

Alibaba Cloud Object Storage Service

Best Practices

Issue: 20180930

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 Access control.....	1
1.1 Overview.....	1
1.2 What is RAM and STS.....	1
1.3 Access a bucket without using the primary account.....	4
1.4 Read/Write permission separation.....	5
1.5 Bucket permission separation.....	6
1.6 STS temporary access authorization.....	8
1.7 FAQs about subaccount settings.....	22
2 Data security.....	24
2.1 Check data transmission integrity by using 64-bit CRC.....	24
2.2 Protect data through client encryption.....	26
3 OSS resource monitoring and alarm service.....	31
4 OSS performance and scalability best practice.....	33

1 Access control

1.1 Overview

Alibaba Cloud's permission management mechanism includes Resource Access Management (RAM) and Security Token Service (STS). This enables users to access OSS through subaccounts with different permissions and grants users temporary access authorization. Usage of RAM and STS can greatly improve management flexibility and security.

The following content is introduced in permission management:

- [What is RAM and STS](#)
- [Access a bucket without using the primary account](#)
- [Read/Write permission separation](#)
- [Bucket permission separation](#)
- [Access control](#)
- [STS temporary access authorization](#)
- [The problem of OSS authority and Its Troubleshooting](#)
- [STS frequently asked questions and troubleshooting](#)
- [OSS sub-account setup Frequently Asked Questions](#)

Click [RAM Policy Editor](#) Online Editing allows you to generate authorization policies.

1.2 What is RAM and STS

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

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

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

Why RAM and STS?

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

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

**Note:**

Therefore, AccessKey of subaccounts must not be disclosed.

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

These functions are further illustrated with the help of examples.

Basic concepts

The following are some explanations of the basic concepts:

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

**Note:**

Subaccounts can assume roles. When a role is assumed, the permissions granted for a subaccount are the permissions of the role.

- **Policy:** Policies are rules used to define permissions; for example, they permit users to read or write certain resources.
- **Resource:** Resources are the cloud resources that users can access like all OSS buckets, a certain OSS bucket, or a certain object in a specific OSS bucket.

A subaccount and roles have the same relationship to each other as you and your identities. At work, you may be an employee, while at home you may be a father. In different scenarios, you may assume different roles. Different roles are assigned corresponding permissions. The concept of “employee” or “father” is not an actual entity that can be the subject of actions. These concepts

are only complete when an individual assumes them. This illustrates an important concept: a role may be assumed by multiple people at the same time.

**Note:**

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

The following example provides better understanding of the concept:

- Assume that Alice is the the Alibaba Cloud user and she has two private OSS buckets, `alice_a` and `alice_b`. Alice has full permission for both buckets.
- To avoid leaking her Alibaba Cloud account AccessKey, which would pose a major security risk, Alice uses RAM to create two subaccounts, Bob and Carol. Bob has read/write permission for `alice_a` and Carol has read/write permission for `alice_b`. Bob and Carol both have their own AccessKeys. This way, if one is leaked, only the corresponding bucket is affected and Alice can easily cancel the leaked user permissions on the console.
- Now, for some reason, Alice must authorize another person to read the objects in `alice_a`. In this situation, she must not only disclose Bob's AccessKey. Rather, she can create a new role like `AliceAReader`, and grant this role the read permission for `alice_a`. However, note that, at this time, `AliceAReader` cannot be used because no AccessKey corresponds to this role. `AliceAReader` is currently only a virtual entity with the permission to access `alice_a`.
- To obtain temporary authorization, Alice can call the STS's AssumeRole interface to notify STS that Bob wants to assume the `AliceAReader` role. If successful, STS returns a temporary AccessKeyId, AccessKeySecret, and SecurityToken, which serve as the access credentials. When these credentials are given to a temporary account, the user obtains temporary permission to access `alice_a`. The credentials' expiration time is specified when the AssumeRole interface is called.

Why are RAM and STS so complex?

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

Subaccounts and roles are separated to separate the entity that executes operations from the virtual entity that represents a permissions set. If a user requires many permissions including the read and write permissions but each operation only requires part of the total permission set, you can create two roles, one with the read permission and the other with the write permission. Then create a user who does not have any permission but can assume these two roles. When the user needs to read or write data, the user can temporarily assume the role with the read permission

or the role with the write permission. This reduces the risk of permission leaks for each operation. Additionally, roles can be used to grant permissions to other Alibaba Cloud users, making the collaboration easier.

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

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

Test tool

During testing, we use `osscmd`, a tool in the OSS PythonSDK that allows you to directly work on OSS through the command line. `osscmd` can be obtained from [PythonSDK](#).

Typical `osscmd` usage:

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

1.3 Access a bucket without using the primary account

Assume that the user is a mobile developer and currently only has one bucket, `ram-test-dev`, for development, testing, and other functions. The user must stop using the primary account to access this bucket. This can avoid problems caused by AccessKey and password leaks. In the following example, replace AccessKey with your own AccessKey. The procedure is as follows:

1. On the console, select **Products and Services > Resource Access Management**.



Note:

The service must be activated first if you have never used it before.

2. Click **Users** to go to the **User Management** page.

3. The page shows that no user is created. Click **New User** on the upper right corner to create a subaccount with the same OSS access permissions as the primary account. Remember to select the **Auto generate AccessKey for this user**.
4. The AccessKey for this account is generated and must be saved for later use.
5. Return to **User Management** interface, which shows the newly created account named `ram_test`. When created, this subaccount does not have any permissions yet. Click the **Authorize** link on the right side and grant this subaccount full access permissions for OSS.

After authorization, click the **Management** link on the right side if you want to give the subaccount console logon or other permissions.

Now we can test the uploading and downloading operations. In the example, the AccessKey is `ram_test`'s AccessKey. During the test, replace this with your own AccessKey.

```
$./osscmd get
oss://ram-test-dev/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.
com -i o0hue*****Frogv -k OmVwFJO3qcT0*****FhOYpg3p0KnA
100% The object test.txt is downloaded to test.txt, please check.
0.069(s) elapsed
```

```
$./osscmd put test.txt oss://ram-test-dev/test.txt --host=oss-cn
-hangzhou.aliyuncs.com -i o0hue*****Frogv -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
100%
Object URL is: http://ram-test-dev.oss-cn-hangzhou.aliyuncs.com/test.
txt
Object abstract path is: oss://ram-test-dev/test.txt
ETag is "E27172376D49FC609E7F46995E1F808F"
0.108(s) elapsed
```

As you can see, this subaccount can basically be used for all operations, so you can avoid leaking the primary account's AccessKey.

1.4 Read/Write permission separation

When the users want to use an application server to provide external service, OSS can store back-end static resources. In this case, we recommend that the application server be granted the OSS read-only permission to reduce the risk of attacks. The read and write permission separation can be configured to grant the application server a user with the read-only permission.

1. Create an account `ram_test_pub`. As shown in the following figure, select `ReadOnly` in the authorization management area:

2. You can now use the AccessKey of the subaccount to test the upload and download permissions. The AccessKey here is a ram_test_pub AccessKey and is to be replaced with your own AccessKey during the test.

```

$./osscli get oss://ram-test-dev/test.txt test.txt --host=oss-cn-hangzhou.aliyuncs.com -i oOhue*****Frogv -k OmVwFJO3qcT0*****FhOYpg3p0KnA
100% The object test.txt is downloaded to test.txt, please check.
0.070(s) elapsed

```

```

$. /Osscli put test.txt OSS: // Ram-test-dev/test.txt -- Host = porteroohue ***** frogv-K OmVwFJO3qcT0 * FhOYpg3p0KnA?
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '5646E49C1790CF0F531BAE0D'), ('date', 'Sat, 14 Nov 2015 07:37:00 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>5646E49C1790CF0F531BAE0D</RequestId>
  <HostId>ram-test-dev.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
put Failed!

```

With reference to the preceding example, we can conclude that the ram_test_pub account cannot be used to upload files.

1.5 Bucket permission separation

Another scenario is introduced in this section. If another user is using the developed app, you can use an individual bucket to store your app data. Assume that the bucket is the ram-test-app. In consideration of permission separation, the application server must not be allowed to access the ram-test-app; that is, the account ram_test_pub is permitted only to read ram-test-dev. This can also be realized through the RAM permission system. The procedure is as follows:

1. Because the system has no default bucket-level policy, we must create a custom policy.

```

{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListObjects",
        "oss:GetObject"
      ],
      "Resource": [
        "acs:oss:*:*:ram-test-dev",

```

```

    "acs:oss:*:*:ram-test-dev/*"
  ]
}
}
}

```

After setting, we can see the policy in the custom authorization policy list.

2. In user authorization management, add this policy to the selected authorization policy list. Also in **Users > Management > Authorization policy**, all previously granted OSS read permissions can be revoked.

3. Test the validity of permission configured.

- The object in ram-test-dev can be accessed:

```

$./osscli get oss://ram-test-dev/test.txt test.txt --host=oss-cn-
hangzhou.aliyuncs.com -i oOhue*****Frogy -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
100% The object test.txt is downloaded to test.txt, please check.
0.047(s) elapsed

```

- The object in ram-test-app cannot be accessed:

```

$./osscli get oss://ram-test-app/test.txt test.txt --host=oss-cn-
hangzhou.aliyuncs.com -i oOhue*****Frogy -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection',
'keep-alive'), ('x-oss-request-id', '5646ED53F9EEA2F3324191A2'),
('date', 'Sat, 14 Nov 2015 08:14:11 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>5646ED53F9EEA2F3324191A2</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
get Failed!

```

- Files cannot be uploaded to oss-test-app:

```

$./osscli put test.txt oss://ram-test-app/test.txt --host=oss-cn-
hangzhou.aliyuncs.com -i oOhue*****Frogy -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection',
'keep-alive'), ('x-oss-request-id', '5646ED7BB8DE437A912DC7A8'),
('date', 'Sat, 14 Nov 2015 08:14:51 GMT'), ('content-type', 'application/xml')]
Error Body:
<? XML version = "1.0" encoding = "UTF-8" ? >
<Error>
  <Code>AccessDenied</Code>

```

```
<Message>AccessDenied</Message>
<RequestId>5646ED7BB8DE437A912DC7A8</RequestId>
<HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error status:
403
put Failed!
```

Using the preceding configuration, we have successfully separated the permissions for ram-test-dev and ram-test-app.

The preceding section explains how to use the subaccount permission control function to separate permissions and minimize the potential risk of information leakage.

1.6 STS temporary access authorization

In the previous documents, we used only the RAM user functions. These user accounts are for long-term normal use. This poses as a serious risk if the RAM user permissions cannot be promptly revoked in case of information leakage.

In the previous example, assume that our developer's app allows users to upload data to the OSS bucket am-test-app and currently, the number of app users is large. In this case, how can the app securely grant data upload permissions to many users and how can it be certain of storage isolation among multiple users?

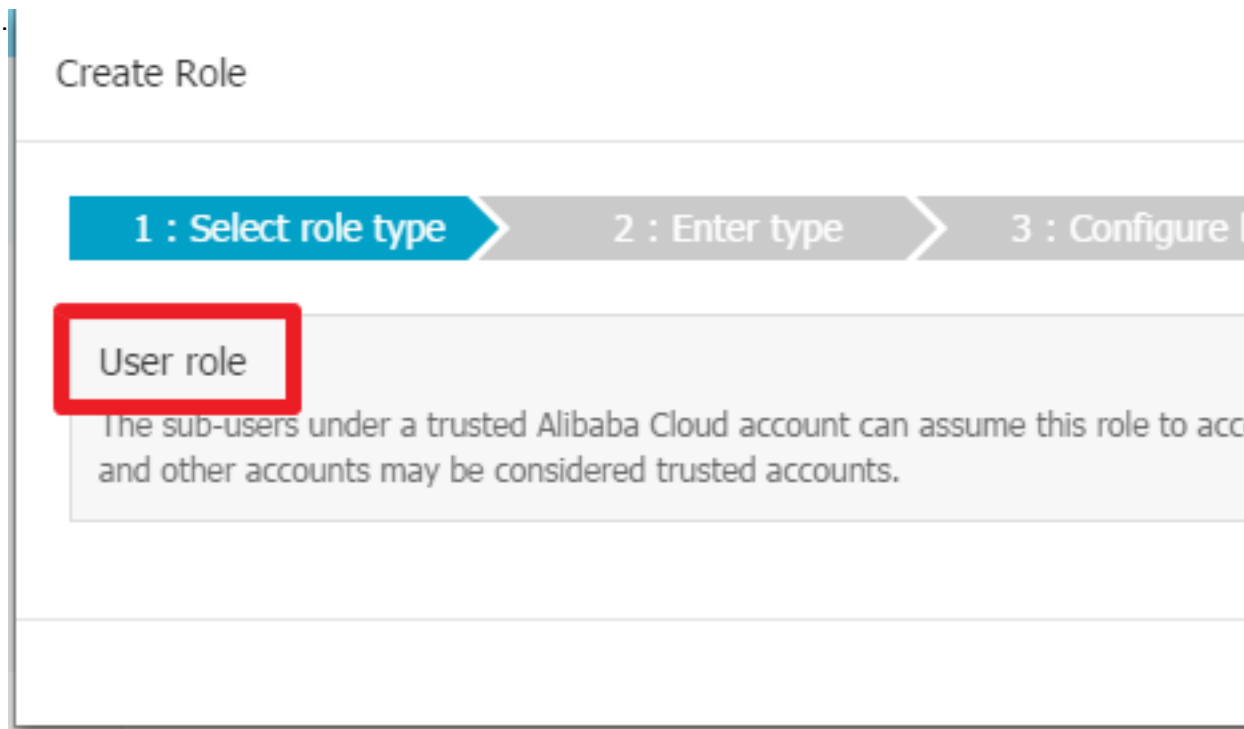
In such scenarios, we need to grant users temporary access using STS. STS can be used to specify a complex policy that restricts specified users by only granting them the minimum necessary permissions.

Create a role

Based on the example in the previous document, the app user has a bucket, ram-test-app, to store personal data. A role can be created as follows:

1. Create a RAM user account named ram_test_app using the process illustrated in the previous documents. Do not grant this account any permissions, because it inherits the permissions of a role which it assumes.
2. Create roles. Here you must create two roles for users to perform read operations and to upload files respectively.
 - Log on to the RAM console and select **Roles > New Role**.

- Select a role type. Here you must select **User** role.



- Enter the role type information. Because this role has been used by its own Alibaba Cloud account. Use the default setting.

- Configure basic role information.

Create Role

1 : Select role type > 2 : Enter type > 3 : Configure permissions

* The role name :

The name must be 1-64 characters long and contain only letters, numbers, and "-"

Remarks :

3. When the role was created, it did not have any permissions. Therefore, we must create a custom authorization policy using the process described earlier. The following is the authorization policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "allow",
      "Action": [
        "oss:ListObjects",
        "Oss: GetObject"
      ],
      "Resource": [
        "acs:oss:*:*:ram-test-app",
        "acs:oss:*:*:ram-test-app/*"
      ]
    }
  ]
}
```

This indicates read-only permission for ram-test-app.

Create Authorization Policy

STEP 1: Select an authorization policy

STEP 2: Edit permissions and submit

* Authorization policy
name :

ram-test-app-readonly

The name must be 1-128 characters long and can contain numbers, and "-"

Remarks :

ram-test-app-readonly

Policy content :

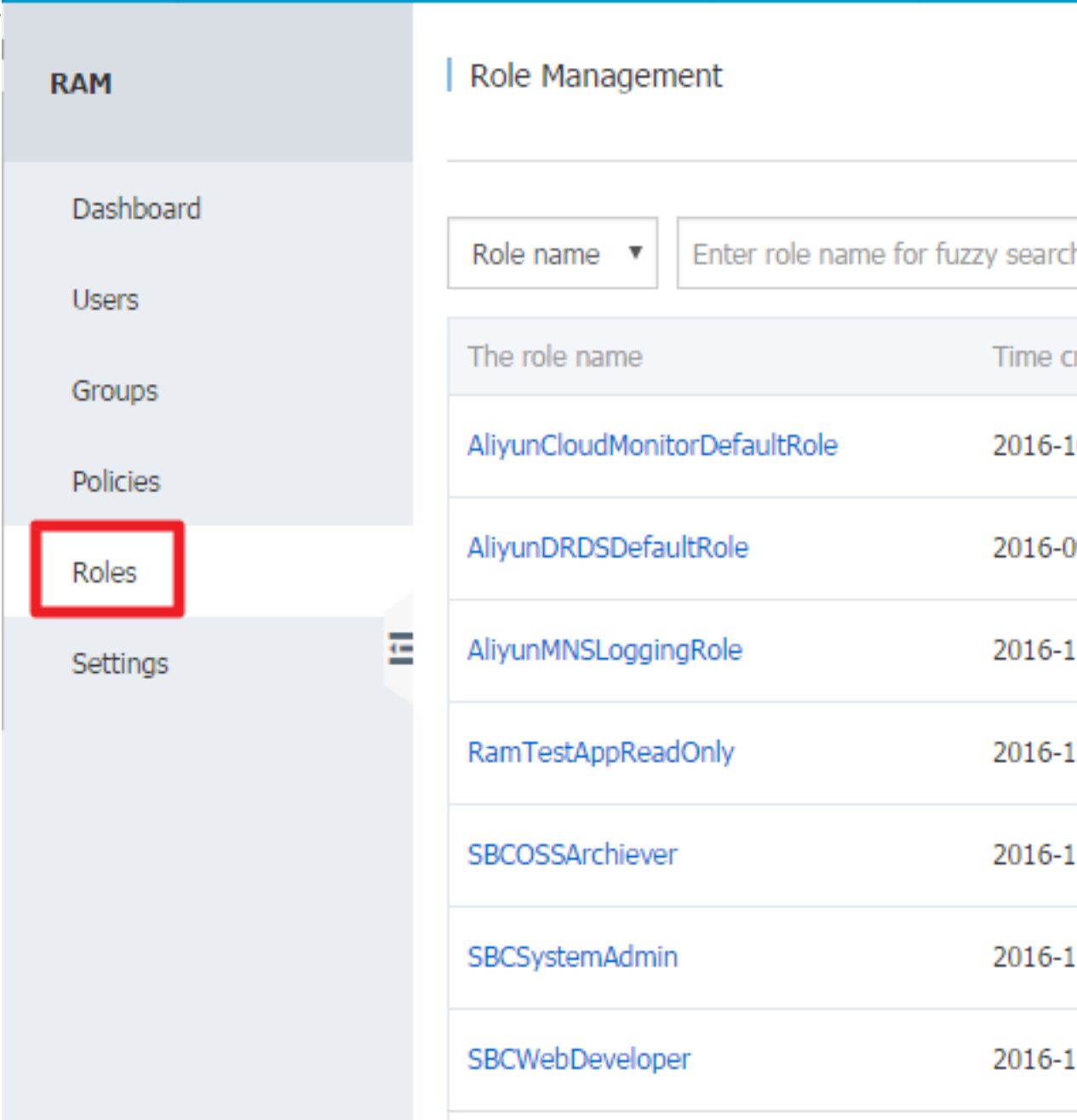
```
1 {  
2   "Version": "1",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": [  
7         "oss:ListObjects",  
8         "oss:GetObject"  
9       ],  
10      "Resource": [  
11        "acs:oss:*:*:ram-test-app",  
12        "acs:oss:*:*:ram-test-app/*"  
13      ]  
14    }  
15  ]  
16 }
```

[Authorization policy format definition](#)

[Authorization policy FAQs](#)

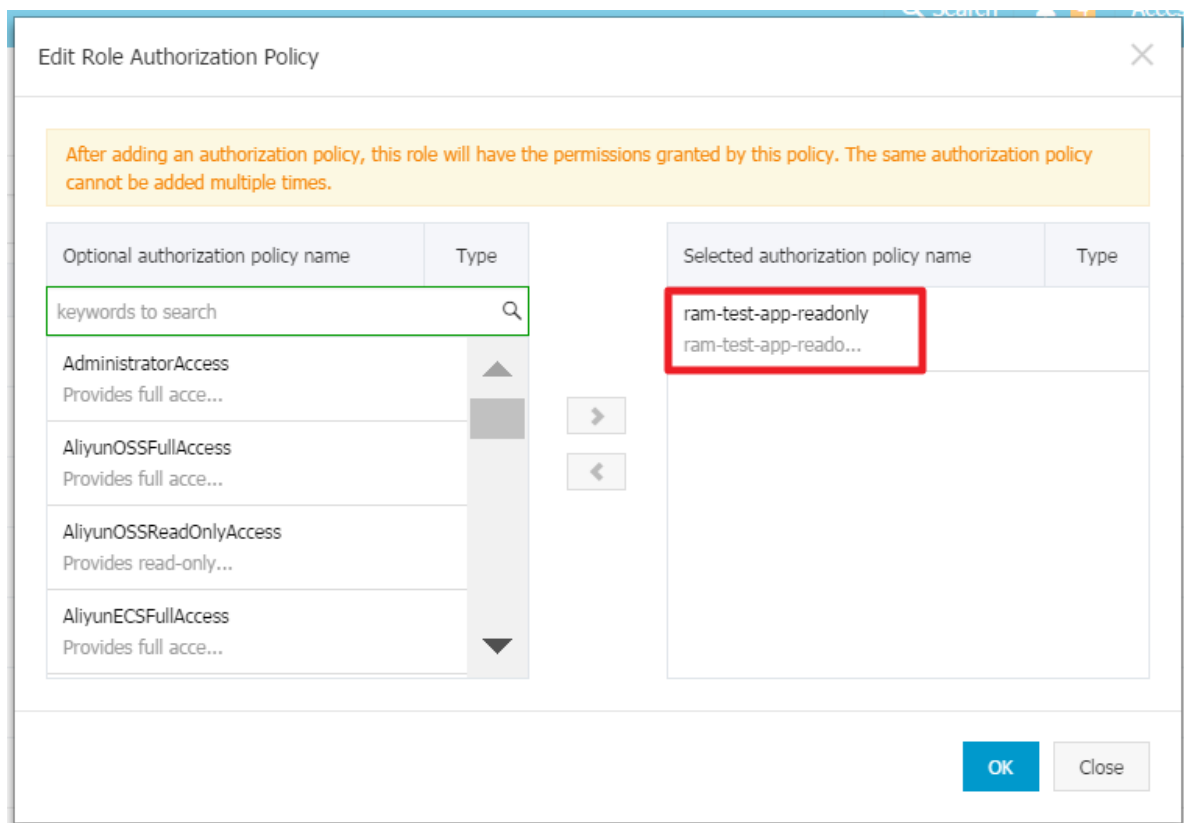
Prev

4. After the policy is established, give the role `RamTestAppReadOnly` the `ram-test-app` read-only permission on the role management page.



The screenshot shows the RAM Role Management interface. On the left, a sidebar menu under the 'RAM' header includes 'Dashboard', 'Users', 'Groups', 'Policies', 'Roles' (highlighted with a red box), and 'Settings'. The main panel is titled 'Role Management' and features a search bar with a 'Role name' dropdown and a text input field labeled 'Enter role name for fuzzy search'. Below the search bar is a table listing roles.

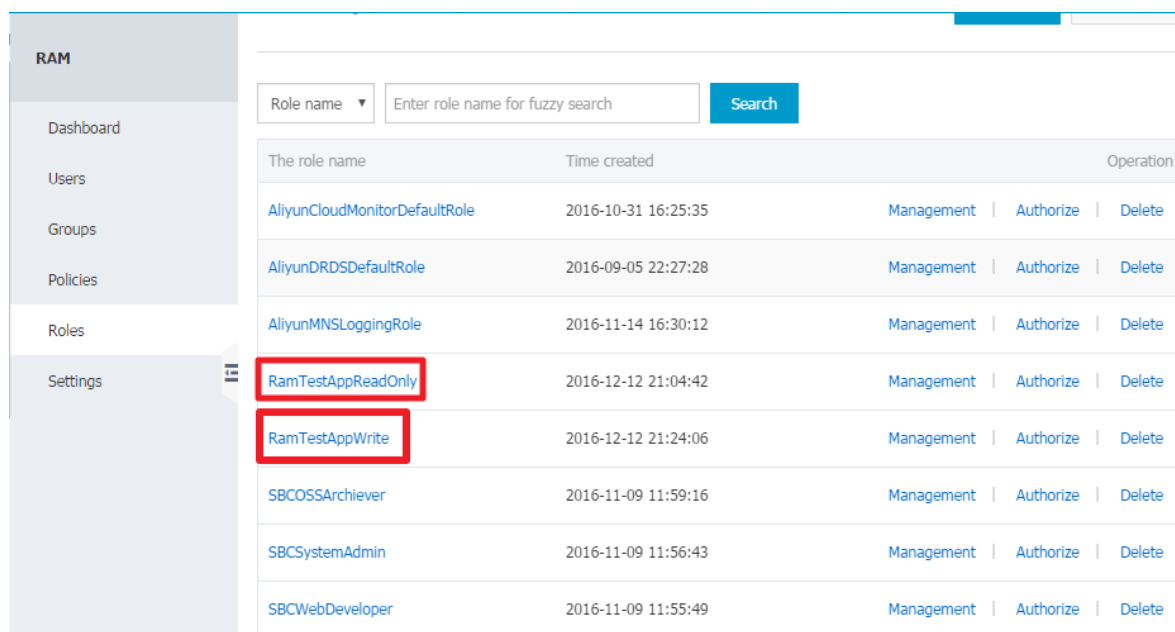
The role name	Time created
AliyunCloudMonitorDefaultRole	2016-1
AliyunDRDSDefaultRole	2016-0
AliyunMNSLoggingRole	2016-1
RamTestAppReadOnly	2016-1
SBCOSSArchiver	2016-1
SBCSystemAdmin	2016-1
SBCWebDeveloper	2016-1



5. Perform the same procedure to create the role RamTestAppWrite and use a custom authorization policy to grant ram-test-app write permission. The authorization policy is as follows:

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:DeleteObject",
        "oss:ListParts",
        "oss:AbortMultipartUpload",
        "oss:PutObject"
      ],
      "Resource": [
        "acs:oss:*:*:ram-test-app",
        "acs:oss:*:*:ram-test-app/*"
      ]
    }
  ]
}
```

Now we have created two roles, RamTestAppReadOnly and RamTestAppWrite, with read-only and write permissions for ram-test-app, respectively.



RAM			
Dashboard	Role name ▾ <input type="text" value="Enter role name for fuzzy search"/>		<input type="button" value="Search"/>
Users			
Groups			
Policies			
Roles			
Settings			

The role name	Time created	Operation		
AliyunCloudMonitorDefaultRole	2016-10-31 16:25:35	Management	Authorize	Delete
AliyunDRDSDefaultRole	2016-09-05 22:27:28	Management	Authorize	Delete
AliyunMNSLoggingRole	2016-11-14 16:30:12	Management	Authorize	Delete
RamTestAppReadOnly	2016-12-12 21:04:42	Management	Authorize	Delete
RamTestAppWrite	2016-12-12 21:24:06	Management	Authorize	Delete
SBCOSSArchiver	2016-11-09 11:59:16	Management	Authorize	Delete
SBCSystemAdmin	2016-11-09 11:56:43	Management	Authorize	Delete
SBCWebDeveloper	2016-11-09 11:55:49	Management	Authorize	Delete

Temporary access authorization

After creating roles, we can use them to grant temporary access to OSS.

Preparation

Authorization is required for assuming roles. Otherwise, any RAM user could assume these roles, which can lead to unpredictable risks. Therefore, to assume corresponding roles, a RAM user needs to have explicitly configured permissions.

1. Create two custom authorization policies in authorization policy management.

Create Authorization Policy

STEP 1: Select an authorization policy

STEP 2: Edit permissions and

* Authorization policy name :

AliyunSTSAssumeRoleAccess20151116044

The name must be 1-128 characters long a numbers, and "-"

Remarks :

Policy content :

```
1 {
2   "Statement": [
3     {
4       "Action": "sts:AssumeRole",
5       "Effect": "Allow",
6       "Resource":
7         "acs:ram::1894189769722283:rol
8     },
9   "Version": "1"
10  }
```

[Authorization policy format definition](#)
[Authorization policy FAQs](#)

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
```

```

    "Effect": "Allow",
    "Resource": "acs:ram::1894189769722283:role/ramtestappreadonly"
  },
  "Version": "1"
}

```

Create another custom authorization policy using the same method:

```

{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Resource": "acs:ram::1894189769722283:role/ramtestappwrite"
    }
  ],
  "Version": "1"
}

```

Here, the content entered after Resource is a role's ID. Role IDs can be found in **Roles > Role Details**.

2. Grant the two authorization policies to the account ram_test_app.

Use STS to grant access permissions

Now, we are ready with the platform to officially use STS to grant access permissions.

Here we use a simple STS Python command line tool [sts.py](#). The calling method is as follows:

```

$python ./sts.py AssumeRole RoleArn=acs:ram::1894189769722283:role/ramtestappreadonly RoleSessionName=usr001 Policy='{ "Version": "1", "Statement": [ { "Effect": "Allow", "Action": [ "oss:ListObjects", "oss:GetObject" ], "Resource": [ "acs:oss:*:*:ram-test-app", "acs:oss:*:*:ram-test-app/*" ] } ] }' DurationSeconds=1000 --id=id --secret=secret

```

- RoleArn: indicates the ID of a role to be assumed. Role IDs can be found in **Roles > Role details**.
- RoleSessionName: indicates the name of the temporary credentials. Generally, we recommend that you separate this using different application users.
- Policy: indicates a permission restriction, which is added when the role is assumed.
- DurationSeconds: indicate the validity time of the temporary credentials in seconds. The minimum value is 900, and the maximum value is 3600.
- id and secret: indicate the AccessKey of the RAM user to assume a role.

Here, we need to explain what is meant by “Policy”. The policy mentioned here is used to restrict the temporary credential permissions after a role is assumed. Ultimately, the permissions obtained

by means of temporary credentials are overlapping permissions of the role and the policy passed in.

When a role is assumed, a policy can be entered to increase the flexibility. For example, when uploading the files, we can add different upload path restrictions for different users. This is shown in the following example.

Now, let's test the STS function. To test the bucket, first use the console to put the file test.txt in ram-test-app, with the content ststest.

Firstly, use the RAM user account ram_test_app to directly access the file. Next, replace AccessKey with your own access key used in the test.

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn
-hangzhou.aliyuncs.com -i oOhue*****Frogv -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', '
keep-alive'), ('x-oss-request-id', '564A94D444F4D8B2225E4AFE'), ('date
', 'Tue, 17 Nov 2015 02:45:40 GMT'), ('content-type', 'application/xml
')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>564A94D444F4D8B2225E4AFE</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
get Failed!
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn
-hangzhou.aliyuncs.com -i oOhue*****Frogv -k OmVwFJO3qcT0*****
FhOYpg3p0KnA
100% Error Headers:
[('content-length', '229'), ('server', 'AliyunOSS'), ('connection', '
keep-alive'), ('x-oss-request-id', '564A94E5B1119B445B9F8C3A'), ('date
', 'Tue, 17 Nov 2015 02:45:57 GMT'), ('content-type', 'application/xml
')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>564A94E5B1119B445B9F8C3A</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
```

```
put Failed!
```

Without access permission, access attempts using the RAM user account `ram_test_app` are failed

Use temporary authorization for downloads

Now, we use STS to download files. To make it simple to understand, the entered policy and the role policy are the same. The expiration time is set to 3600s, and the app user here is `usr001`. The steps are as follows:

1. Use STS to obtain a temporary credential.

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$python ./sts.py AssumeRole RoleArn=acs:ram::1894189769722283:role/ramtestappreadonly RoleSessionName=usr001 Policy='{ "Version": "1", "Statement": [ { "Effect": "Allow", "Action": [ "oss:ListObjects", "oss:GetObject" ], "Resource": [ "acs:oss:*:*:ram-test-app", "acs:oss:*:*:ram-test-app/*" ] } ] }' --id=oOhue*****Frogv --secret=OmVwFJO3qcT0*****Fh0Ypg3p0KnA
https://sts.aliyuncs.com/?SignatureVersion=1.0&Format=JSON&Timestamp=2015-11-17T03%3A07%3A25Z&RoleArn=acs%3Aram%3A%3A1894189769722283%3Arole%2Framtestappreadonly&RoleSessionName=usr001&AccessKeyId=oOhuek56i53Frogv&Policy=%7B%22Version%22%3A%221%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%5B%22oss%3AListObjects%22%2C%22oss%3AGetObject%22%5D%2C%22Resource%22%3A%5B%22acs%3Aoss%3A%2A%3A%2A%3Aram-test-app%22%2C%22acs%3Aoss%3A%2A%3A%2A%3Aram-test-app%2F%2A%22%5D%7D%5D%7D&SignatureMethod=HMAC-SHA1&Version=2015-04-01&Signature=bshxPZpwRJv5ch3SjaBiXLodwq0%3D&Action=AssumeRole&SignatureNonce=53e1be9c-8cd8-11e5-9b86-008cfa5e4938
{
  "AssumedRoleUser": {
    "Arn": "acs:ram::1894189769722283:role/ramtestappreadonly/usr001",
    "AssumedRoleId": "317446347657426289:usr001"
  },
  "Credentials": {
    "AccessKeyId": "STS. 3mQEbNf*****wa180Le",
    "AccessKeySecret": "B1w7rCbR4dzGwNYJ*****3PiPqKZ3gjQhAxb6mB",
    "Expiration": "2015-11-17T04:07:25Z",
    "SecurityToken": "CAESvAMIARKAASQQUUTSE+7683CGlhdGsv2/di8uI+X1BxG7MDxM5FTd0fp5wpPK/7UctYH2MJ//c4yMN1PUCcEHI1zppCINmpDG2XeNA3OS16JwS6ESmI50sHyWBmsYkCJW15gXnfhz/OK+mSplbYxlfB33qfgCFe97Ijeuj8RMgqFx0Hny2BzGhhTVFMuM21RRWJOZnR5Yz11T3dhMTgwTGUieJmXnzQ0Njm0NzY1NzQyNjI4OSoGdXNyMDAxDXNjY2RkjoGUnNhTUQ1QpsBCgExGpUBCgVBbGxvdxI4CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9vc3M6TGldzde9iamVjdHMKDW9zczpHZXRPymp1Y3QSUgoOUmVzb3VyY2VFcXVhbHMSCFJlc291cmNlGjYKGGFjcZpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWNzOm9zczoqOio6cmFtLXRlc3QtYXBwLypKEDE4OTQxODk3Njk3MjIyODNSBTI2ODQyWg9Bc3N1bWVkbW9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJyYW10ZXN0YXBwcmVhZG9ubHk="
  },
  "RequestId": "8C009F64-F19D-4EC1-A3AD-7A718CD0B49B"
}
```



```
}
```

2. Use the temporary credential to download files. Here sts_token is the SecurityToken returned by the STS.

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd get oss://ram-test-app/test.txt test.txt --host=oss-cn-
-hangzhou.aliyuncs.com -i STS. 3mQEbnf*****wa180Le -k Blw7rCbR4d
zGwNYJ*****3PiPqKZ3gjQhAxb6mB --sts_token=CAESvAMIARKAASQQUUTSE+
7683CGLhdGsv2/di8uI+XlBxG7MDxM5FTd0fp5wpPK/7UctYH2MJ///c4yMN1PUCc
EHl1zppCINmpDG2XeNA3OS16JwS6ESmI50sHyWBmsYkCJW15gXnfhz/OK+mSplbYxlfB
33qfgCFE97IjeuJ8RMgqFxoHny2BzGhhTVFmuM21RRWJOZnR5Yzl1T3dhMTg
wTGUiEjMxNzQ0NjM0NzY1NzQyNjI4OSoGdXNyMDAxMJTrgJ2RKjoGUNhTUQ
lQpsBCgExGpUBCgVBbGxvdxI4CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9
vc3M6TG1zdE9iamVjdHMKDW9zcZpHZXRPympLY3QSUgoOUmVzb3VyY2VFcXV
hbHMSCFJlc291cmNlGjYKGGFjcZpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWN
zOm9zczoqOio6cmFtLXRlc3QtYXBwLypKEDE4OTQxODk3Njk3MjIyODNSBTI
2ODQyWg9Bc3N1bWVlUm9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJ
yYW10ZXN0YXBwcmVhZG9ubHk=
100% The object test.txt is downloaded to test.txt, please check.
0.061(s) elapsed
```

3. As you can see, we can use the temporary credentials to download the file. Next, we will test if we can use them to upload a file.

```
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-
-hangzhou.aliyuncs.com -i STS. 3mQEbnf*****wa180Le -k Blw7rCbR4d
zGwNYJ*****3PiPqKZ3gjQhAxb6mB --sts_token=CAESvAMIARKAASQQUUTSE+
7683CGLhdGsv2/di8uI+XlBxG7MDxM5FTd0fp5wpPK/7UctYH2MJ///c4yMN1PUCc
EHl1zppCINmpDG2XeNA3OS16JwS6ESmI50sHyWBmsYkCJW15gXnfhz/OK+mSplbYxlfB
33qfgCFE97IjeuJ8RMgqFxoHny2BzGhhTVFmuM21RRWJOZnR5Yzl1T3dhMTg
wTGUiEjMxNzQ0NjM0NzY1NzQyNjI4OSoGdXNyMDAxMJTrgJ2RKjoGUNhTUQ
lQpsBCgExGpUBCgVBbGxvdxI4CgxBY3Rpb25FcXVhbHMSBkFjdGlvbhogCg9
vc3M6TG1zdE9iamVjdHMKDW9zcZpHZXRPympLY3QSUgoOUmVzb3VyY2VFcXV
hbHMSCFJlc291cmNlGjYKGGFjcZpvc3M6KjoqOnJhbS10ZXN0LWFwcAoaYWN
zOm9zczoqOio6cmFtLXRlc3QtYXBwLypKEDE4OTQxODk3Njk3MjIyODNSBTI
2ODQyWg9Bc3N1bWVlUm9sZVVzZXJgAGoSMzE3NDQ2MzQ3NjU3NDI2Mjg5chJ
yYW10ZXN0YXBwcmVhZG9ubHk=
100% Error Headers:
[('content-length', '254'), ('server', 'AliyunOSS'), ('connection',
'keep-alive'), ('x-oss-request-id', '564A9A2A1790CF0F53C15C82'),
('date', 'Tue, 17 Nov 2015 03:08:26 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9A2A1790CF0F53C15C82</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
put Failed!
```

The file upload is failed. This is because the assumed role only has download permission hence.


```

NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqEjM1NTQwNzg0NzY2
MDAyOTQyOHIPcmFtdGVzdGFwcHdyXRl
Error Headers:
[('content-length', '254'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A9C31FFFC811F24B6E7E3'), ('date', 'Tue, 17 Nov 2015 03:17:05 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9C31FFFC811F24B6E7E3</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
get Failed!
[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/test.txt --host=oss-cn-hangzhou.aliyuncs.com -i STS.rtfx13*****NlIJlS4U -k 2fsaM8E2maB2dn*****wpsKTyK4ajo7TxFr0zIM --sts_token=CAESkwMIARKAAUh3/Uzcgl3YLRB
Wxy0IZjGewMpg3lITxCleBFUleO/3Sgpubid+GVs+OlvlvXJn6DLcvPa8azK
JKtzV0oKSy+mwUrxSvUSRDntrs78CsNfWoOJUMJKjLixdWnGilpgxJCBzNZ2
YV/6ycTaZySSE1V6kqQ7A+GPwYoBSnWmLpdGhhTVFMucnRmeDEzRf1NVWJjTmx
JSmxTNFUiEjM1NTQwNzg0NzY2MDAyOTQyOCgDxNyMDAxMOPzoJ2RKjoGUNN
hTUQ1QnYKATEacQoFQWxs3cSJwoMQWN0aW9uRXFlYWxzEgZBY3Rpb24aDwo
Nb3NzOlBlde9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIIUmVzb3VyY2UaIwoh
YWNzOm9zczoqOio6cmFtLXRlc3QtYXBwL3VzcjAwMS8qShAxODk0MTg5NzY5
NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqEjM1NTQwNzg0NzY2
MDAyOTQyOHIPcmFtdGVzdGFwcHdyXRl
100% Error Headers:
[('content-length', '254'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id', '564A9C3FB8DE437A91B16772'), ('date', 'Tue, 17 Nov 2015 03:17:19 GMT'), ('content-type', 'application/xml')]
Error Body:
<? xml version="1.0" encoding="UTF-8"? >
<Error>
  <Code>AccessDenied</Code>
  <Message>Access denied by authorizer's policy.</Message>
  <RequestId>564A9C3FB8DE437A91B16772</RequestId>
  <HostId>ram-test-app.oss-cn-hangzhou.aliyuncs.com</HostId>
</Error>
Error Status:
403
put Failed!

```

The test.txt upload fails. We have formatted the entered policy discussed at the beginning of this document, which is as follows:

```

{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:PutObject"
      ],
      "Resource": [
        "acs:oss:*:*:ram-test-app/usr001/*"
      ]
    }
  ]
}

```

```

    }
  ]
}

```

This policy indicates that users are only allowed to upload files like `usr001/` to the `ram-test-app` bucket. If the app user is `usr002`, the policy can be changed to only allow for the uploading of files like `usr002/`. By setting different policies for different app users, we can isolate the storage space of different app users.

3. Retry the test and specify the upload destination as `ram-test-app/usr001/test.txt`.

```

[admin@NGIS-CWWF344M01C /home/admin/oss_test]
$./osscmd put test.txt oss://ram-test-app/usr001/test.txt --
host=oss-cn-hangzhou.aliyuncs.com -i STS.rtfx13*****NlIJlS4U -k
2fsaM8E2maB2dn*****wpsKTyK4ajo7TxFr0zIM --sts_token=CAESkwMIAR
KAAUh3/Uzcg13YLRBWxy0IZjGewMpg31ITxCleBFUleO/3Sgpudid+GVs+OlvulvXJn6
DLcvPa8azKJKtzV0oKSy+mwUrxSvUSRVDntrs78CsNfWoOJUMJKjLIxdWnGi1
pgxJCBzNZ2YV/6ycTaZySSE1V6kqQ7A+GPwYoBSnWmLpdGhhTVFMucnRmeDEzR
FlNVWJjTmxJSmxTNFUiEjMlNTQwNzg0NzY2MDAyOTQyOCgGdXNyMDAxMOPzo
J2RKjoGUnNhTUQ1QnYKATEacQoFQWxs3cSjWomQWN0aw9uRxF1YWxzEgZBY
3Rpb24aDwoNb3NzOlBldE9iamVjdBI/Cg5SZXNvdXJjZUVxdWFscxIIUmVzb3
VyY2UaIwohYWNzOm9zczoQio6cmFtLXRlc3QtYXBwL3VzcjAwMS8qShAxOD
k0MTg5NzY5NzIyMjgzUgUyNjg0MloPQXNzdW1lZFJvbGVVc2VyYABqEjMlNT
QwNzg0NzY2MDAyOTQyOHIPcmFtdGVzdGFwcHdyXRl
100%
Object URL is: http://ram-test-app.oss-cn-hangzhou.aliyuncs.com/
usr001%2Ftest.txt
Object abstract path is: oss://ram-test-app/usr001/test.txt
ETag is "946A0A1AC8245696B9C6A6F35942690B"
0.071(s) elapsed

```

The upload is successful.

Summary

This section describes how to grant users temporary access authorization for OSS using STS. In typical mobile development scenarios, STS can be used to grant temporary authorizations to access OSS when different app users need to access the app. The temporary authorization can be configured with expiration time to greatly reduce the hazards caused by leaks. When obtaining temporary authorization, we can enter different authorization policies for different app users to restrict their access permissions. For example, to restrict the object paths accessible to users. This isolates the storage space of different app users.

1.7 FAQs about subaccount settings

How to create an STS temporary account and how to use it to access resources?

See [STS temporary access authorization](#).

Client or console logon error reported for an authorized sub-account

How to authorize a sub-account with the operation permission for a single bucket

How to authorize a sub-account with the operation permission for a directory in a bucket

How to authorize a sub-account with the read-only permission for a bucket

Error upon an OSS SDK call: InvalidAccessKeyId

See [STS errors and troubleshooting](#).

Error upon an STS call: Access denied by authorizer's policy

Detailed error information: ErrorCode: AccessDenied ErrorMessage: Access denied by authorizer's policy.

Cause of the error:

- The temporary account has no access permission.
- The authorization policy specified for assuming the role of this temporary account does not assign the access permission to the account.

For more STS errors and the causes, see [OSS permission errors and troubleshooting](#).

2 Data security

2.1 Check data transmission integrity by using 64-bit CRC

Background

An error may occur when data is transmitted between the client and the server. Currently, OSS can return the 64-bit CRC value for an object uploaded in any mode. To check the data integrity, the client can compare the 64-bit CRC value with the locally calculated value.

- OSS calculates 64-bit CRC value for newly uploaded object, stores the result as metadata of the object, and then adds the x-oss-hash-crc64ecma header to the returned response header, indicating its 64-bit CRC value. This 64-bit CRC is calculated according to ECMA-182 Standard .
- For the object that already exists on OSS before the 64-bit CRC goes live, OSS does not calculate its 64-bit CRC value. Therefore, its 64-bit CRC value is not returned when such object is obtained.

Operation instructions

- Put Object / Append Object / Post Object / Multipart upload part returns the corresponding 64-bit CRC value. The client can get the 64-bit CRC value returned by the server after the upload is completed and can check it against the locally calculated value.
- In the case of Multipart Complete, if all the parts have their respective 64-bit CRC values, then the 64-bit CRC value of the entire object is returned. Otherwise, the 64-bit CRC value is not returned (for example, if a part has been uploaded before the 64-bit CRC goes live).
- Get Object / Head Object / Get ObjectMeta returns the corresponding 64-bit CRC value (if any). After Get Object is completed, the client can get the 64-bit CRC value returned by the server and check it against the locally calculated value.

**Note:**

The 64-bit CRC value of the entire object is returned for the range get object.

- For copy related operations, for example, Copy Object/Upload Part Copy, the newly generated object/Part may not necessarily have the 64-bit CRC value.

Python example

An example of complete Python code is as follows. It shows how to check data transmission integrity based on the 64-bit CRC value.

1. Calculate the 64-bit CRC value.

```
import oss2
from oss2.models import PartInfo
import os
import crcmod
import random
import string
do_crc64 = crcmod.mkCrcFun(0x142F0E1EBA9EA3693L, initCrc=0L, xorOut=
0xffffffffffffffffL, rev=True)
def check_crc64(local_crc64, oss_crc64, msg="check crc64"):
    if local_crc64 != oss_crc64:
        print "{0} check crc64 failed. local:{1}, oss:{2}.".format(msg,
        local_crc64, oss_crc64)
        return False
    else:
        print "{0} check crc64 ok.".format(msg)
        return True
def random_string(length):
    return ''.join(random.choice(string.lowercase) for i in range(length
    ))
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret),
    endpoint, bucket_name)
```

2. Verify Put Object.

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

3. Verify Get Object.

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

4. Verify Upload Part and Complete.

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

```
result = bucket.complete_multipart_upload(key, upload_id,
part_info_list)
oss_crc64 = result.headers.get('x-oss-hash-crc64ecma', '')
local_crc64 = str(do_crc64(part_2, do_crc64(part_1)))
#Check whether the final object on the OSS is consistent with the
local file
check_crc64(local_crc64, oss_crc64, "complete object")
```

OSS SDK support

Part of the OSS SDK already supports the data validation using crc64 for the upload and download, as shown in the following table:

SDK	Support for CRC?	Example
Java SDK	Yes	CRCSample.java
Python SDK	Yes	object_check.py
PHP SDK	No	N/A
C# SDK	No	None
C SDK	Yes	oss_crc_sample.c
JavaScript SDK	No	None
Go SDK	Yes	crc_test.go
Ruby SDK	No	None
iOS SDK	Yes	OSSCrc64Tests.m
Android SDK	Yes	OSSCrc64Tests.m

2.2 Protect data through client encryption

Client encryption means that the encryption is completed before the user data is sent to the remote server, whereas the plaintext of the key used for encryption is kept in the local computer only. Therefore, the security of user data can be ensured because others cannot decrypt the data to obtain the original data even if the data leaks.

This document describes how to protect data through client encryption based on the current Python SDK version of OSS.

Principles

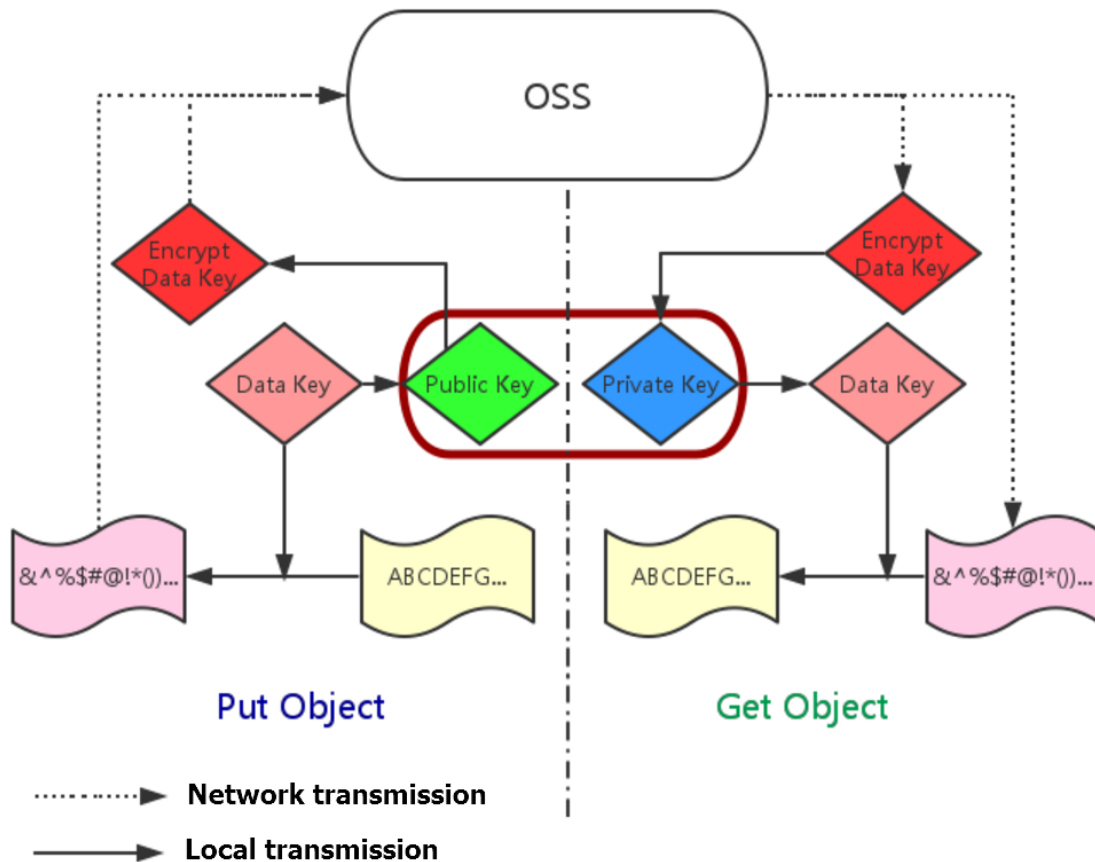
1. The user maintains a pair of RSA keys (`rsa_private_key` and `rsa_public_key`) in the local computer.

2. Each time when any object is uploaded, a symmetric key `data_key` of AES256 type is generated randomly, and then `data_key` is used to encrypt the original content to obtain `encrypt_content`.
3. Use `rsa_public_key` to encrypt `data_key` to obtain `encrypt_data_key`, place it in the request header as the custom meta of the user, and send it together with `encrypt_content` to the OSS.
4. When Get Object is performed, `encrypt_content` and `encrypt_data_key` in the custom meta of the user are obtained first.
5. The user uses `rsa_private_key` to decrypt `encrypt_data_key` to obtain `data_key`, and then uses `data_key` to decrypt `encrypt_content` to obtain the original content.

**Note:**

The user's key in this document is an asymmetric RSA key, and the AES256-CTR algorithm is used when object content is encrypted. For more information, see [PyCrypto Document](#). This document describes how to implement client encryption through the custom meta of an object. The user can select the encryption key type and encryption algorithm as required.

Structural diagram



Preparation

1. Install the PyCrypto library.

```
pip install pycrypto
```

Example of complete Python code

```
# -*- coding: utf-8 -*-
import os
import shutil
import base64
import random
import oss2
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto import Random
from Crypto.Util import Counter
# aes 256, key always is 32 bytes
_AES_256_KEY_SIZE = 32
_AES_CTR_COUNTER_BITS_LEN = 8 * 16
class AESCipher:
    def __init__(self, key=None, start=None):
        self.key = key
```

```

        self.start = start
        if not self.key:
            self.key = Random.new().read(_AES_256_KEY_SIZE)
        if not self.start:
            self.start = random.randint(1, 10)
        ctr = Counter.new(_AES_CTR_COUNTER_BITS_LEN, initial_value=
self.start)
        self.cipher = AES.new(self.key, AES.MODE_CTR, counter=ctr)
        def encrypt(self, raw):
            return self.cipher.encrypt(raw)
        def decrypt(self, enc):
            return self.cipher.decrypt(enc)
# First, initialize the information such as AccessKeyId, AccessKeyS
ecret, and Endpoint.
# Obtain the information through environment variables or replace the
information such as "<Your AccessKeyId>" with the real AccessKeyId,
and so on.

# Use Hangzhou region as an example. Endpoint can be:
# http://oss-cn-hangzhou.aliyuncs.com
# https://oss-cn-hangzhou.aliyuncs.com
# Access using the HTTP and HTTPS protocols respectively.
access_key_id = os.getenv('OSS_TEST_ACCESS_KEY_ID', '<your AccessKeyId
>')
access_key_secret = os.getenv('OSS_TEST_ACCESS_KEY_SECRET', '<Your
AccessKeySecret>')
bucket_name = os.getenv('OSS_TEST_BUCKET', '<Your Bucket>')
endpoint = os.getenv('OSS_TEST_ENDPOINT', '<Your Access Domain Name>')
# Make sure that all the preceding parameters have been filled in
correctly.
for param in (access_key_id, access_key_secret, bucket_name, endpoint
):
    assert '<' not in param, 'Please set the parameter:' + param
##### 0 prepare #####
# 0.1 Generate the RSA key file and save it to the disk
rsa_private_key_obj = RSA.generate(2048)
rsa_public_key_obj = rsa_private_key_obj.publickey()
encrypt_obj = PKCS1_OAEP.new(rsa_public_key_obj)
decrypt_obj = PKCS1_OAEP.new(rsa_private_key_obj)
# save to local disk
file_out = open("private_key.pem", "w")
file_out.write(rsa_private_key_obj.exportKey())
file_out.close()
file_out = open("public_key.pem", "w")
file_out.write(rsa_public_key_obj.exportKey())
file_out.close()
# 0.2 Create the Bucket object. All the object-related interfaces can
be implemented by using the Bucket object
bucket = oss2. Bucket(oss2. Auth(access_key_id, access_key_secret),
endpoint, bucket_name)
obj_name = 'test-sig-1'
content = "test content"
#### 1 Put Object ####
# 1.1 Generate the one-time symmetric key encrypt_cipher used to
encrypt this object, where key and start are values generated at
random
encrypt_cipher = AESCipher()
# 1.2 Use the public key to encrypt the information for assisting
encryption, and save it in the custom meta of the object. When Get
Object is performed later, we can use the private key to perform
decryption and obtain the original content according to the custom
meta

```

```
headers = {}
headers['x-oss-meta-x-oss-key'] = base64.b64encode(encrypt_obj.encrypt(
    encrypt_cipher.key))
headers['x-oss-meta-x-oss-start'] = base64.b64encode(encrypt_obj.
    encrypt(str(encrypt_cipher.start)))
# 1.3. Use encrypt_cipher to encrypt the original content to obtain
encrypt_content
encrypt_content = encrypt_cipher.encrypt(content)
# 1.4 Upload the object
result = bucket.put_object(obj_name, encrypt_content, headers)
if result.status / 100 != 2:
    exit(1)
#### 2 Get Object ####
# 2.1 Download the encrypted object
result = bucket.get_object(obj_name)
if result.status / 100 != 2:
    exit(1)
resp = result.resp
download_encrypt_content = resp.read()
# 2.2 Resolve from the custom meta the key and start that are
previously used to encrypt this object
download_encrypt_key = base64.b64decode(resp.headers.get('x-oss-meta-x-
    -oss-key', ''))
key = decrypt_obj.decrypt(download_encrypt_key)
download_encrypt_start = base64.b64decode(resp.headers.get('x-oss-meta-
    -x-oss-start', ''))
start = int(decrypt_obj.decrypt(download_encrypt_start))
# 2.3 Generate the cipher used for decryption, and decrypt it to
obtain the original content
decrypt_cipher = AESCipher(key, start)
download_content = decrypt_cipher.decrypt(download_encrypt_content)
if download_content != content:
    print "Error!"
else:
    print "Decrypt ok. Content is: %s" % download_content
```

3 OSS resource monitoring and alarm service

The CloudMonitor service can monitor OSS resources. You can use CloudMonitor to view resource usage, performance, and health status on Alibaba Cloud. Using the alarm service, you can react rapidly to keep applications running smoothly. This article introduces how to monitor OSS resources, set OSS alarm rules, and create custom monitoring dashboard.

Prerequisites

- Activate the OSS service.
- Activate the CloudMonitor service.

Monitor OSS resources

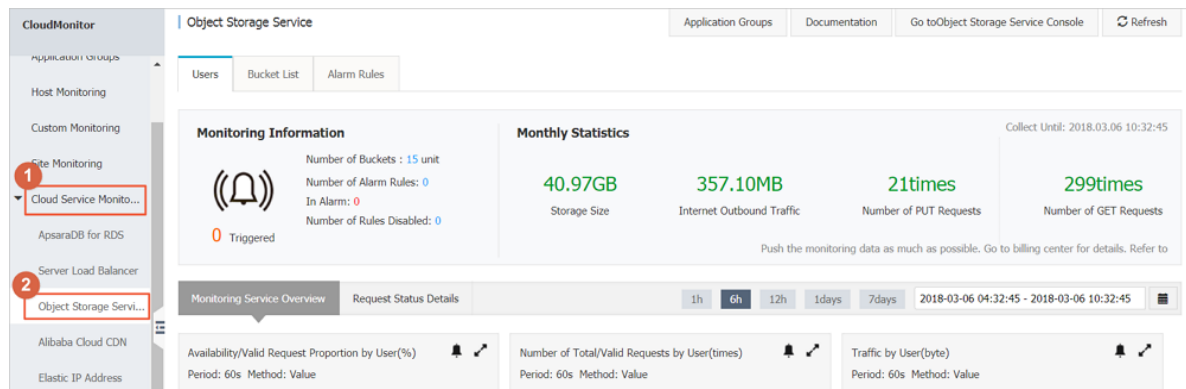
1. Log on to the CloudMonitor console.
2. Select **Cloud Service Monitoring** > **Object Storage Service** from the left-side navigation pane to enter the OSS monitoring page, as shown in the following figure.

You can obtain monitoring data on the OSS monitoring page.



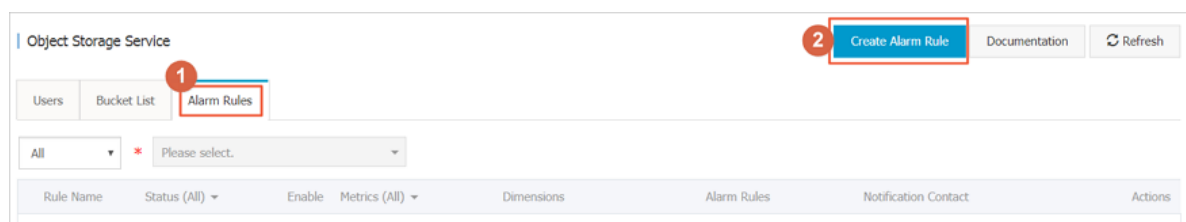
Note:

“by User” refers to user-level data, that is, all bucket data of this user.



Set alarm rules

1. Find the **Alarm Rules** tab on OSS monitoring page, and then click **Create Alarm Rule**.



2. Configure your alarm rules.

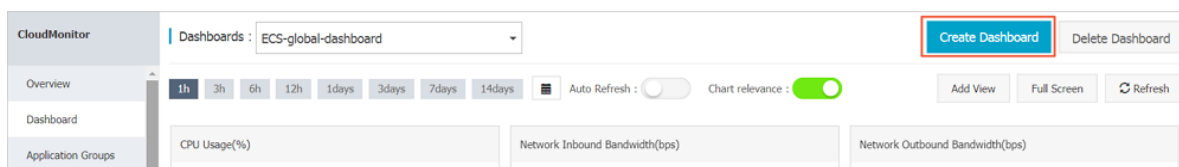
For configuration details, see [Manage alarm rules](#).

3. The alarm rule is generated when the configuration is completed. You can use test data to check whether the rule has taken effect by verifying if the alarm information was received successfully (over email, SMS, Trademanager, or DingTalk).

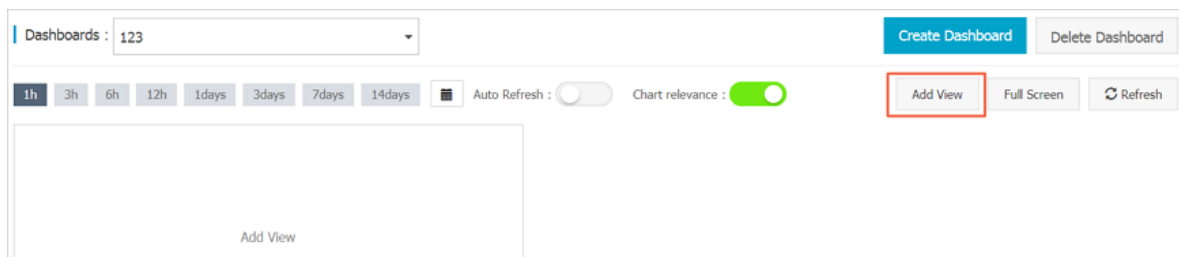
Custom monitoring dashboard

You can customize the OSS resource monitoring map on the CloudMonitor Console. The procedure is as follows.

1. Log on to the CloudMonitor console.
2. Click **Dashboard** from the left-side navigation pane.
3. Click **Create Dashboard**.



4. Enter the name of dashboard, and then click **Add View**.



5. Configure tables as required, and then click **Save**.

For configuration details, see [Monitoring indicators reference](#).

4 OSS performance and scalability best practice

Partitions and naming conventions

OSS automatically partitions user data by file names encoded in UTF-8 to process massive data and meet the needs for high request rates. However, if you use sequential prefixes (such as timestamps and sequential numbers) as part of the names when uploading a large number of objects, there may be lots of file indexes stored in a single partition. In this way, when the request rates exceed 2,000 operations per second (downloading, uploading, deleting, copying, and obtaining metadata are each counted as one operation, while deleting or enumerating more than one files in batch is considered as multiple operations), the following results may occur:

- This partition becomes a hotspot partition, leading to the exhausted I/O capacity and low request rate limited automatically by the system.
- With a hotspot partition, the partitioned data is constantly rebalanced, which may increase the processing time.

Therefore, the horizontal scaling capability of OSS is affected, thus resulting in limited request rate

To address these issues, you must delete the sequential prefixes in the file names. Instead, you can add random prefix in file names. In this way, the file indexes (and I/O loads) are evenly distributed in different partitions.

The following shows the examples of changing sequential prefixes into random prefixes.

- Example 1: Add hex hash prefixes into file names

As shown in this example, you may use a combination of dates and customer IDs (including sequential timestamp prefixes) in file names:

```
sample-bucket-01/2017-11-11/customer-1/file1
sample-bucket-01/2017-11-11/customer-2/file2
sample-bucket-01/2017-11-11/customer-3/file3
...
sample-bucket-01/2017-11-12/customer-2/file4
sample-bucket-01/2017-11-12/customer-5/file5
sample-bucket-01/2017-11-12/customer-7/file6
```

...

In this case, you can calculate a hash value for the customer ID, that is, the MD5 (customer-id), and combine a hash prefix of several characters as the prefix to the file name. If you use a 4-character hash prefix, the file names are as follows:

```
sample-bucket-01/2c99/2017-11-11/customer-1/file1
sample-bucket-01/7a01/2017-11-11/customer-2/file2
sample-bucket-01/1dbd/2017-11-11/customer-3/file3
...
sample-bucket-01/7a01/2017-11-12/customer-2/file4
sample-bucket-01/b1fc/2017-11-12/customer-5/file5
sample-bucket-01/2bb7/2017-11-12/customer-7/file6
...
```

In this case, a 4-character hex hash value is used as the prefix, and each character can be any one of the 16 values (0-f), so there are $16^4=65,536$ possible character combinations. Technically, the data in the storage system is constantly partitioned into up to 65,536 partitions. Leveraging the performance bottleneck limit (2,000 operations per second) and the request rate of your service, you can determine a proper number of hash buckets.

If you want to list all the files with a specific date in the file name, for example, files with 2017-11-11 in the name in sample-bucket-01, you must enumerate the files in sample-bucket-01 (acquire all files in sample-bucket-01 in batch by multiple calls of the List Object API) and combine files with this date in the file names.

- Example 2: Reverse the file name

In this example, you may use a UNIX timestamp with millisecond precision to generate file names, which is also a sequential prefix:

```
sample-bucket-02/1513160001245.log
sample-bucket-02/1513160001722.log
sample-bucket-02/1513160001836.log
sample-bucket-02/1513160001956.log
...
sample-bucket-02/1513160002153.log
sample-bucket-02/1513160002556.log
sample-bucket-02/1513160002859.log
...
```

As mentioned in the preceding paragraph, if you use the sequential prefix in file names, the performance may be affected when the request rate exceeds a certain limit. To address this issue, you can reverse the timestamp prefix to exclude the sequential prefix. The result is as follows:

```
sample-bucket-02/5421000613151.log
sample-bucket-02/2271000613151.log
```



```
sample-bucket-02/6381000613151.log  
sample-bucket-02/6591000613151.log  
...  
sample-bucket-02/3512000613151.log  
sample-bucket-02/6552000613151.log  
sample-bucket-02/9582000613151.log  
...
```

The first three digits of the file name represent the millisecond, which can be any one of the 1,000 values. The fourth digit changes every second. Similarly, the fifth digit changes every 10 seconds. In this way, the prefixes are randomly specified and the loads are distributed evenly to multiple partitions, thus avoiding the performance bottleneck.