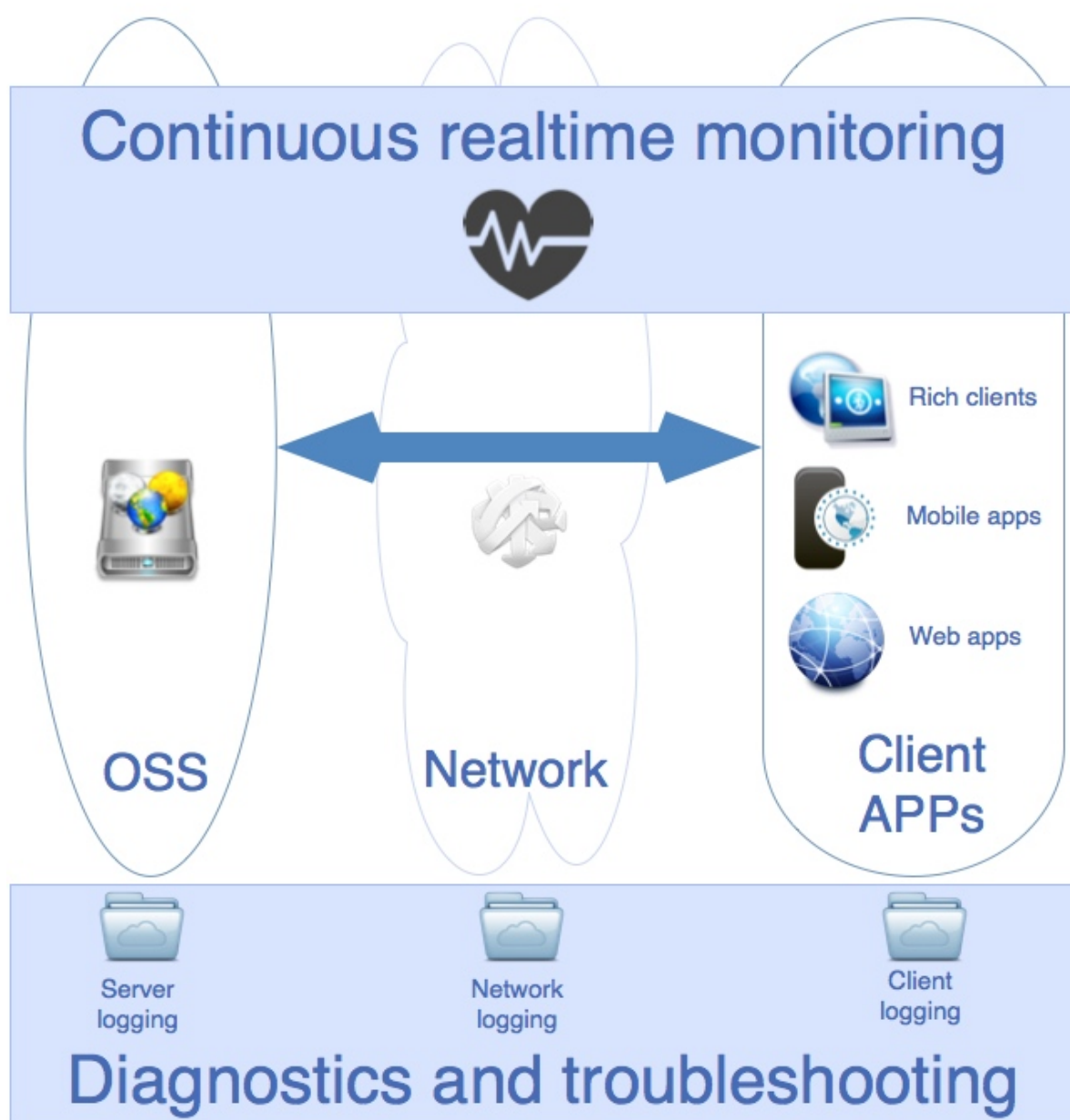
This chapter instructs you how to monitor, diagnose, and troubleshoot OSS problems by using the OSS monitoring service, logging, and other third-party tools, helping you achieve the following goals:

- Monitors in real time the running status and performance of OSS and provides prompt alarm notifications.
- Provides effective methods and tools to help you locate problems.

- Provides methods to help you quickly solve common OSS-related problems.

This chapter is organized as follows:

- **OSS real-time monitoring**: Describes how to use the OSS monitoring service to continuously monitor the running status and performance of OSS.
- **Tracking and diagnosis**: Describes how to use the OSS monitoring service and logging function to diagnose problems, and how to associate the relevant information in log files for tracking and diagnosis.
- **Troubleshooting**: Describes typical problems and corresponding troubleshooting methods.
-

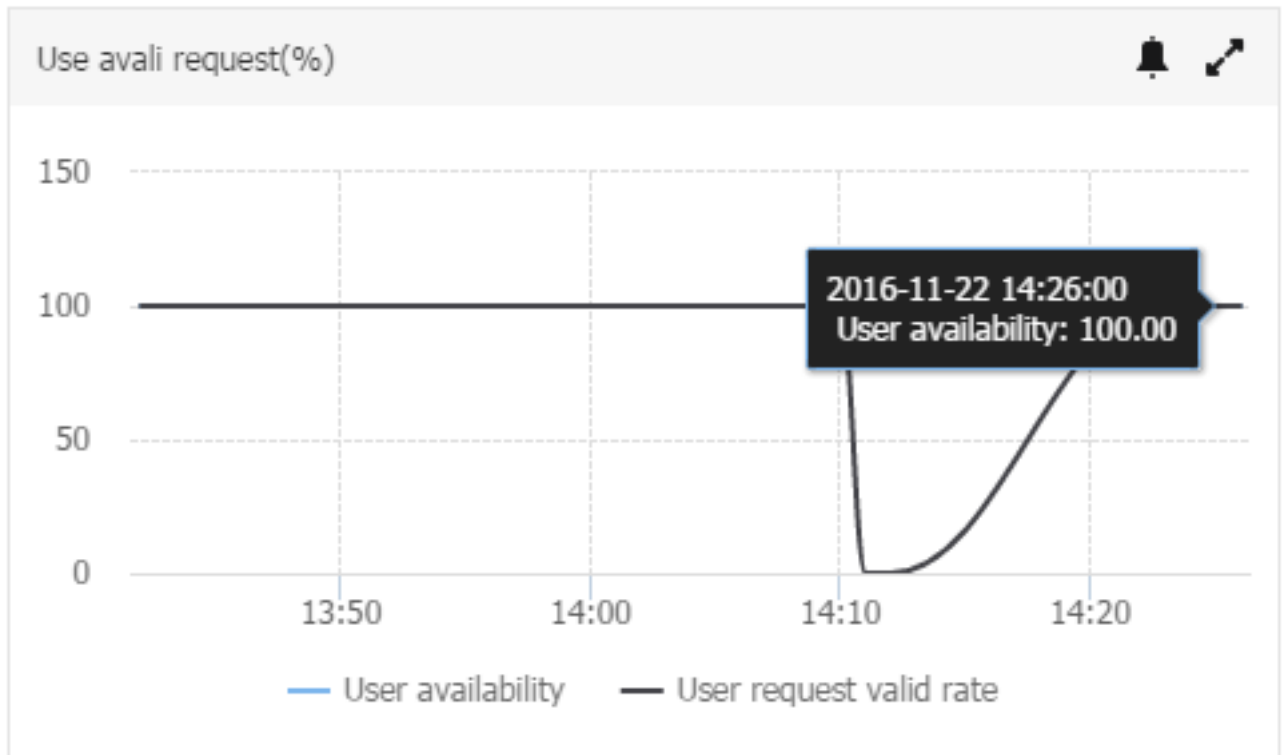


OSS monitoring

Overall operating conditions

- Availability and percentage of valid requests

This is an important indicator related to system stability and the ability of users to correctly use the system. Any value lower than 100% indicates that some requests have failed.



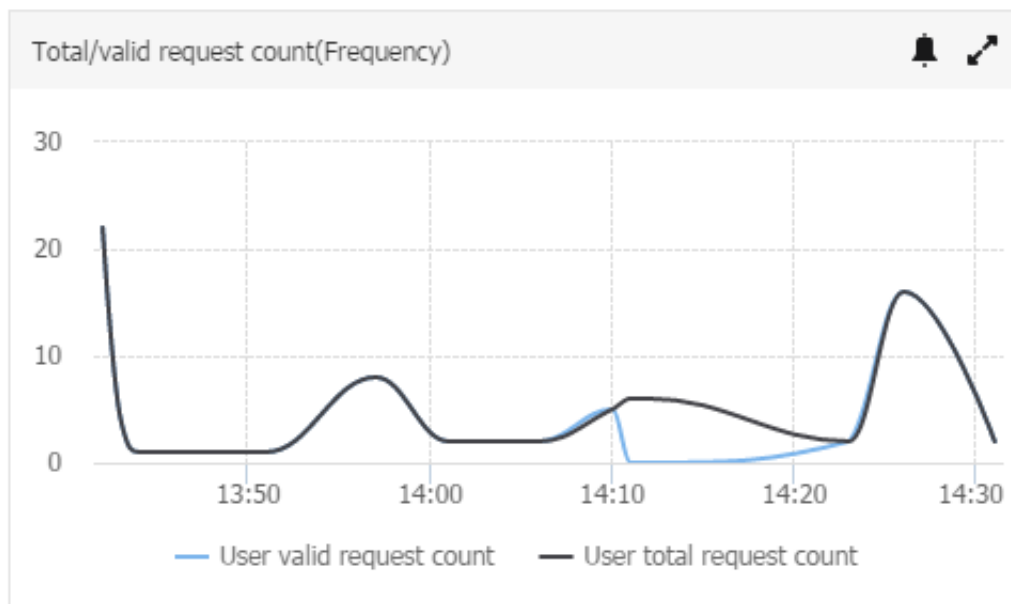
Availability may also temporarily fall below 100% due to system optimization factors, such as partition migration for load balancing. In these cases, OSS SDKs can provide relevant retry mechanisms to handle this type of intermittent failure, keeping the service end unaware.

Also, when the percentage of valid requests falls below 100%, you must analyze the issue based on your own usage. You can use request distribution statistics or request status details to determine the actual types of request errors. Then, you can use [Tracking and Diagnosis](#) to determine the cause and perform [Troubleshooting](#). In some business scenarios, a valid request rate is expected to fall below 100%. For example, you may need to first check that an object exists and then perform a certain operation based on the existence of the object. In this case, if the object does not exist, the read request that checks its existence returns a 404 error code (resource does not exist error). This inevitably produces a valid request rate of less than 100%.

For businesses that require high system availability, you can set an alarm rule that is triggered when the indicator falls below the expected threshold value.

- Total No. of requests and No. of valid requests

This indicator reflects the system operation status from the perspective of the total traffic volume. When the No. of valid requests is not equal to the total No. of requests, this indicates that some requests have failed.



You can watch the fluctuations in the total No. of requests and No. of valid requests, especially when they sharply increase or decrease. In such cases, follow-up action is required. You can set alarm rules to make sure you receive prompt notifications. For periodic businesses, you can set periodic alarm rules (periodic alarms will be available soon). For more information, see [Alarm Service User Guide](#).

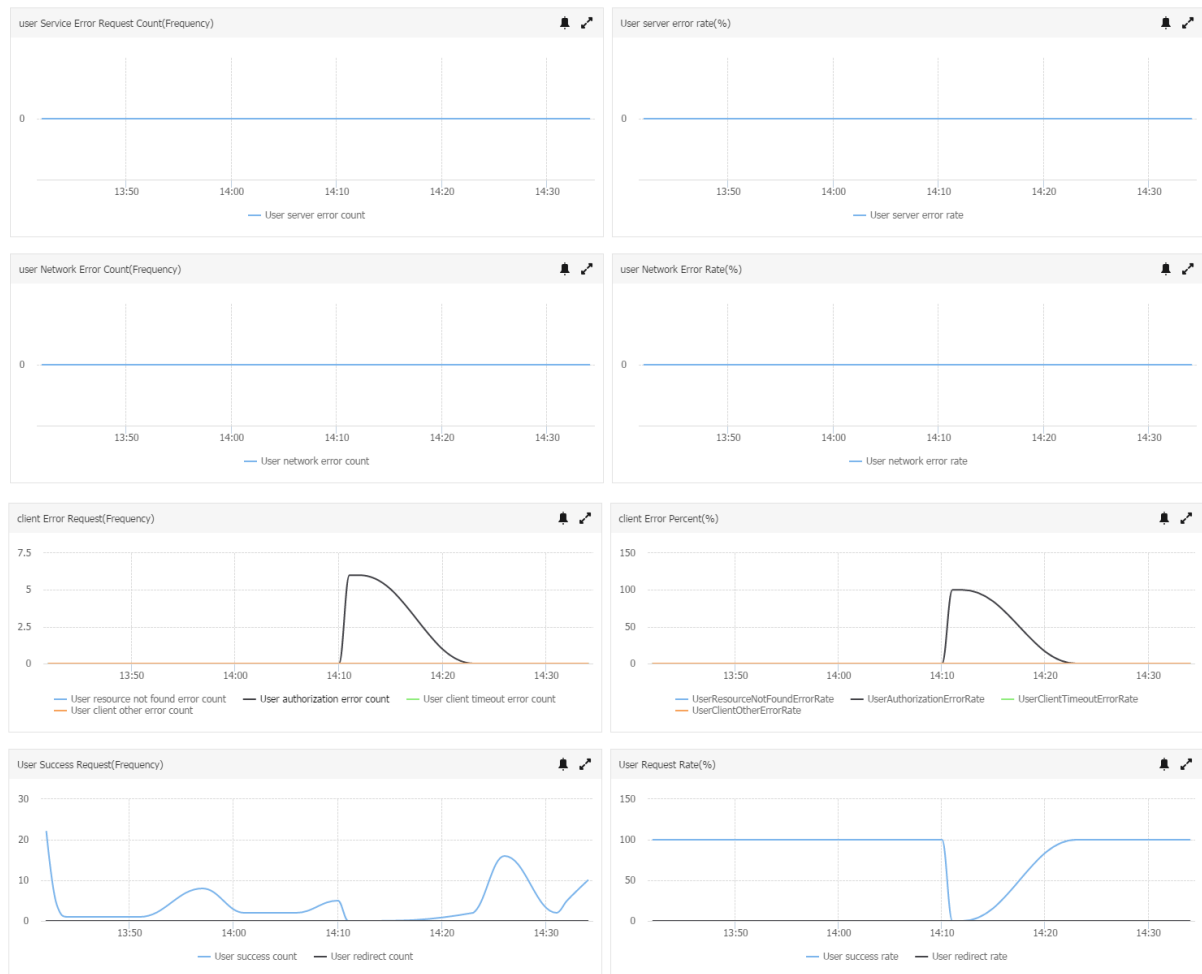
- Request status distribution statistics

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

User level request		
Metric	Sum value	Percent
User authorization error count	12yunjiankong.metric.unitName.frequency	14.29%
User success count	72yunjiankong.metric.unitName.frequency	85.71%
Sum	84yunjiankong.metric.unitName.frequency	100%

Request status details monitoring

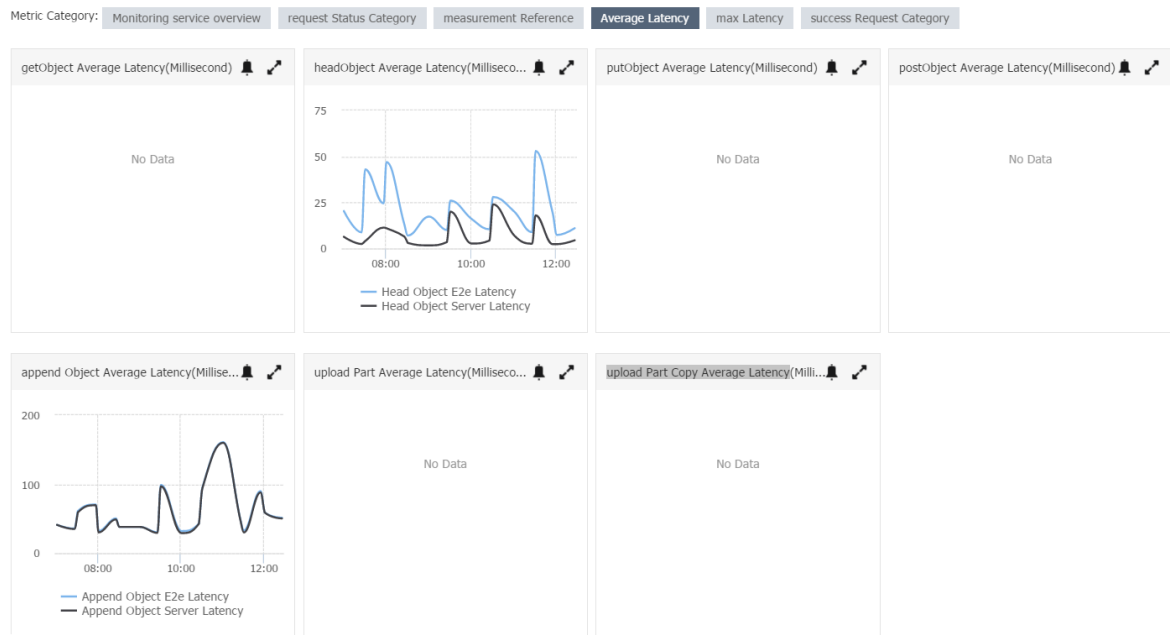
Request status details provides more details about the request monitoring status on the basis of request status distribution statistics. They let you monitor certain types of requests in more detail.



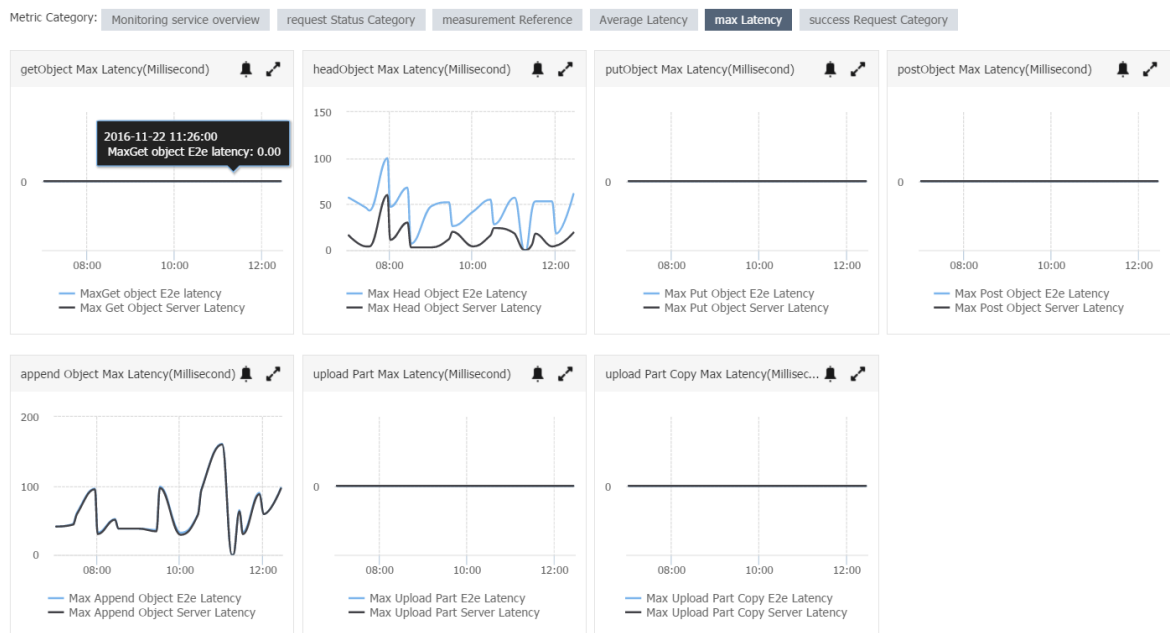
Performance monitoring

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

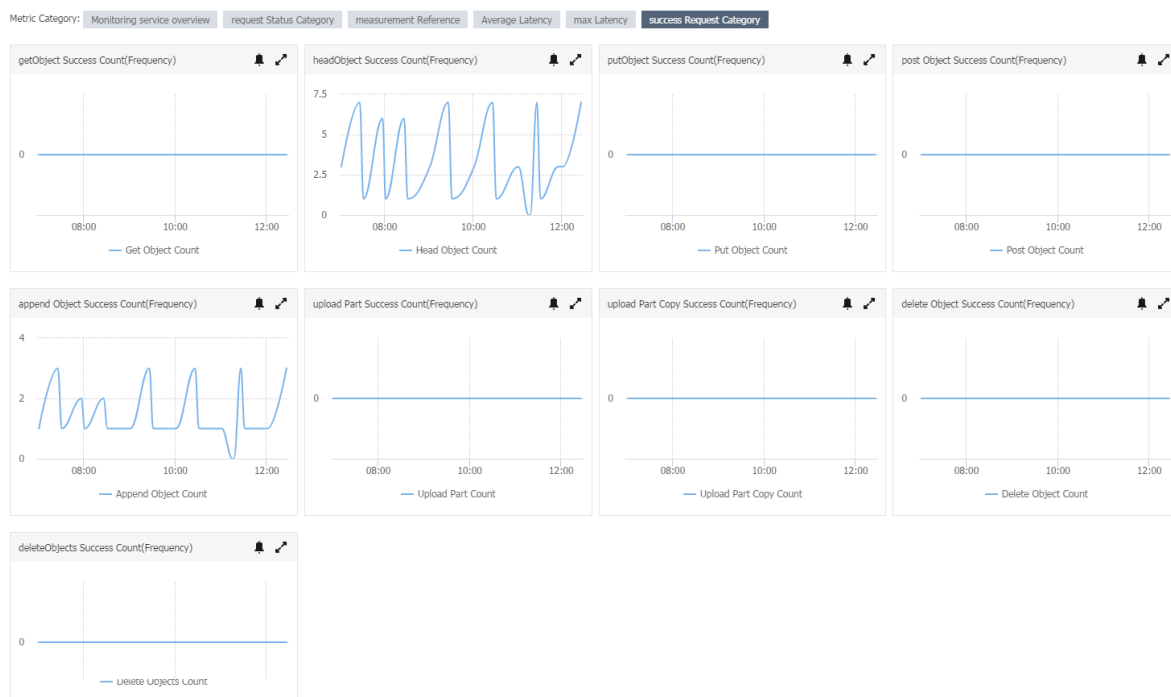
- Average latency: E2E average latency and Server average latency



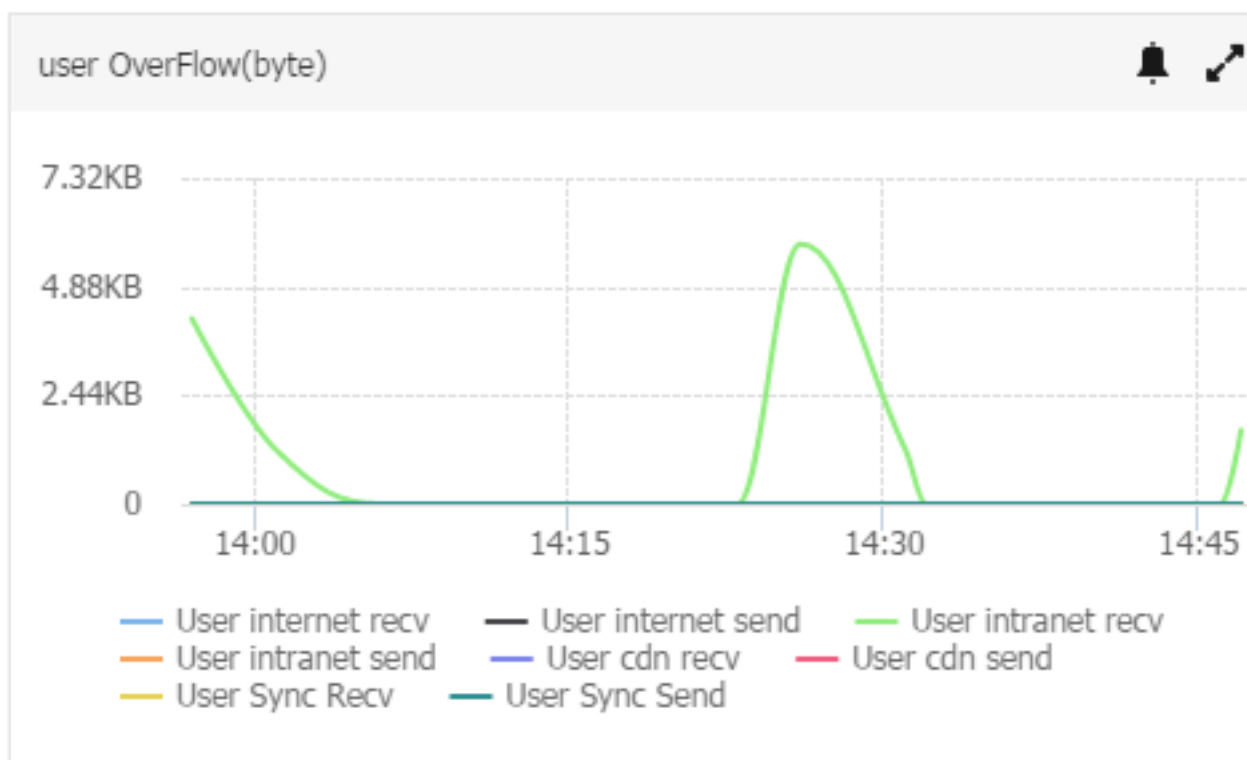
- Maximum latency: E2E maximum latency and Server maximum latency



- Successful request categories



- Traffic



The preceding metric items (except for 'Traffic') implement categorized monitoring based on API operation types:

- GetObject
- HeadObject
- PutObject

- PostObject
- AppendObject
- UploadPart
- UploadPartCopy

The latency indicators show the average or maximum time needed for API operation types to process requests. E2E latency is the indicator for end-to-end latency. Besides the time needed to process requests, it also includes the time needed to read requests and send responses, and the delay caused by network transmission. Server latency only includes the time needed to process the requests on the server, not the client-side transmission network latency. Therefore, if the E2E latency suddenly increases but the server latency does not change significantly, you can determine that the poor performance has been caused by network instability, instead of an OSS system fault.

In addition to the APIs mentioned previously, "successful request operation categories" also monitors the quantity of requests for the following two API operation types:

- DeleteObject
- Deleteobjects

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

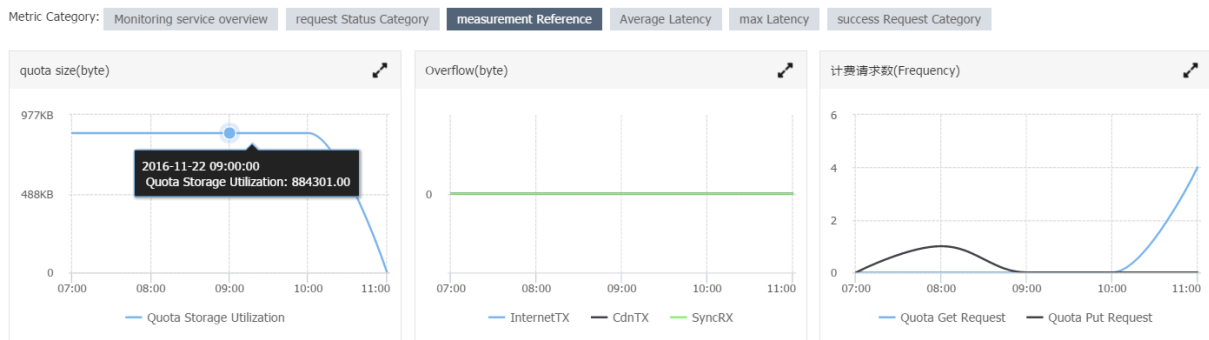
For performance-type indicators, we must focus on sudden and abnormal changes, such as when the average latency suddenly spikes or remains above the normal request latency baseline for a long period of time. You can set alarm rules that correspond to performance indicators, so that the relevant personnel are immediately notified if an indicator falls below or exceeds a threshold value. For businesses with periodic peaks and troughs, you can set periodic alarm rules for week on week, day on day, or hour on hour comparisons (periodic alarms will be available soon).

Billing monitoring

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

The OSS monitoring service collects bucket-level billing monitoring data on an hourly basis. In the monitoring view for a specific bucket, you can see graphs of continuous monitoring trends. Using

the monitoring view, you can analyze your businesses' OSS resource usage trends and estimate future costs. See the following figure:



The OSS monitoring service also provides statistics on the quantity of user and bucket-level resources consumed each month. For example, the total amount of OSS resources consumed by an account or bucket starting from the 1st day of the month. These statistics are updated hourly. This increases your understanding of your resource usage and computation fees for the current month in real time, as shown in the following figure:

as shown in the following figure:



Note:

In the monitoring service, the provided billing data is pushed to the maximum extent possible, but this may cause some discrepancies with the actual bill amount. Please note that the Billing Center data is used in actual billing applications.

Tracking and diagnosis

Problem diagnosis

- Performance diagnosis

Many subjective factors are involved in the determination of application performance. You must use the satisfaction of your business needs in your specific business scenario as a baseline, to determine if a performance problem occurs. Also, when a client initiates a request, factors that may cause performance problems may come from anywhere in the request chain. For example, problems may be caused by OSS overloads, client TCP configuration problems, or traffic bottlenecks in the basic network architecture.

Therefore, when diagnosing performance problems, you must first set a reasonable baseline. Then, you use the performance indicators provided by the monitoring service to determine

the potential root cause of any performance problem. Next, you find detailed information in the relevant logs to help you further diagnose and troubleshoot any faults.

In the [Troubleshooting](#) section, we give examples of many common performance problems and troubleshooting measures. This can be used as a reference.

- Error diagnosis

When requests from client applications are at fault, the clients receive error information from the server. The monitoring service records these errors and shows statistics for the various types of errors that may affect requests. You can also retrieve detailed information for individual requests from the server log, client log, and network log. Generally, the returned HTTP status code, OSS error code, and OSS error information can indicate the cause of the request failure.

For error response information details, see [OSS error responses](#).

- Using the logging function

OSS provides a server logging function for user requests. This helps you track end-to-end detailed request logs.

For instructions on the activation and use of the logging function, see [Set logging](#).

For more information on Log Service naming rules and record formats, see [Set access logging](#).

- Using network logging tools

In many situations, you can diagnose problems by using the logging function to record storage log and client application log data. However, in certain situations, you may need more details by using network logging tools.

This is because capturing traffic exchanged between clients and the server can give you more detailed information on the data exchanged between clients and server and the underlying network conditions, which can help you investigate problems. For example, in some situations, user requests may report an error, but no request can be seen in the server log. In such cases, you can use the records logged by the OSS logging function to see if the cause of the problem lies with the client, or you can use network monitoring tools to check for a network problem.

[Wireshark](#) is one of the most common network log analysis tools. This free protocol analyzer runs on the packet level and provides a view of detailed packet information for various network protocols. This can help you troubleshoot packet loss and connection problems.

see [Wireshark User Guide](#).

E2E tracking and diagnosis

Requests are initiated by a client application process and pass through the network environment to the OSS server, where they are processed. Then, a response is sent by the server over the network environment and received by the client. This is an end-to-end tracking process.

Associating client application logs, network tracking logs, and server logs provides detailed information for you to troubleshoot the root cause of a problem and discover potential problems.

In OSS, the provided RequestIDs serve as identifiers used to associate the information from various logs. In addition, the log timestamps not only allow you to quickly query specific log time ranges, but can also show you the time points when request events and other client application, network, and service system events occurred during this period. This helps you analyze and investigate problems.

- RequestID

Whenever the OSS receives a request, it allocates it a unique server request ID, its RequestID.

In different logs, the RequestID is located in different fields:

- In server logs recorded by the OSS logging function, the RequestID is located in the “Request ID” column.
- In the process of network tracking (for example, when using Wireshark to capture data streams), the RequestID is the x-oss-request-id header value in the response message.
- In client applications, you must use the client code to manually print the RequestID in the client log. At the present 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 `OSSEException`'s `getRequestId` method to obtain this information.

- Timestamps

You can use timestamps to find relevant log entries. You must note that there may be some deviations between the client time and server time. On a client, you can use timestamps to search for server log entries recorded by the logging function. For this, you must add or subtract 15 minutes.

Troubleshooting

Common performance-related problems

- High average E2E latency, with low average server latency

We have already discussed the differences between average E2E latency and average server latency. Therefore, we can say that high E2E latency and low server latency are caused by two possible reasons:

- Slow client application response speed
- Network factors

A slow client application response speed can be caused by several possible reasons:

- Limited number of available connections or threads
 - Use the relevant command to check if the system has a large number of connections in the `TIME_WAIT` status. If yes, adjust the core parameters to solve this problem.
 - When the number of available threads is limited, first check for bottlenecks affecting the client CPU, memory, network, or other resources. If no bottleneck is found, increase the number of concurrent threads properly.
 - If the problem persists, you have to optimize the client code. For example, you can use an asynchronous access method. You can also use the performance analysis function to analyze client application hotspots, and then perform the necessary optimization.
- Insufficient resources, such as CPU, memory, or bandwidth
 - For this type of problem, you must first use the relevant system monitoring function to find client resource bottlenecks. Then, optimize the client code to rationalize resource usage or increase the client resources (increase the number of cores or the memory).

Investigate network latency problems

Generally, high E2E latency due to network factors is temporary. You can use Wireshark to investigate temporary and persistent network problems, such as packet loss problems.

- Low average E2E latency, low average server latency, but high client request latency

When the client experiences high request latency, the most probable cause is that the requests are not reaching the server. Therefore, we must find out why the client requests are not arriving at the server.

Two client-side factors can cause high client request sending latency:

- A limited number of available connections or threads: see the solution described in the preceding section.

- Client requests are retried multiple times: In this situation, you must find and solve the cause of the request retries based on the retry information. You can follow these steps to determine if the client has a retry problem:
 - Check the client log. The detailed log entries indicate if retries have occurred. Using the OSS Java SDK as an example, you can search for the following warn or info-level log entries. If such entries are found in the log, this indicates that requests have been retried.

```
[Server]Unable to execute HTTP request:
Or
[Client]Unable to execute HTTP request:
```

- If the client log level is debug, search for the following log entries (again we are using the OSS Java SDK as an example). If such entries exist, this indicates requests have been retried.

```
Retrying on
```

If no problem with the client occurs, you must check for potential network problems, such as packet loss. You can use a tool such as Wireshark to investigate network problems.

- High average server latency

If the server latency during downloads or uploads is high, this may be caused by the following two factors:

- A large number of clients are frequently accessing the same small object.

In this situation, you can view the server log recorded by the logging function to determine if a small object or a group of small objects are being frequently accessed in a short period of time.

For download scenarios, we suggest you activate the CDN service for this bucket, to improve performance. This also reduces your traffic fees. In the case of upload, you may consider revoking write permissions for this object (bucket), if this does not affect your business.

- Internal system factors

For internal system problems or problems that cannot be solved through optimization, please provide our system staff with the RequestIDs in your client logs or in the logs recorded by the logging function, and they can help you solve the problem.

Server errors

When the number of server-side errors increases, two scenarios must to be considered:

- Temporary increase

For this type of problem, you must adjust the retry policy in the client program and adopt a reasonable concession mechanism, such as exponential backoff. This not only avoids temporary service unavailability due to system optimization, upgrades, and other such operations (such as partition migration for system load balancing), but also avoids high pressure during business peaks.

- Permanent increase

When the number of server-side errors sustainably increases, please provide our back-end staff with the RequestIDs in your client logs or in the logs recorded by the logging function, and they can help you find the problem.

Network errors

Network errors occur when the server is processing a request and the connection is lost (not due to a server-side issue), so the HTTP request header cannot be returned. In such a situation, the system records an [HTTP Status Code of 499](#) for this request. In the following situations, the server may change the request status code to 499:

- Before processing a received read/write request, if the server detects that the connection is unavailable, the request is recorded as 499.
- When the server is processing a request and the client preemptively closes the connection, the request is recorded as 499.

In summary, a network error occurs during the request process when a client independently closes the request or the client is disconnected from the network. If the client independently closes requests, you can check the client code, to identify the cause and time of the client's disconnection from OSS. When the client loses its network connection, you can use a tool such as Wireshark to investigate network connection problems.

Client errors

- Increase in client authorization errors

If you detect an increase in client authorization errors or the client receives a large number of 403 request errors, this is most commonly caused by the following problems:

- The bucket domain name accessed by the user is incorrect.

- If the user uses a third-level or second-level domain name to access a bucket, this may cause a 403 error if the bucket is not in the region indicated by the domain name. For example, if you have created a bucket in the Hangzhou region, but a user attempts to access it using the domain name `Bucket.oss-cn-shanghai.aliyuncs.com`. In this case, you must confirm the bucket's region and then correct the domain name information.
- If you have activated the CDN acceleration service, this problem may occur when CDN binds an incorrect origin retrieval domain name. In this case, check that the CDN origin retrieval domain name is the bucket's third-level domain name.
- If you encounter 403 errors when using JavaScript clients, this may be caused by a problem in the CORS (Cross-Origin Resource Sharing) settings, because web browsers implement “same source policy” security restrictions. In this case, you must check the bucket's CORS settings and correct any errors. For information about CORS settings, see [CORS](#).
- Access control problems can be divided into four types:
 - When you use a primary AK for access, you must check the AK settings for errors if the AK is invalid.
 - When you use a RAM sub-account for access, you must check that the sub-account is using the correct sub-account AK and that the sub-account has the relevant permissions.
 - When you use temporary STS tokens for access, you must confirm that the temporary token has not expired. If the token has expired, apply for a new one.
 - If you use bucket or object settings for access control, you must check that the bucket or object to be accessed supports the relevant operations.
- When you authorize third-party downloads (using signed URLs to access OSS resources), if access was previously normal and then suddenly reports a 403 error, it is likely that the URL has expired.
- When RAM sub-accounts use OSS utilities, this may also produce 403 errors. These utilities include `ossftp`, `ossbrowser`, and the OSS console client. When you enter the relevant AK information during logon and the system throws an error, if you entered the correct AK, you must check that the AK is a sub-account AK and that this sub-account has permission for `GetService` and other operations.
- Increase in client-side ‘resource does not exist’ errors

When the client receives a 404 error, this means that you are attempting to access a resource or information that does not exist. When the monitoring service detects an increase in 'resource does not exist' errors, this is most likely caused by one of the following problems:

- Service usage: For example, when you first need to check that an object exists before performing another operation and you call the `doesObjectExist` method (using the Java SDK as an example), if the object does not exist, the client receives the value "false". However, the server actually produces a 404 request error. Therefore, in this business scenario, 404 errors are normal.
- The client or another process previously deleted this object. You can confirm this problem by searching for the relevant object operation in the server log recorded by the logging function.
- Network faults case packet loss and retries. For example, the client may initiate a delete operation to delete a certain object. The request reaches the server and successfully executes the delete operation. However, if the response packet is lost during transmission on the network, the client initiates a retry. This second request then produces a 404 error. You can confirm that network problems are producing 404 errors using the client log and server log:
 - Check for retry requests in the client application log.
 - Check if the server log shows two delete operations for this object and that the first delete operation has an HTTP status of 2xx.
- Low valid request rate and high number of other client-side request errors

The valid request rate is the number of requests that return an HTTP status code of 2xx/3xx as a percentage of total requests. Status codes of 4XX or 5XX indicate a failed request and reduce the valid request rate. Other client-side request errors indicate requests errors other than the following: server errors (5xx), network errors (499), client authorization errors (403), resource does not exist errors (404), and client time-out errors (408 or OSS error code: `RequestTimeout 400`).

Check the server log recorded by the logging function to determine the specific errors encountered by these requests. You can see [OSS error responses](#) to find a list of common error codes returned by OSS. Then, check the client code to find and solve the specific cause of these errors.

Abnormal increase in storage capacity

If storage capacity increases abnormally without a corresponding increase in upload requests, this is generally caused by a delete problem. In such a case, check for the following two factors:

- When the client application uses a specific process to regularly delete storage objects to free up space: The investigation processes for this request are as follows:
 1. Check if the valid request rate has decreased, because a failed delete request may cause storage objects to fail to be deleted as expected.
 2. Find the specific cause for the decrease in the valid request rate by looking at the error types of the requests. Then, you can combine the specific client logs to see the detailed error information (for example, the STS temporary token used to free up storage space may have expired).
- When the client sets a LifeCycle to delete storage objects: Use the console or an API to check that the current bucket LifeCycle value is the same as before. If not, modify the configuration and use the server log recorded by the logging function to find information on the previous modification of this value. If the LifeCycle is normal but inactive, contact an OSS system administrator to help identify the problem.

Other OSS problems

If the Troubleshooting section did not cover your problem, use one of the following methods to diagnose and troubleshoot the problem.

1. View the OSS monitoring service, to see if there have been any changes compared to the expected baseline behavior. Using the monitoring view, you may be able to determine if this problem is temporary or permanent and which storage operations are affected.
2. The monitoring information can help you search the server log data recorded by the logging function, to find information on any errors that may have occurred when the problem started. This information may be able to help you find and solve the problem.
3. If the information in the server log is insufficient, use the client log to investigate the client application, or use a network tool such as Wireshark to check your network for problems.

13 Cloud data processing

Image Processing

For introduction and more information about functions, see [Image Processing](#).

Media Processing

Media Processing is a transcoding computing service for multimedia data. It provides an economic , easy-to-use, elastic, and highly scalable method for conversion of audio and video stored on OSS into formats suitable for playing on PCs, TVs, or mobile devices.

Media Processing was constructed based on Alibaba Cloud computing services. In the past, users had to make a high investment to purchase, build, and manage transcoding software and hardware, and perform complex configuration optimization, transcoding parameter adaptation, and other operations. Media Processing has transformed everything. It has enhanced the elasticity of cloud computing services. Media Processing offers transcoding capabilities to fulfill business transcoding demands to its extreme and also curbs the wastage of resources.

Media Processing functions include the Web management console, service APIs, and SDKs. Users can use and manage Media Processing and integrate transcoding functions into their own apps and services.

Media Processing function list

- Transcoding
- Pipelines
- Screenshot
- Media information
- Watermark
- Preset templates
- Custom templates
- Video clip output
- Resolution scaling
- M3U8 custom segment length output
- Audio/Video extraction
- Video image rotation
- Video-to-GIF conversion

For introduction and more information about functions, see [Media Processing documentation](#).