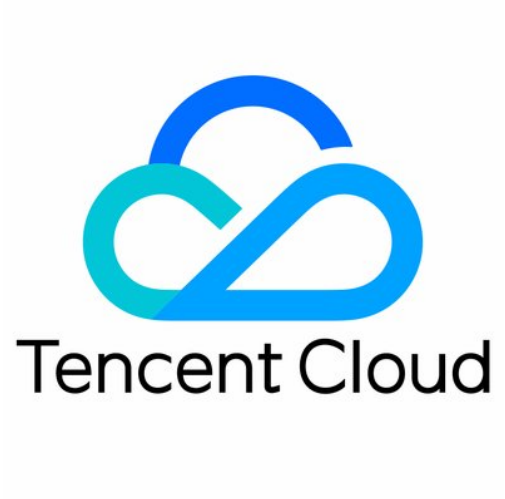
Cloud Infinite

Developer Manual

Product Documentation



Contents

Developer Manual Access Management Overview

Last updated : 2020-04-24 09:57:48

You can create users or roles by using Cloud Access Management (CAM). By associating CAM preset or custom policies, you can specify which users have which access permissions to which resources. You can also specify that access permissions are only available under certain conditions.

Cloud Infinite (CI) has two major modules: image processing and media processing. For image processing, you can configure persistent permissions for your sub-account to process data when uploading or process COS data already stored in the cloud. To grant permissions to your sub-account to process data when downloading, please submit a ticket to contact us. For media processing, CI allows you to grant permissions at the resource level. You can grant permissions to allow a sub-account to manage a single resource through policy syntax.

See the basic concepts below. For more information, see the CAM User Guide documentation.

Accounts

- **Root account**: this account owns all the Tencent Cloud resources and has full access to these resources.
- **Sub-account**: includes sub-users and collaborators.
- **Sub-user**: created by a root account and is subordinate to the root account.
- **Collaborator**: when a root account is added as the collaborator of the current root account, it becomes one of the sub-accounts of the current root account. The account can be switched from collaborator back to root account.
- **Identity credentials**: include login credentials and access certificates.
  - **Login credentials**: refers to usernames and passwords.
  - **Access certificate**: refers to the Cloud API key pairs (SecretId and SecretKey).

Resources and Permissions

- **Resource**: an object that you can operating on, such as a COS bucket or an image in CI.

- **Permission**: an authorization to allow or forbid some users to perform certain operations. By default, the root account has full access to all resources under it, while a sub-account has no access to any resources under its root account.
- **Policy**: a set of syntax rules that define and describe one or more permissions. A root account grants access for users or user groups by associating policies with them. For more information, see Access Policy Language Overview.

Authorizing Sub-Accounts to Access CI Services

Last updated : 2020-07-10 17:01:13

Cloud Infinite (CI) provides a suite of data processing services, among which CI storage is based on COS. Therefore, for a sub-account to use all CI services, it must be granted necessary read/write permissions both for CI and COS.
Granting CI operation permissions to a sub-account involves three steps: **Create a Sub-Account and Grant CI Permissions**, **Grant COS permissions to the Sub-Account**, and **Use the Sub-Account to Process Data**.

⚠ Note :

A sub-account may need the following permissions to view data or change settings in the CI console:

Operation	Permission
Create a bucket	cos:PutBucket permission for COS bucket
Unbind a bucket	cos:DeleteBucket permission for COS bucket
View feature configuration	cos:GetBucket permission for COS bucket
Modify feature configuration	cos:PutObject permission for COS bucket

Step 1. Create a Sub-Account and Grant CI Permissions

You can create a sub-account in CAM Console and grant CI access permissions for it. The specific procedure is as follows:

1. Log in to CAM Console. In the left sidebar, choose **Users** > **User List**.

2. In the user list page, click **Create User**.

3. Click **Custom Create** to open the **Select Type** page.

4. Click **Access to resources and receive messages** > **Next** to enter the **Enter user info** page.

5. Enter user information. Here, you can create multiple sub-users, set the access type and console password, or perform other operations.

6. Click **Next** to set user permissions. Choose **Select policies from the policy list**, and then **QcloudCIFullAccess** for full CI access from the policy list. Click **Next**.

7. After confirming that the information is correct, click **OK** to create the sub-account.

Step 2. Grant COS Permissions to the Sub-Account

To grant **COS resource access permissions** to the sub-account, associate a preset policy with it as follows:

1. Log in to CAM Console. In the left sidebar, choose **Users** > **User List**.
2. In the user list page, find the sub-user that you just created, and click **Grant Permission** under **Operation**.
3. In the policy list, select the appropriate permission policy, and click **OK** to associate it with the sub-user

You can also grant COS permissions to a sub-account using a custom policy. For more information, see the Authorization Management document and policy examples.

Step 3. Use the Sub-Account to Process Data

To use a sub-account to process data, you need the APPID of the root account and the SecretId and SecretKey of the sub-account.

1. Log in to CAM Console with your root account. In the left sidebar, choose **Users** > **User List**.
2. Click on the name of the sub-account whose SecretId and SecretKey you want.
3. Select the **API Key** tab where you can get the SecretId and SecretKey.

You can also get the API Key from the CAM console using a sub-account by granting it CAM read permission.

Configuring Data Persistence Permissions for Sub-Accounts

Last updated : 2020-03-02 10:46:56

Currently, you can grant permissions to a Cloud Infinite (CI) sub-account to perform persistence operations by associating the write permission with given resources in COS. The following cases illustrate how to grant permissions to a CI sub-account to perform persistence operations on **all resources** and on **specific resources**.

> Before configuring data persistence permissions for your sub-account, you must associate the CI full read-write access **QcloudCIFullAccess**.

When configuring a custom policy, you can copy and paste the following reference policy into the **Edit Policy Content** input box and modify it based on the actual settings. For more information, see the CAM Policy Syntax document.

Authorizing a sub-account to perform persistence operations on all resources

Assume that an enterprise account named CompanyExample (whose OwnerUin is 100000000001 and APPID is 1250000000) has a sub-account named Developer, which needs to perform persistence processing on all resources under CompanyExample.

In this case, the following solutions are available to authorize the CI sub-account to perform persistence processing by granting the write permission to all resources under the account CompanyExample in COS.

Solution A:

CompanyExample grants the **QcloudCOSDataWriteOnly** preset policy to Developer. For more information on authorization, see Authorization Management.

Solution B:

Step 1: create the following policy by using policy syntax.

```
{
"version": "2.0",
"statement": [
{
"effect": "allow",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource": "*"
}
]
}
```

Step 2: associate this policy with the sub-account. For more information on authorization, see Authorization Management.

Authorizing a sub-account to perform persistence operations on resources under a specific directory

Assume that an enterprise account named CompanyExample (whose OwnerUin is 100000000001 and APPID is 1250000000) has a sub-account named Developer, which needs to perform persistence processing on resources under the doc directory in a bucket named examplebucket locating in the Shanghai region.

In this case, the following solutions are available to authorize the CI sub-account to perform persistence processing by granting the write permission to resources under the specified directory in COS.

Solution A:

Configure policies and the ACL for resources in COS Console. For more information, see the Adding Bucket Policies document for COS.

Solution B:

Step 1: create the following policy by using policy syntax.

```
{
"version": "2.0",
"statement": [
{
"effect": "allow",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource":"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-1250000000/doc/*"
}
]
}
```

Step 2: associate this policy with the sub-account. For more information on authorization, see Authorization Management.

Authorizing a sub-account to perform persistence operations on a specific resource

Assume that an enterprise account named CompanyExample (whose OwnerUin is 100000000001 and APPID is 1250000000) has a sub-account named Developer, which needs to perform persistence processing on the picture.jpg image file under the doc directory in a bucket named examplebucket locating in the Shanghai region.

In this case, the following solutions are available to authorize the CI sub-account to perform persistence processing by granting the write permission to the specified resource in COS.

Solution A:

Configure policies and the ACL for resources in COS Console. For more information, see the Adding Bucket Policies document for COS.

Solution B:

Step 1: create the following policy by using policy syntax.

```
{
"version": "2.0",
"statement": [
{
"effect": "allow",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource":"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-1250000000/doc/picture.jpg"
}
]
}
```

Step 2: associate this policy with the sub-account. For more information on authorization, see Authorization Management.

Authorizing a sub-account to perform persistence operations on a specific resources with a specific prefix

Assume that an enterprise account named CompanyExample (whose OwnerUin is 100000000001 and APPID is 1250000000) has a sub-account named Developer, which needs to perform persistence processing on resources with the prefix of **test** under the doc directory in a bucket named examplebucket locating in the Shanghai region.

In this case, the following solutions are available to authorize the CI sub-account to perform persistence processing by granting the write permission to resources with the specified prefix in COS.

Solution A:

Configure policies and the ACL for resources in COS Console. For more information, see the Adding Bucket Policies document for COS.

Solution B:

Step 1: create the following policy by using policy syntax.

```
{
"version": "2.0",
"statement": [
{
"effect": "allow",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource":"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-1250000000/doc/test*"
}
]
}
```

Step 2: associate this policy with the sub-account. For more information on authorization, see Authorization Management.

Authorizing a sub-account to perform persistence operations on all resources under a specific directory except specified files

Assume that an enterprise account named CompanyExample (whose OwnerUin is 100000000001 and APPID is 1250000000) has a sub-account named Developer, which needs to perform persistence processing on all resources under the doc directory in a bucket except the image file picture.jpg. The bucket is named examplebucket and locates in the Shanghai region.

In this case, the following solutions are available to authorize the CI sub-account to perform persistence processing by granting the write permission to a given files under the specified directory in COS.

Solution A:

Configure policies and the ACL in COS Console. For more information, see the Adding Bucket Policies document for COS.

Solution B:

Step 1: create the following policy by using policy syntax.

```
{
"version": "2.0",
"statement": [
{
"effect": "allow",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource":"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-1250000000/doc/*"
},

{
"effect": "deny",
"action": [
"cos:ListParts",
"cos:PostObject",
"cos:PutObject*",
"cos:InitiateMultipartUpload",
"cos:UploadPart",
"cos:UploadPartCopy",
"cos:CompleteMultipartUpload",
"cos:AbortMultipartUpload"
],
"resource":"qcs::cos:ap-shanghai:uid/1250000000:examplebucket-1250000000/doc/picture.jpg"
}
]
}
```

Step 2: associate this policy with the sub-account. For more information on authorization, see Authorization Management.

Request Signature

Last updated : 2020-06-23 14:57:00

Tencent Cloud Infinite (CI) verifies the validity of requests through signatures. Developers grant a signature to the client to allow it to upload, download, and manage specific resources.

With CI, an anonymous HTTP request or signed HTTP request can be made by using the RESTful API. For a signature request, the server will authenticate the initiator.

- Anonymous request: the HTTP request does not include any identity or authentication information, and the HTTP request is made through the RESTful API.
- Signature request: a signature is included in the HTTP request, and authentication is performed after the server receives the request. The request is approved and executed after a successful authentication, otherwise it is discarded with an error message.

Cloud Infinite uses the same signature algorithm as Cloud Object Storage (COS) on which it is based. It performs authentication using a custom scheme based on HMAC (Hash Message Authentication Code).

Signature Algorithm

There are XML and JSON signatures:

- Use **JSON** signatures for **downloading** operations.
- Use **XML** signatures for the **uploading** and **bucket API** operations. For more information, see XML Request Signature.

- The host in the downloading request header looks like `<BucketName-APPID>.<picture region>.myqcloud.com/<picture name>` , for example, `examplebucket-1250000000.picsh.myqcloud.com/picture.jpeg` .
- The host in the uploading request header looks like `<BucketName-APPID>.pic.<Region>.myqcloud.com` , for example `examplebucket-1250000000.pic.ap-shanghai.myqcloud.com/picture.jpeg` .

Scenarios

Scenario		Applicable Signature
Processing during data	Hotlink protection is disabled	No signature

downloading		verification
	Hotlink protection is enabled	JSON signature
Processing during data uploading	Persistence processing	XML signature
Bucket API operations	Query, enable, delete, and others	XML signature
Content recognition	Detect content related to pornography, politics, violence, and terrorism	XML signature

Signature Tool

The information that is required for generating a signature includes the APPID (such as 1250000000), bucket name (such as examplebucket-ci), and SecretID and SecretKey of the project.

The preceding information can be obtained as follows:

1. Log in to CI Console, and click **Bucket Management** in the left sidebar.
2. Click the bucket that you want to manage to go to the bucket management page.
3. Click **Bucket Configuration** to view the bucket name and bucket ID. Create a bucket if the current project does not contain one yet. For more information, see Creating Buckets.
4. Go to the API key management page in CAM to get the SecretID and SecretKey.

CI follows the same process for computing a signature as COS.

Using Signatures

Signed HTTP requests initiated through RESTful APIs can pass signatures in the following ways:

1. Pass through a standard HTTP Authorization header, such as `Authorization: q-sign-algorithm=sha1&q-ak=...&q-sign-time=1557989753;1557996953&...&q-signature=...`
2. Pass as an HTTP request parameter (be sure to implement UrlEncode), such as `/exampleobject?q-sign-algorithm=sha1&q-ak=...&q-sign-time=1557989753%3B1557996953&...&q-signature=...`

> In the preceding example, `...` is used to substitute the specific signing information.